Ghi chú của một coder

Vũ Anh

Tháng 01 năm 2018

Mục lục

Chương 1

Lời nói đầu

Đọc quyển Deep Learning quá xá hay luôn. Rồi lại đọc SLP 2. Thấy sao các thánh viết hay và chuẩn thế (đấy là lý do các thánh được gọi là ... các thánh chăng =))

Tính đến thời điểm này đã được 2 năm 10 tháng rồi. Quay lại với latex. Thỏa mãn được điều kiện của mình là một tool offline. Mình thích xuất ra pdf (có gì đọc lại hoặc tra cứu cũng dễ).

Hi vọng gắn bó với thằng này được lâu.

Chào từ hồi magizbox.wordpress.com, cái này tồn tại được 77 ngày (hơn 2 tháng) (từ 01/11/2017 đến 17/01/2018)

> Chào Khách,
> Mình là Vũ Anh. Tính đến thời điểm viết bài này thì đã lập trình được 7 năm (lập trình từ hồi năm 2010). Mình thích viết lách, bằng chứng là đã thay đổi host 2 lần datayo.wordpress.com, magizbox.com. Thành tựu ổn nhất hiện tại chỉ có một project underthesea, xếp loại tạm được.
> Blog này chứa những ghi chép loạn cào cào của mình về mọi thứ. Đúng với phong cách "vô tổ chức" của mình. Chắc chắn nó sẽ không hữu ích lắm với bạn. Tuy nhiên, cảm ơn bạn đã ghé qua.
> Nếu Khách quan tâm, thì mình chỉ post bài xàm vào thứ 7 thôi nhé. Những ngày còn lại chỉ post bài nghiêm túc thôi. (bài này quá xàm nhưng được post vào thứ 5 nhé)
> Làm sao để thực hiện blog này <ul> Viết markdown và latex hỗ trợ bởi wordpress Server cho phép lưu ảnh động giphy Vấn đề lưu trữ ảnh: sử dụng tính năng live upload của github.com

Bỏ cái này vì quá chậm. Không hỗ trợ tốt latex (công thức toán và reference). Mình vẫn thích một công cụ offline hơn.

Chào từ hồi magizbox.com, cái này tồn tại được 488 ngày (1 năm 4 tháng. wow) (từ 01/07/2016 đến 01/11/2017)

> Hello World,
> My name is Vu Anh. I'm a developer working at a startup in Hanoi, Vietnam. Coding and writing is fun, so I make this site to share my gists about computer science, data science, and more. It helps me keep my hobby and save information in case I forget. I wish it will be useful for you too.
> PS: I always looking for collaboration. Feel free to contact me via email brother.rain.1024[at]gmail.com
> Magizbox Stories
> Oct 2, 2016: Wow. It's 524th day of my journey. I added some notes in README.md, index.html, changed structure of website. Today I feel like at begin day when I start writing at datayo.wordpress.com blog. In this pass, there are times when I want to make a professional website like tutorialpoints but it doesn't work that way. Because in my heart, I don't want it, I don't to make a professional website. I just love coding, writing and sharing my hobby with people around the world. So today I come back to starting point, I will keep my writing schedule, make some fun stuffs each week.
> In July 2016, I turn to use HTML and mkdocs, and opensource magizbox.
> In March 2015, I start writing blog with wordpress.

Bỏ cái này vì thời gian build quá lằng nhằng. Quản lý dependencies các kiểu rất lâu. Muốn có một cái gì đó giúp viết thật nhanh và đơn giản.

Chào từ hồi datayo.wordpress.com, cái này tồn tại được 489 ngày (1 năm 4 tháng) (từ 01/03/2015 đến 01/07/2016)

> I'm a junior data scientist, working as a researcher in big data team at a company in Vietnam. I love listening music when I'm writing code because it's make me coding better. I love reading books before sleeping because it take me sleep easier and discover the world with data mining via beautiful language R.
> I write this blog because I want to share my hobbies with everybody. I hope you will enjoy it. Feel free to contact me via twitter @rain$_1$024$oremailbrother.rain.1024@gmail.com(Iwillanswerallemailsforsure)foranythingyouwantto$
> In case you find one of my posts could be better, don't hesitate to drop me a line in comment. I'm very appreciated and I will try my best to make it better and better.

Bỏ cái này. Bỏ wordpress. Vì muốn một site interative hơn.

Phần I

Lập trình

Chương 2

Giới thiệu

2.1 Các vấn đề lập trình

I will to do crazy and dummy things, I will rewrite article for basic languages (which tutorialpoints do very goods)

Each language I will cover these concepts:

Table of content

```
code/
 1. introduction
 2. syntax
 3. data structure
 4. oop
 5. networking
 6. os
 7. parallel
 8. event based
 9. error handling
 10. logging
 11. configuration
 12. documentation
 13. test
 14. ui
 15. web
 16. database
 17. ide
 18. package manager
 19. build tool
 20. make module
 21. production (docker)
```

2.1.1 Introduction

Installation (environment, IDE)

Hello world

Courses

Resources

Syntax

variables and expressions
conditional
loops and Iteration
functions
define, use
parameters
scope of variables
anonymous functions
callbacks
self-invoking functions, inner functions
functions that return functions, functions that redefined themselves
closures
naming convention
comment convention

2.1.2 Data Structure

Number
String
Collection
DateTime
Boolean
Object

2.1.3 OOP

Classes Objects
Inheritance
Encapsulation
Abstraction
Polymorphism
For OOP Example: see Python: OOP

2.1.4 Networking

REST (example with chat app sender, receiver, message)

2.1.5 Sample Project Ideas

Guess My Number Game
Create Analog Clock
Create Pong Game
Create flappy bird

2.2 How to ask a question

Focus on questions about an actual problem you have faced. Include details about what you have tried and exactly what you are trying to do.

Ask about...

Specific programming problems

Software algorithms

Coding techniques

Software development tools

Not all questions work well in our format. Avoid questions that are primarily opinion-based, or that are likely to generate discussion rather than answers.

Don't ask about...

Questions you haven't tried to find an answer for (show your work!)

Product or service recommendations or comparisons

Requests for lists of things, polls, opinions, discussions, etc.

Anything not directly related to writing computer programs

2.3 Các vấn đề lập trình

Generic

KISS (Keep It Simple Stupid)

YAGNI

Do The Simplest Thing That Could Possibly Work

Keep Things DRY

Code For The Maintainer

Avoid Premature Optimization

Inter-Module/Class

Minimise Coupling

Law of Demeter

Composition Over Inheritance

Orthogonality

Module/Class

Maximise Cohesion

Liskov Substitution Principle

Open/Closed Principle

Single Responsibility Principle

Hide Implementation Details

Curly's Law

Software Quality Laws

First Law of Software Quality

2.4 Các mô hình lập trình

Main paradigm approaches 1

1. Imperative

Description:

Computation as statements that directly change a program state (datafields)

Main Characteristics:

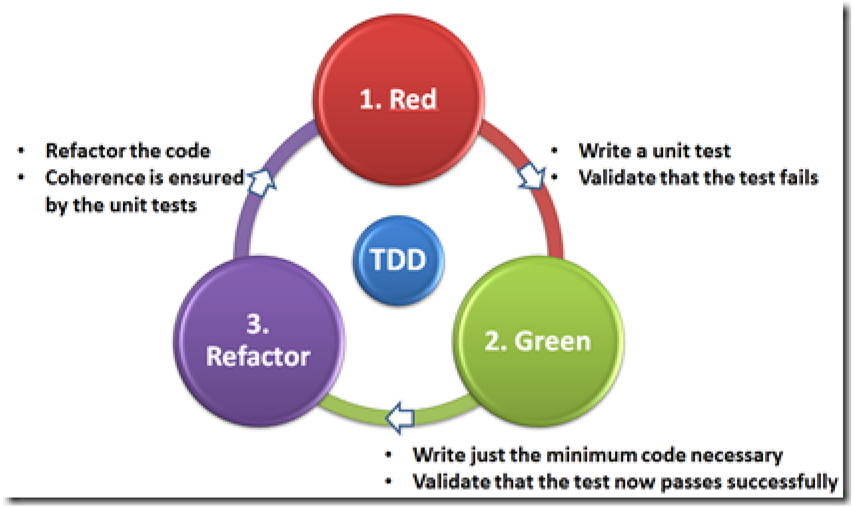
Direct assignments, common data structures, global variables

Critics: Edsger W. Dijkstra, Michael A. Jackson
Examples: Assembly, C, C++, Java, PHP, Python
2. Structured
Description:
A style of imperative programming with more logical program structure
Main Characteristics:
Structograms, indentation, either no, or limited use of, goto statements
Examples: C, C++, Java, Python
3. Procedural
Description:
Derived from structured programming, based on the concept of modular programming or the procedure call
Main Characteristics:
Local variables, sequence, selection, iteration, and modularization
Examples: C, C++, Lisp, PHP, Python
4. Functional
Description:
Treats computation as the evaluation of mathematical functions avoiding state and mutable data
Main Characteristics:
Lambda calculus, compositionality, formula, recursion, referential transparency, no side effects
Examples: Clojure, Coffeescript, Elixir, Erlang, F, Haskell, Lisp, Python, Scala, SequenceL, SML
5. Event-driven including time driven
Description:
Program flow is determined mainly by events, such as mouse clicks or interrupts including timer
Main Characteristics:
Main loop, event handlers, asynchronous processes
Examples: Javascript, ActionScript, Visual Basic
6. Object-oriented
Description:
Treats datafields as objects manipulated through pre-defined methods only
Main Characteristics:
Objects, methods, message passing, information hiding, data abstraction, encapsulation, polymorphism, inheritance, serialization-marshalling
Examples: Common Lisp, C++, C, Eiffel, Java, PHP, Python, Ruby, Scala
7. Declarative
Description:
Defines computation logic without defining its detailed control flow
Main Characteristics:
4GLs, spreadsheets, report program generators
Examples: SQL, regular expressions, CSS, Prolog
8. Automata-based programming
Description:
Treats programs as a model of a finite state machine or any other formal automata
Main Characteristics:

State enumeration, control variable, state changes, isomorphism, state transition table

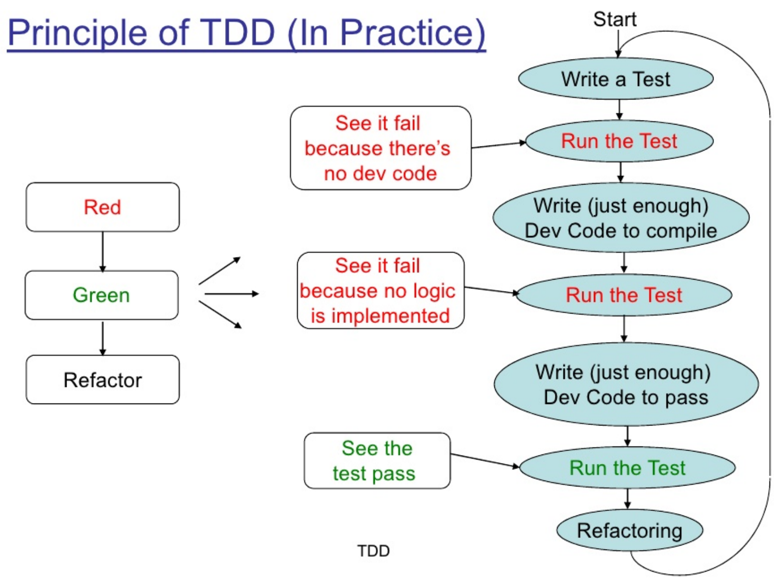
Examples: AsmL

2.5 Testing



1. Definition 1 2

Test-driven development (TDD) is a software development process that relies on the repetition of a very short development cycle:



Step 1: First the developer writes an (initially failing) automated test case

that defines a desired improvement or new function,

Step 2: Then produces the minimum amount of code to pass that test,

Step 3: Finally refactors the new code to acceptable standards.

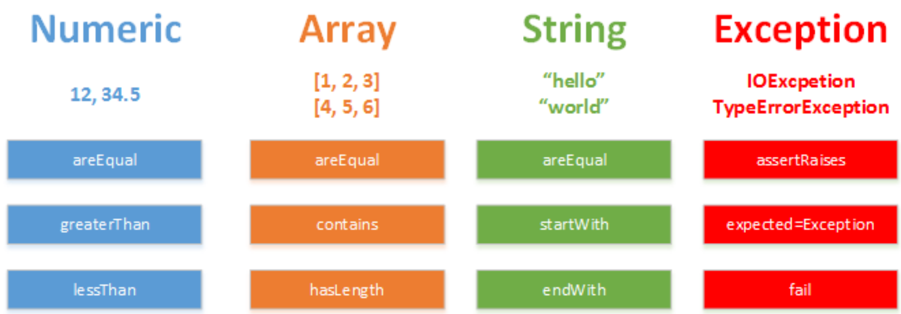
Kent Beck, who is credited with having developed or 'rediscovered' the technique, stated in 2003 that TDD encourages simple designs and inspires confidence.

2. Principles 2

Kent Beck defines

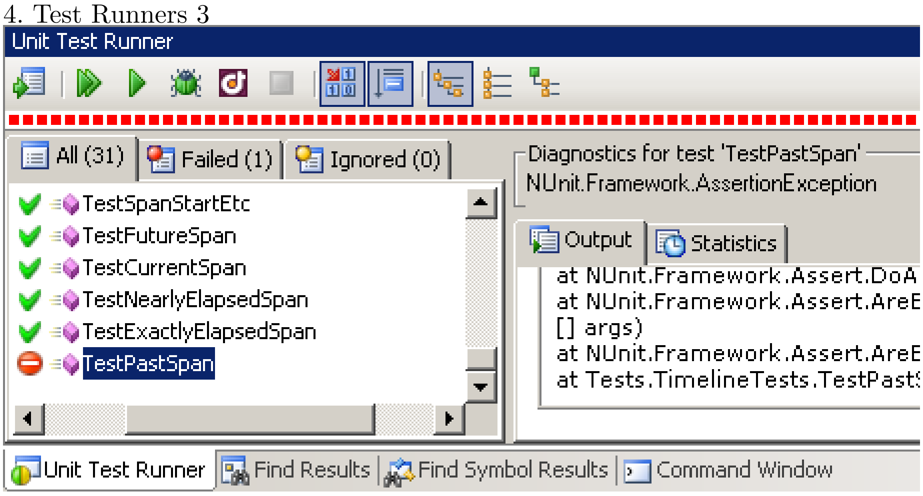
Never with a single line of code unless you have a failing automated test. Eliminate duplication Red: (Automated test fail) Green (Automated test pass because dev code has been written) Refactor (Eliminate duplication, Clean the code)

3. Assertions Assert Framework



Assert that the expected results have occurred. [code lang="java"] @Test public void test() assertEquals(2, 1 + 1); [/code]

4. Test Runners 3



When testing a large real-world web app there may be tens or hundreds of test cases, and we certainly don't want to run each one manually. In such as scenario we need to use a test runner to find and execute the tests for us, and in this article we'll explore just that.

A test runner provides the a good basis for a real testing framework. A test runner is designed to run tests, tag tests with attributes (annotations), and provide reporting and other features.

In general you break your tests up into 3 standard sections; setUp(), tests, and tearDown(), typical for a test runner setup.

The setUp() and tearDown() methods are run automatically for every test, and contain respectively:

The setup steps you need to take before running the test, such as unlocking the screen and killing open apps. The cooldown steps you need to run after the test, such as closing the Marionette session.

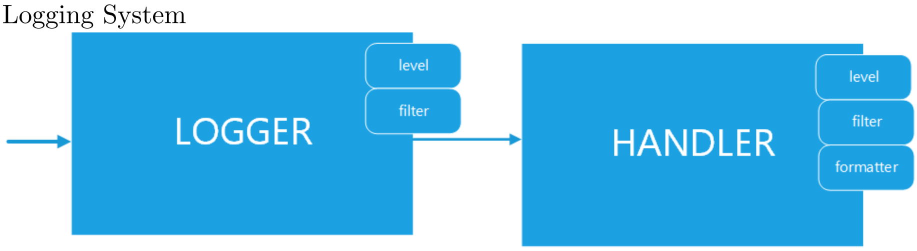
5. Test Frameworks

Language Test Frameworks C++/VisualStudio C++: Test Web Service rest-assured Web UI SeleniumHQ

2.6 Logging

Logging is the process of recording application actions and state to a secondary interface.

Logging System



Levels

Level When it's used DEBUG Detailed information, typically of interest only when diagnosing problems. INFO Confirmation that things are working as expected. WARNING An indication that something unexpected happened, or indicative of some problem in the near future (e.g. 'disk space low'). The software is still working as expected.

ERROR

Due to a more serious problem, the software has not been able to perform some function. CRITICAL A serious error, indicating that the program itself may be unable to continue running. Best Practices 2 4 5 Logging should always be considered when handling an exception but should never take the place of a real handler. Keep all logging code in your production code. Have an ability to enable more/less detailed logging in production, preferably per subsystem and without restarting your program. Make logs easy to parse by grep and by eye. Stick to several common fields at the beginning of each line. Identify time, severity, and subsystem in every line. Clearly formulate the message. Make every log message easy to map to its source code line. If an error happens, try to collect and log as much information as possible. It may take long but it's OK because normal processing has failed anyway. Not having to wait when the same condition happens in production with a debugger attached is priceless.

2.7 Lập trình hàm

Functional Without mutable variables, assignment, conditional

Advantages 1 Most functional languages provide a nice, protected environment, somewhat like JavaLanguage. It's good to be able to catch exceptions instead of having CoreDumps in stability-critical applications. FP encourages safe ways of programming. I've never seen an OffByOne mistake in a functional program, for example... I've seen one. Adding two lengths to get an index but one of them was zero-indexed. Easy to discover though. – AnonymousDonor Functional programs tend to be much more terse than their ImperativeLanguage counterparts. Often this leads to enhanced programmer productivity. FP encourages quick prototyping. As such, I think it is the best software design paradigm for ExtremeProgrammers... but what do I know. FP is modular in the dimension of functionality, where ObjectOrientedProgramming is modular in the dimension of different components. Generic routines (also provided by CeePlusPlus) with easy syntax. ParametricPolymorphism The ability to have your cake and eat it. Imagine you have a complex OO system processing messages - every component might make state changes depending on the message and then forward the message to some objects it has links to. Wouldn't it be just too cool to be able to easily roll back every change if some object deep in the call hierarchy decided the message is flawed? How about having a history of different states? Many housekeeping tasks made for you: deconstructing data structures (PatternMatching), storing variable bindings (LexicalScope with closures), strong typing (TypeInference), * GarbageCollection, storage allocation, whether to use boxed (pointer-to-value) or unboxed (value directly) representation... Safe multithreading! Immutable data structures are not subject to data race conditions, and consequently don't have to be protected by locks. If you are always allocating new objects, rather than destructively manipulating existing ones, the locking can be hidden in the allocation and GarbageCollection system.

2.8 Lập trình song song

Paralell/Concurrency Programming 1. Callback Pattern 2 Callback functions are derived from a programming paradigm known as functional programming. At a fundamental level, functional programming specifies the use of functions as arguments. Functional programming was—and still is, though to a much lesser extent today—seen as an esoteric technique of specially trained, master programmers.

Fortunately, the techniques of functional programming have been elucidated so that mere mortals like you and me can understand and use them with ease. One of the chief techniques in functional programming happens to be callback functions. As you will read shortly, implementing callback functions is as easy as passing regular variables as arguments. This technique is so simple that I wonder why it is mostly covered in advanced JavaScript topics.

[code lang="javascript"] function getN() return 10;

var n = getN();

function getAsyncN(callback) setTimeout(function() callback(10); , 1000);

function afterGetAsyncN(result) var n = 10; console.log(n);

getAsyncN(afterGetAsyncN); [/code]

2. Promise Pattern 1 3 What is a promise? The core idea behind promises is that a promise represents the result of an asynchronous operation.

A promise is in one of three different states:

pending - The initial state of a promise. fulfilled - The state of a promise representing a successful operation. rejected - The state of a promise representing a failed operation. Once a promise is fulfilled or rejected, it is immutable (i.e. it can never change again).

```
function aPromise(message){
  return new Promise(function(fulfill, reject){
    if(message == "success"){
      fulfill("it is a success Promise");
    } if(message == "fail"){
      reject("it is a fail Promise");
    }
  });
}
```

Usage:

```
aPromise("success").then(function(successMessage){
  console.log(successMessage) }, function(failMessage){
  // it is a success Promise
  console.log(failMessage)
})

aPromise("fail").then(function(successMessage){
  console.log(successMessage) }, function(failMessage){
  console.log(failMessage)
}) // it is a fail Promise
```

2.9 IDE

An integrated development environment (IDE) is a software application that provides comprehensive facilities to computer programmers for software development. An IDE normally consists of a source code editor, build automation tools and a debugger. Most modern IDEs have intelligent code completion.

1. Navigation

Word Navigation Line Navigation File Navigation

2. Editing

Auto Complete Code Complete Multicursor Template (Snippets)

3. Formatting

Debugging Custom Rendering for Object

Chương 3

Python

Hướng dẫn online tại http://magizbox.com/training/python/site/

3.1 Giới thiệu

'Python' is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java.

The language provides constructs intended to enable clear programs on both a small and large scale.

Python Tutorial Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

Python is Interpreted

Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

Python is Interactive

You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python is Object-Oriented

Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

Python is Beginner Friendly

Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Audience This tutorial is designed for software programmers who need to learn Python programming language from scratch.

Sách

Tập hợp các sách python

Khoá học
Tập hợp các khóa học python
Tham khảo
Top 10 Python Libraries Of 2015

3.2 Cài đặt

Get Started Welcome! This tutorial details how to get started with Python.

For Windows Anaconda 4.3.0 Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

1. Download the installer 2. Optional: Verify data integrity with MD5 or SHA-256 3. Double-click the .exe file to install Anaconda and follow the instructions on the screen Python 3.6 version 64-BIT INSTALLER Python 2.7 version 64-BIT INSTALLER Step 2. Discover the Map

https://docs.python.org/2/library/index.html

For CentOS Developer tools The Development tools will allow you to build and compile software from source code. Tools for building RPMs are also included, as well as source code management tools like Git, SVN, and CVS.

```
yum groupinstall "Development tools"
yum install zlib-devel
yum install bzip2-devel
yum install openssl-devel
yum install ncurses-devel
yum install sqlite-devel
```

Python Anaconda Anaconda is BSD licensed which gives you permission to use Anaconda commercially and for redistribution.

```
cd /opt
wget --no-check-certificate https://www.python.org/ftp/python
    ↪ /2.7.6/Python-2.7.6.tar.xz
tar xf Python-2.7.6.tar.xz
cd Python-2.7.6
./configure --prefix=/usr/local
make && make altinstall
## link
ln -s /usr/local/bin/python2.7 /usr/local/bin/python
# final check
which python
python -V
# install Anaconda
cd ~/Downloads
wget https://repo.continuum.io/archive/Anaconda-2.3.0-Linux-
    ↪ x86_64.sh
bash ~/Downloads/Anaconda-2.3.0-Linux-x86_64.sh
```

3.3 Cơ bản

3.4 Cú pháp cơ bản

Print, print

```
print "Hello World"
```

Conditional

```
if you_smart:
    print "learn python"
else:
    print "go away"
```

Loop

In general, statements are executed sequentially: The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.

Programming languages provide various control structures that allow for more complicated execution paths. A loop statement allows us to execute a statement or group of statements multiple times. The following diagram illustrates a loop statement

Python programming language provides following types of loops to handle looping requirements.

while loop Repeats a statement or group of statements while a given condition is TRUE. It tests the condition before executing the loop body. for loop Executes a sequence of statements multiple times and abbreviates the code that manages the loop variable. nested loops You can use one or more loop inside any another while, for or do..while loop. While Loop A while loop statement in Python programming language repeatedly executes a target statement as long as a given condition is true.

Syntax

The syntax of a while loop in Python programming language is

```
while expression:
   statement(s)
```

Example

```
count = 0
while count < 9:
   print 'The count is:', count
   count += 1
print "Good bye!"
```

For Loop

It has the ability to iterate over the items of any sequence, such as a list or a string.

Syntax

```
for iterating_var in sequence:
   statements(s)
```

If a sequence contains an expression list, it is evaluated first. Then, the first item in the sequence is assigned to the iterating variable iterating$_{v}ar.Next, thestatementsblockisexecuted.Eac$

Example

```
for i in range(10):
    print "hello", i

for letter in 'Python':
   print 'Current letter :', letter

fruits = ['banana', 'apple', 'mango']
for fruit in fruits:
   print 'Current fruit :', fruit

print "Good bye!"
```

Yield and Generator

Yield is a keyword that is used like return, except the function will return a generator.

```
def createGenerator():
    yield 1
    yield 2
    yield 3
mygenerator = createGenerator() # create a generator
print(mygenerator) # mygenerator is an object!
# <generator object createGenerator at 0xb7555c34>
for i in mygenerator:
    print(i)
# 1
# 2
# 3
```

Visit Yield and Generator explained for more information

Functions

Variable-length arguments

```
def functionname([formal_args,] *var_args_tuple ):
   "function_docstring"
   function_suite
   return [expression]
```

Example

```
#!/usr/bin/python

# Function definition is here
def printinfo( arg1, *vartuple ):
   "This prints a variable passed arguments"
   print "Output is: "
   print arg1
   for var in vartuple:
      print var
```

```
   return;

# Now you can call printinfo function
printinfo( 10 )
printinfo( 70, 60, 50 )
```

Coding Convention Code layout Indentation: 4 spaces

Suggest Readings

"Python Functions". www.tutorialspoint.com "Python Loops". www.tutorialspoint.com "What does the "yield" keyword do?". stackoverflow.com "Improve Your Python: 'yield' and Generators Explained". jeffknupp.com

Vấn đề với mảng

Random Sampling [1] - sinh ra một mảng ngẫu nhiên trong khoảng (0, 1), mảng ngẫu nhiên số nguyên trong khoảng (x, y), mảng ngẫu nhiên là permutation của số từ 1 đến n

3.5 Yield and Generators

Coroutines and Subroutines When we call a normal Python function, execution starts at function's first line and continues until a return statement, exception, or the end of the function (which is seen as an implicit return None) is encountered. Once a function returns control to its caller, that's it. Any work done by the function and stored in local variables is lost. A new call to the function creates everything from scratch.

This is all very standard when discussing functions (more generally referred to as subroutines) in computer programming. There are times, though, when it's beneficial to have the ability to create a "function" which, instead of simply returning a single value, is able to yield a series of values. To do so, such a function would need to be able to "save its work," so to speak.

I said, "yield a series of values" because our hypothetical function doesn't "return" in the normal sense. return implies that the function is returning control of execution to the point where the function was called. "Yield," however, implies that the transfer of control is temporary and voluntary, and our function expects to regain it in the future.

In Python, "functions" with these capabilities are called generators, and they're incredibly useful. generators (and the yield statement) were initially introduced to give programmers a more straightforward way to write code responsible for producing a series of values. Previously, creating something like a random number generator required a class or module that both generated values and kept track of state between calls. With the introduction of generators, this became much simpler.

To better understand the problem generators solve, let's take a look at an example. Throughout the example, keep in mind the core problem being solved: generating a series of values.

Note: Outside of Python, all but the simplest generators would be referred to as coroutines. I'll use the latter term later in the post. The important thing

[1] tham khảo [pytorch](http://pytorch.org/docs/master/torch.html?highlight=randntorch.randn), [numpy](https://docs.scipy.org/doc/numpy-1.13.0/reference/routines.random.html))

to remember is, in Python, everything described here as a coroutine is still a generator. Python formally defines the term generator; coroutine is used in discussion but has no formal definition in the language.

Example: Fun With Prime Numbers Suppose our boss asks us to write a function that takes a list of ints and returns some Iterable containing the elements which are prime1 numbers.

Remember, an Iterable is just an object capable of returning its members one at a time.

"Simple," we say, and we write the following:

```
def get_primes(input_list):
    result_list = list()
    for element in input_list:
        if is_prime(element):
            result_list.append()

    return result_list
```

or better yet...

```
def get_primes(input_list):
    return (element for element in input_list if is_prime(element)
        ↪ )

# not germane to the example, but here's a possible
    ↪ implementation of
# is_prime...

def is_prime(number):
    if number > 1:
        if number == 2:
            return True
        if number % 2 == 0:
            return False
        for current in range(3, int(math.sqrt(number) + 1), 2):
            if number % current == 0:
                return False
        return True
    return False
```

Either get$_{p}rimesimplementationabovefulfillstherequirements, sowetellourbosswe'redone.Shereportso$

Dealing With Infinite Sequences Well, not quite exactly. A few days later, our boss comes back and tells us she's run into a small problem: she wants to use our get$_{p}rimesfunctiononaverylargelistofnumbers.Infact, thelistissolargethatmerelycreatingitwouldconsume$

Once we think about this new requirement, it becomes clear that it requires more than a simple change to get$_{p}rimes.Clearly, wecan'treturnalistofalltheprimenumbersfromstarttoinfi$

Before we give up, let's determine the core obstacle preventing us from writing a function that satisfies our boss's new requirements. Thinking about it, we arrive at the following: functions only get one chance to return results, and thus must return all results at once. It seems pointless to make such an obvious statement; "functions just work that way," we think. The real value lies in asking, "but what if they didn't?"

Imagine what we could do if get$_{p}rimescouldsimplyreturnthenextvalueinsteadofallthevaluesatonce.Itwo$

Unfortunately, this doesn't seem possible. Even if we had a magical function that allowed us to iterate from n to infinity, we'd get stuck after returning the first value:

def get$_{p}rimes(start) : forelementinmagical_{i}nfinite_{r}ange(start) : ifis_{p}rime(element) : returnelementImagineget_{p}rimesiscalledlikeso :$

def solve$_{n}umber_{1}0() : She * is * workingonProjectEuler10, Iknewit!total = 2fornext_{p}rimeinget_{p}rimes(3) : ifnext_{p}rime < 2000000 : total+ = next_{p}rimeelse : print(total)returnClearly, inget_{p}rimes, wewouldimmediatelyhitthecasewherenumber = 3andreturnatline4.Insteadofreturn, weneedawaytogenerateavalueand, whenaskedforthenextone, pickupw$

Functions, though, can't do this. When they return, they're done for good. Even if we could guarantee a function would be called again, we have no way of saying, "OK, now, instead of starting at the first line like we normally do, start up where we left off at line 4." Functions have a single entry point: the first line.

Enter the Generator This sort of problem is so common that a new construct was added to Python to solve it: the generator. A generator "generates" values. Creating generators was made as straightforward as possible through the concept of generator functions, introduced simultaneously.

A generator function is defined like a normal function, but whenever it needs to generate a value, it does so with the yield keyword rather than return. If the body of a def contains yield, the function automatically becomes a generator function (even if it also contains a return statement). There's nothing else we need to do to create one.

generator functions create generator iterators. That's the last time you'll see the term generator iterator, though, since they're almost always referred to as "generators". Just remember that a generator is a special type of iterator. To be considered an iterator, generators must define a few methods, one of which is next(). To get the next value from a generator, we use the same built-in function as for iterators: next().

This point bears repeating: to get the next value from a generator, we use the same built-in function as for iterators: next().

(next() takes care of calling the generator's next() method). Since a generator is a type of iterator, it can be used in a for loop.

So whenever next() is called on a generator, the generator is responsible for passing back a value to whomever called next(). It does so by calling yield along with the value to be passed back (e.g. yield 7). The easiest way to remember what yield does is to think of it as return (plus a little magic) for generator functions.**

Again, this bears repeating: yield is just return (plus a little magic) for generator functions.

Here's a simple generator function:

»> def simple$_{g}enerator_{f}unction() :>>> yield1 >>> yield2 >>> yield3Andherearetwosimplewaystous$

»> for value in simple$_{g}enerator_{f}unction() :>>> print(value)123 >>> our_{g}enerator = simple_{g}enerator_{f}unction() >>> next(our_{g}enerator)1 >>> next(our_{g}enerator)2 >>> next(our_{g}enerator)3Magic?What'sthemagicpart?Gladyouasked!Whenagenera$

Let's rewrite get$_{p}rimesasageneratorfunction.Noticethatwenolongerneedthemagical_{i}nfinite_{r}angefunc$

def get$_{p}rimes(number) : whileTrue : ifis_{p}rime(number) : yieldnumbernumber+ = 1Ifageneratorfunctioncallsreturnorreachestheenditsdefinition, aStopIterationexceptionisraised.Thissig$ $loopispresentinget_{p}rimes.Ifitweren't, thefirsttimenext()wascalledwewouldcheckifthenumberisprimeand$

»> our$_{g}enerator = simple_{g}enerator_{f}unction() >>> forvalueinour_{g}enerator :>>> print(value)$

»> our$_{g}eneratorhasbeenexhausted... >>> print(next(our_{g}enerator))Traceback(mostrecentcalllast) : File" < ipython - input - 13 - 7e48a609051a > ", line1, in < module > next(our_{g}enerator)StopIteration$

»> however, we can always create a new generator »> by calling the generator function again...

»> new$_{g}enerator = simple_{g}enerator_{f}unction() >>> print(next(new_{g}enerator))perfectlyvalid1Thus, t$

Visualizing the flow Let's go back to the code that was calling get$_{p}rimes : solve_{n}umber_{1}0.$

def solve$_{n}umber_{1}0() : She*is*workingonProjectEuler10, Iknewit!total = 2fornext_{p}rimeinget_{p}rimes(3) : ifnext_{p}rime < 2000000 : total+ = next_{p}rimeelse : print(total)returnIt'shelpfultovisualizehowthefirstfewelementsarecreatedwhenwecallget_{p}rimesinsolve_{n}$

We enter the while loop on line 3 The if condition holds (3 is prime) We yield the value 3 and control to solve$_{n}umber_{1}0.Then, backinsolve_{n}umber_{1}0 :$

The value 3 is passed back to the for loop The for loop assigns next$_{p}rimetothisvaluenext_{p}rimeisaddedtotot$

def get$_{p}rimes(number) : whileTrue : ifis_{p}rime(number) : yieldnumbernumber+ = 1 <<<<<<<<<< Mostimportantly, numberstillhasthesamevalueitdidwhenwecalledyield(i.e.3).Rememb$

Moar Power In PEP 342, support was added for passing values into generators. PEP 342 gave generators the power to yield a value (as before), receive a value, or both yield a value and receive a (possibly different) value in a single statement.

To illustrate how values are sent to a generator, let's return to our prime number example. This time, instead of simply printing every prime number greater than number, we'll find the smallest prime number greater than successive powers of a number (i.e. for 10, we want the smallest prime greater than 10, then 100, then 1000, etc.). We start in the same way as get$_{p}rimes :$

def print$_{s}uccessive_{p}rimes(iterations, base = 10) : likenormalfunctions, ageneratorfunctioncanbeassig$

prime$_{g}enerator = get_{p}rimes(base)missingcode...forpowerinrange(iterations) : missingcode...$

def get$_{p}rimes(number) : whileTrue : ifis_{p}rime(number) : ...whatgoeshere?Thenextlineofget_{p}rimestak$
$yieldfoomeans, "yieldfooand, whenavalueissenttome, setothertothatvalue."Youcan"send"valuestoagenera$

def get$_{p}rimes(number) : whileTrue : ifis_{p}rime(number) : number = yieldnumbernumber+ = 1Inthisway, wecansetnumbertoadifferentvalueeachtimethegeneratoryields.Wec$

def print$_{s}uccessive_{p}rimes(iterations, base = 10) : prime_{g}enerator = get_{p}rimes(base)prime_{g}enerator.se$
$print(prime_{g}enerator.send(base**power))Twothingstonotehere : First, we'reprintingtheresultofgenerato$

Second, notice the prime$_{g}enerator.send(None)line.Whenyou'reusingsendto"start"agenerator(thatis, ex$

Round-up In the second half of this series, we'll discuss the various ways in which generators have been enhanced and the power they gained as a result. yield has become one of the most powerful keywords in Python. Now that we've built a solid understanding of how yield works, we have the knowledge necessary to understand some of the more "mind-bending" things that yield can be used for.

Believe it or not, we've barely scratched the surface of the power of yield. For example, while send does work as described above, it's almost never used when generating simple sequences like our example. Below, I've pasted a small demonstration of one common way send is used. I'll not say any more about it as figuring out how and why it works will be a good warm-up for part two.

```
import random
```

```
def get_data():
    """Return 3 random integers between 0 and 9"""
    return random.sample(range(10), 3)

def consume():
    """Displays a running average across lists of integers sent to
        ↪ it"""
    running_sum = 0
    data_items_seen = 0

    while True:
        data = yield
        data_items_seen += len(data)
        running_sum += sum(data)
        print('The running average is {}'.format(running_sum /
            ↪ float(data_items_seen)))

def produce(consumer):
    """Produces a set of values and forwards them to the pre-
        ↪ defined consumer
    function"""
    while True:
        data = get_data()
        print('Produced {}'.format(data))
        consumer.send(data)
        yield

if __name__ == '__main__':
    consumer = consume()
    consumer.send(None)
    producer = produce(consumer)

    for _ in range(10):
        print('Producing...')
        next(producer)
```

Remember... There are a few key ideas I hope you take away from this discussion:

generators are used to generate a series of values yield is like the return of generator functions The only other thing yield does is save the "state" of a generator function A generator is just a special type of iterator Like iterators, we can get the next value from a generator using next() for gets values by calling next() implicitly

3.6 Cấu trúc dữ liệu

3.6.1 Number

Basic Operation

```
1
1.2
1 + 2
abs(-5)
```

3.6.2 Collection

In this post I will cover 4 most popular data types in python list, tuple, set, dictionary

List The most basic data structure in Python is the sequence. Each element of a sequence is assigned a number - its position or index. The first index is zero, the second index is one, and so forth.

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Usage

A list keeps order, dict and set don't: when you care about order, therefore, you must use list (if your choice of containers is limited to these three, of course) Most Popular Operations

Create a list a = ["a", "b", 3] Access values in list a[1] Updated List a[0] = 5 Delete list elements del a[1] Reverse a list a[::-1] Itertools [a + b for (a, b) in itertools.product(x, y)] Select random elements in list random.choice(x) random.sample(x, 3) Create a list a = [1, 2, 3] [1, 2, 3] Access values in list list1 = ['physics', 'chemistry', 1997, 2000] list2 = [1, 2, 3, 4, 5, 6, 7]

print list1[0] physics

print list2[1:5] [2, 3, 4, 5] Updated lists list = ['physics', 'chemistry', 1997, 2000] print list[2] 1997

list[2] = 2001 print list[2] 2001 Delete list elements list1 = ['physics', 'chemistry', 1997, 2000];

print list1 ['physics', 'chemistry', 1997, 2000]

del list1[2]

print list1 ['physics', 'chemistry', 2000] Reverse a list [1, 3, 2][::-1] [2, 3, 1] Itertools import itertools

x = [1, 2, 3] y = [2, 4, 5]

[a + b for (a, b) in itertools.product(x, y)] [3, 5, 6, 4, 6, 7, 5, 7, 8] Select random elements in list import random

x = [13, 23, 14, 52, 6, 23]

random.choice(x) 52

random.sample(x, 3) [23, 14, 52] Tuples A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Usage

Tuples have structure, lists have order Tuples being immutable there is also a semantic distinction that should guide their usage. Tuples are heterogeneous data structures (i.e., their entries have different meanings), while lists are homogeneous sequences Most Popular Operations

Create a tuple t = ("a", 1, 2) Accessing Values in Tuples t[0], t[1:] Updating Tuples Not allowed Create a tuple tup1 = ('physics', 'chemistry', 1997, 2000);

tup2 = (1, 2, 3, 4, 5); tup3 = "a", "b", "c", "d"; tup4 = () tup5 = (50,)
Accessing Values in Tuples !/usr/bin/python

tup1 = ('physics', 'chemistry', 1997, 2000); tup2 = (1, 2, 3, 4, 5, 6, 7);

tup1[0] physics

tup2[1:5] [2, 3, 4, 5] Updating Tuples Tuples are immutable which means you cannot update or change the values of tuple elements. You are able to take portions of existing tuples to create new tuples as the following example demonstrates

tup1 = (12, 34.56); tup2 = ('abc', 'xyz');

Following action is not valid for tuples tup1[0] = 100;

So let's create a new tuple as follows tup3 = tup1 + tup2; print tup3 Set Sets are lists with no duplicate entries.

The sets module provides classes for constructing and manipulating unordered collections of unique elements. Common uses include membership testing, removing duplicates from a sequence, and computing standard math operations on sets such as intersection, union, difference, and symmetric difference.

Usage

set forbids duplicates, list does not: also a crucial distinction. Most Popular Operations

Create a set x = set(["Postcard", "Radio", "Telegram"]) Add elements to a set x.add("Mobile") Remove elements to a set x.remove("Radio") Subset y.issubset(x) Intersection x.intersection(y) Difference between two sets x.difference(y) Create a set x = set(["Postcard", "Radio", "Telegram"]) x set(['Postcard', 'Telegram', 'Radio']) Add elements to a set x = set(["Postcard", "Radio", "Telegram"]) x.add("Mobile") x set(['Postcard', 'Telegram', 'Mobile', 'Radio']) Remove elements to a set x = set(["Postcard", "Radio", "Telegram"]) x.remove("Radio") x set(['Postcard', 'Telegram']) Subset x = set(["a","b","c","d"]) y = set(["c","d"]) y.issubset(x) True Intersection x = set(["a","b","c","d"]) y = set(["c","d"]) x.intersection(y) set(['c', 'd']) Difference between two sets x = set(["Postcard", "Radio", "Telegram"]) y = set(["Radio","Television"]) x.difference(y) set(['Postcard', 'Telegram']) Dictionary Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: .

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

Usage

dict associates with each key a value, while list and set just contain values: very different use cases, obviously. Most Popular Operations

Create a dictionary d = "a": 1, "b": 2, "c": 3 Update dictionary d["a"] = 4 Delete dictionary elements del d["a"] Create a dictionary dict = 'Name': 'Zara', 'Age': 7, 'Class': 'First'

print "dict['Name']: ", dict['Name'] print "dict['Age']: ", dict['Age'] Update dictionary dict = 'Name': 'Zara', 'Age': 7, 'Class': 'First'

dict['Age'] = 8; update existing entry dict['School'] = "DPS School"; Add new entry

print "dict['Age']: ", dict['Age'] print "dict['School']: ", dict['School'] Delete dictionary elements dict = 'Name': 'Zara', 'Age': 7, 'Class': 'First' del dict['Name']; remove entry with key 'Name' dict.clear(); remove all entries in dict del dict ; delete entire dictionary

print "dict['Age']: ", dict['Age'] print "dict['School']: ", dict['School'] Related Readings Python Lists, tutorialspoint.com Python Dictionary, tutorialspoint.com Python Dictionary Methods, guru99 In Python, when to use a Dictionary, List or Set?, stackoverflow What's the difference between lists and tuples?, stackoverflow

3.6.3 String

Format '0, 1, 2'.format('a', 'b', 'c') 'a, b, c' Regular Expressions The aim of this chapter of our Python tutorial is to present a detailed led and descriptive introduction into regular expressions. This introduction will explain the theoretical aspects of regular expressions and will show you how to use them in Python scripts.

Regular Expressions are used in programming languages to filter texts or textstrings. It's possible to check, if a text or a string matches a regular expression.

There is an aspect of regular expressions which shouldn't go unmentioned: The syntax of regular expressions is the same for all programming and script languages, e.g. Python, Perl, Java, SED, AWK and even X.

Functions match function This function attempts to match RE pattern to string with optional flags.

re.match(pattern, string, flags=0) Example

import re

line = "Cats are smarter than dogs"

matched$_object = re.match(r'(.*)are(.*?).*', line, re.M|re.I)$

if matched$_object : print"matched_object.group() : ", matched_object.group()print"matched_object.group(1)$
$", matched_object.group(1)print"matched_object.group(2) : ", matched_object.group(2)else :$
$print"Nomatch!!"Whenthecodeisexecuted, itproducesfollowingresults$

matched$_object.group() : Catsaresmarterthandogsmatched_object.group(1) :$
$Catsmatched_object.group(2) : smartersearchfunctionThisfunctionsearchesforfirstoccurrenceofREpatte$

re.search(pattern, string, flags=0) Example

!/usr/bin/python import re

line = "Cats are smarter than dogs"

search$_object = re.search(r'dogs', line, re.M|re.I)ifsearch_object : print"search-$
$- > search_object.group() : ", search_object.group()else : print"Nothingfound!!"Whenthecodeisexecuted, it$

search −> search$_object.group() : dogssubfunctionThismethodreplacesalloccurrencesoftheREpatternins$

re.sub(pattern, repl, string, max=0) Example

!/usr/bin/python import re

phone = "2004-959-559 This is Phone Number"

Delete Python-style comments num = re.sub(r'.*$', "", phone)print"PhoneNum :$
$", num$

Remove anything other than digits num = re.sub(r", "", phone) print "Phone Num : ", num When the code is executed, it produces following results

Phone Num : 2004-959-559 Phone Num : 2004959559 Tokens Cheatsheet Character Classes . any character except newline /go.gle/ google goggle gogle word, digit, whitespace // AaYyz09 ?! // 012345 aZ? // 0123456789 abcd?/ §not word, digit, whitespace // abcded 1234 ?> // abc 12345 ?<. /§/ abc 123? <. [abc] any of a, b or c /analy[sz]e/ analyse analyze analyxe [$^abc]nota, borc/analy[^sz]e/analyseanalyzeanalyxe[$

$g]characterbetweenag/[2-4]/demo1demo2demo3demo4demo5QuantifiersAlternationa*$ $a+a?0ormore, 1ormore, 0or1/go*gle/goglegoglegooglegoooooglehgle/go+gle/gglegoglegooglegoooooglehgle/$
start / end of the string /$^a bc$/ abc /$^a bc/abcabc/abc$/ abc abc _word, not-word boundary //_ This island is beautiful. // cat certificate Escaped characters ˙ escaped special characters //˙ username@exampe.com 300.000 USD // abc@/ / abc@ t̊ab, linefeed, carriage return //̂ abc def /ab/ ab //̊ abc@ŭ00A9 unicode escaped © /ŭ00A9/ Copyright©2017 - All rights reserved Groups and Lockaround (abc) capture group /(demo|example)[0-9]/ demo1example4demo backreference to group 1 /(abc|def)=/ abc=abc def=defabc=def (?:abc) non-capturing group /(?:abc)3/ abcabcabc abcabc (?=abc) positive lookahead /t(?=s)/ tttsssttts (?!abc) negative lookahead /t(?!s)/ tttsssttts (?<=abc) positive lookbehind /(?<=foo)bar/ foobar fuubar (?<!abc) negative lookbehind /(?<!foo)bar/ foobar fuubar Related Readings

Online regex tester and debugger: PHP, PCRE, Python, Golang and JavaScript, regex101.com RegExr: Learn, Build, Test RegEx, regexr.com

3.6.4 Datetime

Print current time

from datetime import datetime datetime.now().strftime(' '2015-12-29 14:02:27' Get current time

import datetime datetime.datetime.now() datetime(2009, 1, 6, 15, 8, 24, 78915) Unixtime

import time int(time.time()) Measure time elapsed

import time

start = time.time() print("hello") end = time.time() print(end - start) Moment Dealing with dates in Python shouldn't have to suck.

Installation

pip install moment Usage

import moment from datetime import datetime

Create a moment from a string moment.date("12-18-2012")

Create a moment with a specified strftime format moment.date("12-18-2012", "

Moment uses the awesome dateparser library behind the scenes moment.date("2012-12-18")

Create a moment with words in it moment.date("December 18, 2012")

Create a moment that would normally be pretty hard to do moment.date("2 weeks ago")

Create a future moment that would otherwise be really difficult moment.date("2 weeks from now")

Create a moment from the current datetime moment.now()

The moment can also be UTC-based moment.utcnow()

Create a moment with the UTC time zone moment.utc("2012-12-18")

Create a moment from a Unix timestamp moment.unix(1355875153626)

Create a moment from a Unix UTC timestamp moment.unix(1355875153626, utc=True)

Return a datetime instance moment.date(2012, 12, 18).date

We can do the same thing with the UTC method moment.utc(2012, 12, 18).date

Create and format a moment using Moment.js semantics moment.now().format("YYYY-M-D")

Create and format a moment with strftime semantics moment.date(2012, 12, 18).strftime("

Update your moment's time zone moment.date(datetime(2012, 12, 18)).locale("US/Central").date

Alter the moment's UTC time zone to a different time zone moment.utcnow().timezone("US/Eastern").dat

Set and update your moment's time zone. For instance, I'm on the west coast, but want NYC's current time. moment.now().locale("US/Pacific").timezone("US/Eastern")

In order to manipulate time zones, a locale must always be set or you must be using UTC. moment.utcnow().timezone("US/Eastern").date

You can also clone a moment, so the original stays unaltered now = moment.utcnow().timezone("US/Pacific") future = now.clone().add(weeks=2) Related Readings How to get current time in Python, stackoverflow Does Python's time.time() return the local or UTC timestamp?, stackoverflow Measure time elapsed in Python?, stackoverflow momnet, https://github.com/zachwill/moment

3.6.5 Object

Convert dict to object Elegant way to convert a normal Python dict with some nested dicts to an object

class Struct: def $_{init}{}_{(self,**entries):self.}{}_{dict}{}_{.update(entries)Then,youcanuse}$

$>$ args = 'a': 1, 'b': 2 $>$ s = Struct(**args) $>$ s $<$ $_{main}{}_{.Structinstanceat0x01D6A738>>s.a1>s.b2RelatedReadings}$
stackoverflow, Convert Python dict to object?

3.7 Object Oriented Programming

Object Oriented Programming Python has been an object-oriented language since it existed. Because of this, creating and using classes and objects are downright easy. This chapter helps you become an expert in using Python's object-oriented programming support.

If you do not have any previous experience with object-oriented (OO) programming, you may want to consult an introductory course on it or at least a tutorial of some sort so that you have a grasp of the basic concepts.

Classes and Objects Classes can be thought of as blueprints for creating objects. When I define a BankAccount class using the class keyword, I haven't actually created a bank account. Instead, what I've created is a sort of instruction manual for constructing "bank account" objects. Let's look at the following example code:

class BankAccount: id = None balance = 0

def $_{init}{}_{(self,id,balance=0):self.id=idself.balance=balance}$

def $_{get_balance(self):returnself.balance}$

def withdraw(self, amount): self.balance = self.balance - amount

def deposit(self, amount): self.balance = self.balance + amount

john = BankAccount(1, 1000.0) john.withdraw(100.0) The class BankAccount line does not create a new bank account. That is, just because we've defined a BankAcount doesn't mean we've created on; we've merely outlined the blueprint to create a BankAccount object. To do so, we call the class's $_{init}{}_{methodwiththepropernumberofarguments(minusself,whichwe'llgettoinamoment)}$

So, to use the "blueprint" that we crated by defining the class BankAccount (which is used to create BankAccount objects), we call the class name almost as if it were a function: john = BankAccount(1, 1000.0). This line simple say "use the BankAccount blueprint to create me a new object, which I'll refer to as john".

The john object, known as an instance, is the realized version of the BankAccount class. Before we called BankAccount(), no BankAccount object existed. We can, of course, create as many BankAccount objects as we'd like. There is still, however, only one BankAccount class, regardless of how many instances of the class we create.

self So what's with that self parameter to all of the BankAccount methods? What is it? Why, it's the instance, of course! Put another way, a method like withdraw defines the instructions for withdrawing money from some abstract customer's account. Calling john.withdraw(100) puts those instructions to use on the john instance.

So when we say def withdraw(self, amount):, we're saying, "here's how you withdraw money from a BankAccount object (which we'll call self) and a dollar figure (which we'll call amount). self is the instance of the BankAccount that withdraw is being called on. That's not me making analogies, either. john.withdraw(100.0) is just shorthand for BankAccount.withdraw(john, 100.0), which is perfectly valid (if not often seen) code.

Constructors: $_init_{selfmaymakesenseforothermethods,butwhatabout}{}_init_{?Whenwecall}{}_init_{,we'reintheprocessofcreatinganobject,sohow}$

This is why when we call $_init_{,weinitializeobjectsbysayingthingslikeself.id=id.Remember,sinceselfistheinstance,thisisequivalentto}$

Be careful what you $_init_{After}{}_init_{hasfinished,thecallercanrightlyassumethattheobjectisreadytouse.Thatis,afterjohn=BankAccount}$

Inheritance While Object-oriented Programming is useful as a modeling tool, it truly gains power when the concept of inheritance is introduced. Inheritance is the process by which a "child" class derives the data and behavior of a "parent" class. An example will definitely help us here.

Imagine we run a car dealership. We sell all types of vehicles, from motorcycles to trucks. We set ourselves apart from the competition by our prices. Specifically, how we determine the price of a vehicle on our lot: $5,000xnumberofwheelsavehiclehas.Welovebuyingbac$
10

If we wanted to create a sales system for our dealership using Object-oriented techniques, how would we do so? What would the objects be? We might have a Sale class, a Customer class, an Inventory class, and so forth, but we'd almost certainly have a Car, Truck, and Motorcycle class.

What would these classes look like? Using what we've learned, here's a possible implementation of the Car class:

class Car(object): def $_init_{(self,wheels,miles,make,model,year,sold_on):self.wheels=wheelsself.miles=milesself.make=makeself.mod}$

def sale$_price(self):ifself.sold_onisnotNone:return0.0Alreadysoldreturn5000.0*$
$self.wheels$

def purchase$_price(self):ifself.sold_onisNone:return0.0Notyetsoldreturn8000-$
$(.10*self.miles)OK,thatlooksprettyreasonable.Ofcourse,wewouldlikelyhaveanumberofothermethodsonth$
$sale_priceandpurchase_price.We'llseewhythesearei mportantinabit.$

Now that we've got the Car class, perhaps we should create a Truck class? Let's follow the same pattern we did for car:

class Truck(object): def $_init_{(self,wheels,miles,make,model,year,sold_on):self.wheels=wheelsself.miles=milesself.make=makeself.m}$

def sale$_{p}rice(self) : ifself.sold_{o}nisnotNone : return0.0Alreadysoldreturn5000.0*$ $self.wheels$

def purchase$_{p}rice(self) : ifself.sold_{o}nisNone : return0.0Notyetsoldreturn10000-$ $(.10*self.miles)Wow.That'salmostidenticaltothecarclass.Oneofthemostimportantrulesofprogramming(i$

So what gives? Where did we go wrong? Our main problem is that we raced straight to the concrete: Car and Truck are real things, tangible objects that make intuitive sense as classes. However, they share so much data and functionality in common that it seems there must be an abstraction we can introduce here. Indeed there is: the notion of Vehicle.

Abstract Classes A Vehicle is not a real-world object. Rather, it is a concept that some real-world objects (like cars, trucks, and motorcycles) embody. We would like to use the fact that each of these objects can be considered a vehicle to remove repeated code. We can do that by creating a Vehicle class:

class Vehicle(object): base$_{s}ale_{p}rice = 0$

def $_{i}nit_{(self,wheels,miles,make,model,year,sold_{o}n):self.wheels=wheelsself.miles=milesself.make=makeself.model=modelself.year=year}$

def sale$_{p}rice(self) : ifself.sold_{o}nisnotNone : return0.0Alreadysoldreturn5000.0*$ $self.wheels$

def purchase$_{p}rice(self) : ifself.sold_{o}nisNone : return0.0Notyetsoldreturnself.base_{s}ale_{p}rice-$ $(.10*self.miles)NowwecanmaketheCarandTruckclassinheritfromtheVehicleclassbyreplacingobjectinthel$

We can now define Car and Truck in a very straightforward way:

class Car(Vehicle):

def $_{i}nit_{(self,wheels,miles,make,model,year,sold_{o}n):self.wheels=wheelsself.miles=milesself.make=makeself.model=modelself.year=year}$

class Truck(Vehicle):

def $_{i}nit_{(self,wheels,miles,make,model,year,sold_{o}n):self.wheels=wheelsself.miles=milesself.make=makeself.model=modelself.year=year}$

class Struct: def $_{i}nit_{(self,**entries):self.{_d}ict_{.}update(entries)Then,youcanuse}$

$>$ args = 'a': 1, 'b': 2 $>$ s = Struct(**args) $>$ s $<$ $_{m}ain_{.Structinstanceat0x01D6A738>>s.a1>s.b2SuggestedReadingsImproveY}$

3.7.1 Metaclasses

Metaclasses Python, Classes, and Objects Most readers are aware that Python is an object-oriented language. By object-oriented, we mean that Python can define classes, which bundle data and functionality into one entity. For example, we may create a class IntContainer which stores an integer and allows certain operations to be performed:

class IntContainer(object): def $_{i}nit_{(self,i):self.i=int(i)}$

def add$_{o}ne(self) : self.i+ = 1ic = IntContainer(2)ic.add_{o}ne()print(ic.i)3Thisisabitofasillyexample, bu$ $theirabilitytobundledataandoperationsintoasingleobject, whichleadstocleaner, moremanageable, andmorea$ $orientedapproachtoprogrammingcanbeveryintuitiveandpowerful.$

What many do not realize, though, is that quite literally everything in the Python language is an object.

For example, integers are simply instances of the built-in int type:

print type(1) <type 'int'> To emphasize that the int type really is an object, let's derive from it and specialize the $_{a}dd_{m}ethod(whichisthemachineryunderneaththe+operator):$

(Note: We'll used the super syntax to call methods from the parent class: if you're unfamiliar with this, take a look at this StackOverflow question).

class MyInt(int): def $_{a}dd_{(self,other):print"specializingaddition"returnsuper(MyInt,self).{_a}dd_{(other)}}$

i = MyInt(2) print(i + 2) specializing addition 4 Using the + operator on our derived type goes through our $_{a}dd_{m}ethod,asexpected.Weseethatintreallyisanobjectthatcanbesubclassedandextendedjustlikeuser-defin$

Down the Rabbit Hole: Classes as Objects We said above that everything in python is an object: it turns out that this is true of classes themselves. Let's look at an example.

We'll start by defining a class that does nothing

class DoNothing(object): pass If we instantiate this, we can use the type operator to see the type of object that it is:

d = DoNothing() type(d) $_{main_{.DoNothingWeseethatourvariabledisaninstanceoftheclass}}{}_{main_{.DoNothing.}}$

We can do this similarly for built-in types:

L = [1, 2, 3] type(L) list A list is, as you may expect, an object of type list.

But let's take this a step further: what is the type of DoNothing itself?

type(DoNothing) type The type of DoNothing is type. This tells us that the class DoNothing is itself an object, and that object is of type type.

It turns out that this is the same for built-in datatypes:

type(tuple), type(list), type(int), type(float) (type, type, type, type) What this shows is that in Python, classes are objects, and they are objects of type type. type is a metaclass: a class which instantiates classes. All new-style classes in Python are instances of the type metaclass, including type itself:

type(type) type Yes, you read that correctly: the type of type is type. In other words, type is an instance of itself. This sort of circularity cannot (to my knowledge) be duplicated in pure Python, and the behavior is created through a bit of a hack at the implementation level of Python.

Metaprogramming: Creating Classes on the Fly Now that we've stepped back and considered the fact that classes in Python are simply objects like everything else, we can think about what is known as metaprogramming. You're probably used to creating functions which return objects. We can think of these functions as an object factory: they take some arguments, create an object, and return it. Here is a simple example of a function which creates an int object:

def int$_factory(s) : i = int(s)returni$

i = int$_factory('100')print(i)100Thisisoverly-simplistic, butanyfunctionyouwriteinthecourseofanorma$
$takesomearguments, dosomeoperations, andcreatereturnanobject.Withtheabovediscussioninmind, though,$
$-thisisametafunction :$

def class$_factory() : classFoo(object) : passreturnFoo$

F = class$_factory()f = F()print(type(f)) < class'_{main_{.Foo'>Justasthefunctionint_factoryconstructsanreturnsaninstance}}$

But the above construction is a bit awkward: especially if we were going to do some more complicated logic when constructing Foo, it would be nice to avoid all the nested indentations and define the class in a more dynamic way. We can accomplish this by instantiating Foo from type directly:

def class$_factory() : returntype('Foo', (),)$

F = class$_factory()f = F()print(type(f)) < class'_{main_{.Foo'>Infact,theconstruct}}$

class MyClass(object): pass is identical to the construct

MyClass = type('MyClass', (),) MyClass is an instance of type type, and that can be seen explicitly in the second version of the definition. A potential confusion arises from the more common use of type as a function to determine the type of an object, but you should strive to separate these two uses of the keyword in your mind: here type is a class (more precisely, a metaclass), and MyClass is an instance of type.

The arguments to the type constructor are: type(name, bases, dct) - name is a string giving the name of the class to be constructed - bases is a tuple giving the parent classes of the class to be constructed - dct is a dictionary of the attributes and methods of the class to be constructed

So, for example, the following two pieces of code have identical results:

class Foo(object): i = 4

class Bar(Foo): def get$_i(self) : return self.i$

b = Bar() print(b.get$_i$())$4 Foo = type('Foo', (), dict(i = 4))$

Bar = type('Bar', (Foo,), dict(get$_i = lambda self : self.i$))

b = Bar() print(b.get$_i$())$4 This perhaps seems a bit over-complicated in the case of this contrived example, bu$ $the - fly.$

Making Things Interesting: Custom Metaclasses Now things get really fun. Just as we can inherit from and extend a class we've created, we can also inherit from and extend the type metaclass, and create custom behavior in our metaclass.

Example 1: Modifying Attributes Let's use a simple example where we want to create an API in which the user can create a set of interfaces which contain a file object. Each interface should have a unique string ID, and contain an open file object. The user could then write specialized methods to accomplish certain tasks. There are certainly good ways to do this without delving into metaclasses, but such a simple example will (hopefully) elucidate what's going on.

First we'll create our interface meta class, deriving from type:

class InterfaceMeta(type): def $_{new}{}_{(cls,name,parents,dct):create a class_i d if it's not specified if 'class_i d' not in dct:dct['class_i d']=name}$

open the specified file for writing if 'file' in dct: filename = dct['file'] dct['file'] = open(filename, 'w')

we need to call type.$_{new}{}_{to complete the initialization return super(InterfaceMeta,cls).}{}_{new}{}_{(cls,name,parents,dct) Notice that we've modi}$

Now we'll use our InterfaceMeta class to construct and instantiate an Interface object:

Interface = InterfaceMeta('Interface', (), dict(file='tmp.txt'))

print(Interface.class$_i d) print(Interface.file) interface < open file 'tmp.txt', mode 'w' at 0x21b8810 >$ $This behaves as we'd expect : the class_i d class variable is created, and the file class variable is replaced with an open$

class Interface(object): $_metaclass{}_{= InterfaceMeta file='tmp.txt'}$

print(Interface.class$_i d) print(Interface.file) interface < open file 'tmp.txt', mode 'w' at 0x21b8ae0 >$ $by defining the_metaclass{}_{attribute of the class, we've told the class that it should be constructed using InterfaceMeta rather than using type. To make}$

type(Interface) $_main{}_{.InterfaceMeta Furthermore, any class derived from Interface will now be constructed using the same metaclass:}$

class UserInterface(Interface): file = 'foo.txt'

print(UserInterface.file) print(UserInterface.class$_i d) < open file 'foo.txt', mode 'w' at 0x21b8c00 >$ $user interface This simple example shows how metaclasses can be used to create powerful and flexible APIs for pr$

Example 2: Registering Subclasses Another possible use of a metaclass is to automatically register all subclasses derived from a given base class. For example, you may have a basic interface to a database and wish for the user to be able to define their own interfaces, which are automatically stored in a master registry.

You might proceed this way:

class DBInterfaceMeta(type): we use $_{init}{}_{rather than}{}_{new}{}_{here because we want to modify attributes of the class * after * they have been}$

super(DBInterfaceMeta, cls).$_{init}{}_{(name,bases,dct) Our metaclass simply adds a registry dictionary if it's not already present, and adds the}$

class DBInterface(object): $_metaclass{}_{= DBInterfaceMeta}$

print(DBInterface.registry) Now let's create some subclasses, and double-check that they're added to the registry:

class FirstInterface(DBInterface): pass

class SecondInterface(DBInterface): pass
class SecondInterfaceModified(SecondInterface): pass
print(DBInterface.registry) 'firstinterface': <class 'main.FirstInterface'>,'secondinterface':<class'main.SecondInterface

Conclusion: When Should You Use Metaclasses? I've gone through some examples of what metaclasses are, and some ideas about how they might be used to create very powerful and flexible APIs. Although metaclasses are in the background of everything you do in Python, the average coder rarely has to think about them.

But the question remains: when should you think about using custom metaclasses in your project? It's a complicated question, but there's a quotation floating around the web that addresses it quite succinctly:

Metaclasses are deeper magic than 99

– Tim Peters

In a way, this is a very unsatisfying answer: it's a bit reminiscent of the wistful and cliched explanation of the border between attraction and love: "well, you just... know!"

But I think Tim is right: in general, I've found that most tasks in Python that can be accomplished through use of custom metaclasses can also be accomplished more cleanly and with more clarity by other means. As programmers, we should always be careful to avoid being clever for the sake of cleverness alone, though it is admittedly an ever-present temptation.

I personally spent six years doing science with Python, writing code nearly on a daily basis, before I found a problem for which metaclasses were the natural solution. And it turns out Tim was right:

I just knew.

3.7.2 Design Patterns

Design Patterns Singleton Non-thread-safe Paul Manta's implementation of singletons

@Singleton class Foo: def init(self):print'Foocreated'

f = Foo() Error, this isn't how you get the instance of a singleton

f = Foo.Instance() Good. Being explicit is in line with the Python Zen g = Foo.Instance() Returns already created instance

print f is g True

class Singleton: """ A non-thread-safe helper class to ease implementing singletons. This should be used as a decorator – not a metaclass – to the class that should be a singleton.

The decorated class can define one 'init'functionthattakesonlythe'self'argument.Also,thedecoratedclasscannotbeinheritedfrom.

To get the singleton instance, use the 'Instance' method. Trying to use 'call'willresultina'TypeError'beingraised.

"""

def init(self,decorated):self.decorated=decorated

def Instance(self): """ Returns the singleton instance. Upon its first call, it creates a new instance of the decorated class and calls its 'init'method.Onallsubsequentcalls,thealreadycreatedinstanceisreturn

""" try: return self.instanceexceptAttributeError : self.instance = self.decorated()returnself.instance

def call(self):raiseTypeError('Singletonsmustbeaccessedthrough'Instance()'.')

def instancecheck(self,inst):returnisinstance(inst,self.decorated)Threadsafeweredediver'simplementationofsingletons.Athreadsafeimple import threading

Based on tornado.ioloop.IOLoop.instance() approach. See https://github.com/facebook/tornado class SingletonMixin(object): $_{singleton_lock=threading.Lock()_singleton_instance=None}$

@classmethod def instance(cls): if not cls.$_{singleton_instance:withcls._{singleton_lock:ifnotcls._{singleton_instance:cls._{singleton_i}}}}$

class A(SingletonMixin): pass

class B(SingletonMixin): pass

if $_{name}{}_{=='}{}_{main}{}_{':a,a2=A.instance(),A.instance()b,b2=B.instance(),B.instance()}$

assert a is a2 assert b is b2 assert a is not b

print('a: print('b: Suggested Readings Is there a simple, elegant way to define singletons?

3.8 File System IO

JSON Write json file with pretty format and unicode

import json import io

data = "menu": "header": "Sample Menu", "items": ["id": "Open", "id": "OpenNew", "label": "Open New", None, "id": "Help", "id": "About", "label": "About Adobe CVG Viewer..."]

with io.open("sample$_json.json", "w", encoding = "utf8")asf : content = json.dumps(data, indent = 4, sort_keys = True, ensure_ascii = False)f.write(unicode(content))Result$

"menu": "header": "Sample Menu", "items": ["id": "Open" , "id": "OpenNew", "label": "Open New" , null, "id": "Help" , "id": "About", "label": "About Adobe CVG Viewer..."] Read json file

import json from pprint import pprint

with open('sample$_json.json')asdata_file : data = json.load(data_file)$

pprint(data) Result

u'menu': u'header': u'Sample Menu', u'items': [u'id': u'Open', u'id': u'OpenNew', u'label': u'Open New', None, u'id': u'Help', u'id': u'About', u'label': u'About Adobe CVG Viewer...'] Related Reading

Parsing values from a JSON file in Python, stackoverflow How do I write JSON data to a file in Python?, stackoverflow XML Write xml file with lxml package

import lxml.etree as ET root declaration root = ET.Element('catalog') insert comment comment = ET.Comment(' this is a xml sample file ') root.insert(1, comment) book element book = ET.SubElement(root, 'book', id="bk001") book data author = ET.SubElement(book, 'author') author.text = "Gambardella, Matthew" title = ET.SubElement(book, 'title') title.text = "XML Developer's Guide" write xml to file tree = ET.ElementTree(root) tree.write("sample$_book.xml", pretty_print = True, xml_declaration = True, encoding =' utf-8')Result$

<?xml version='1.0' encoding='UTF-8'?> <catalog> <!– this is a xml sample file –> <book id="bk001"> <author>Gambardella, Matthew</author> <title>XML Developer's Guide</title> </book> </catalog> Read xml file with lxml package

from lxml import etree as ET

tree = ET.parse("sample$_book.xml")root = tree.getroot()book = root.find('book')print"BookInformatio$ "$, book.attrib["id"]print"Author : ", book.find('author').textprint"Title : ", book.find('title').textResult$

Book Information ID : bk001 Author : Gambardella, Matthew Title : XML Developer's Guide

3.9 Operating System

File Operations Copy folder 1

import shutil shutil.copyfile("src", "dst") CLI shutil — High-level file operations

3.10 Networking

REST JSON 1 2 GET

import requests url = "http://localhost:8080/messages" response = requests.get(url) data = response.json() POST 3

import requests import json

url = "http://localhost:8080/messages" data = 'sender': 'Alice', 'receiver': 'Bob', 'message': 'Hello!' headers = 'Content-type': 'application/json', 'Accept': 'application/json' r = requests.post(url, data=json.dumps(data), headers=headers)

3.11 Concurrency and Parallelism

Running several threads is similar to running several different programs concurrently, but with the following benefits

Multiple threads within a process share the same data space with the main thread and can therefore share information or communicate with each other more easily than if they were separate processes. Threads sometimes called light-weight processes and they do not require much memory overhead; they are cheaper than processes. A thread has a beginning, an execution sequence, and a conclusion. It has an instruction pointer that keeps track of where within its context it is currently running.

It can be pre-empted (interrupted) It can temporarily be put on hold (also known as sleeping) while other threads are running - this is called yielding. Starting a New Thread To spawn another thread, you need to call following method available in thread module:

thread.start$_new_thread(function, args[, kwargs])Thismethodcallenablesafastandefficientwaytocreaten$

The method call returns immediately and the child thread starts and calls function with the passed list of args. When function returns, the thread terminates.

Here, args is a tuple of arguments; use an empty tuple to call function without passing any arguments. kwargs is an optional dictionary of keyword arguments.

Example

!/usr/bin/python

import thread import time

Define a function for the thread def print$_time(threadName, delay) : count = 0whilecount < 5 : time.sleep(delay)count+ = 1print"$

Create two threads as follows try: thread.start$_new_thread(print_time, ("Thread-1", 2,))thread.start_new_thread(print_time, ("Thread-2", 4,))except : print"Error : unabletostartthread"$

while 1: pass When the above code is executed, it produces the following result

Thread-1: Thu Jan 22 15:42:17 2009 Thread-1: Thu Jan 22 15:42:19 2009 Thread-2: Thu Jan 22 15:42:19 2009 Thread-1: Thu Jan 22 15:42:21 2009 Thread-2: Thu Jan 22 15:42:23 2009 Thread-1: Thu Jan 22 15:42:23 2009 Thread-1: Thu Jan 22 15:42:25 2009 Thread-2: Thu Jan 22 15:42:27 2009 Thread-2: Thu Jan 22 15:42:31 2009 Thread-2: Thu Jan 22 15:42:35 2009 Although it is very effective for low-level threading, but the thread module is very limited compared to the newer threading module.

The Threading Module The newer threading module included with Python 2.4 provides much more powerful, high-level support for threads than the thread module discussed in the previous section.

The threading module exposes all the methods of the thread module and provides some additional methods:

threading.activeCount(): Returns the number of thread objects that are active. threading.currentThread(): Returns the number of thread objects in the caller's thread control. threading.enumerate(): Returns a list of all thread objects that are currently active. In addition to the methods, the threading module has the Thread class that implements threading. The methods provided by the Thread class are as follows:

run(): The run() method is the entry point for a thread. start(): The start() method starts a thread by calling the run method. join([time]): The join() waits for threads to terminate. isAlive(): The isAlive() method checks whether a thread is still executing. getName(): The getName() method returns the name of a thread. setName(): The setName() method sets the name of a thread. Creating Thread Using Threading Module To implement a new thread using the threading module, you have to do the following

Define a new subclass of the Thread class. Override the init(self [,args]) method to add additional arguments. Then, override the run(self [,args]) method to implement what the thread should do when started. Once you have created the new Thread subclass, you can create an instance of it and then start a new thread by invoking the start(), which in turn calls run() method.

Example

!/usr/bin/python

import threading import time

exitFlag = 0

class myThread (threading.Thread): def $_{init}{}_{(self,threadID,name,counter):threading.Thread.}{}_{init}{}_{(self)self.threadID=threadI}$

def print$_time(threadName, delay, counter) : whilecounter : ifexitFlag : threadName.exit()time.sleep(delay)print" counter- = 1$

Create new threads thread1 = myThread(1, "Thread-1", 1) thread2 = myThread(2, "Thread-2", 2)

Start new Threads thread1.start() thread2.start()

print "Exiting Main Thread" When the above code is executed, it produces the following result

Starting Thread-1 Starting Thread-2 Exiting Main Thread Thread-1: Thu Mar 21 09:10:03 2013 Thread-1: Thu Mar 21 09:10:04 2013 Thread-2: Thu Mar 21 09:10:04 2013 Thread-1: Thu Mar 21 09:10:05 2013 Thread-1: Thu Mar 21 09:10:06 2013 Thread-2: Thu Mar 21 09:10:06 2013 Thread-1: Thu Mar 21 09:10:07 2013 Exiting Thread-1 Thread-2: Thu Mar 21 09:10:08 2013 Thread-2: Thu Mar 21 09:10:10 2013 Thread-2: Thu Mar 21 09:10:12 2013 Exiting Thread-2 Synchronizing Threads The threading module provided with Python includes a simple-to-implement locking mechanism that allows you to synchronize threads. A new lock is created by calling the Lock() method, which returns the new lock.

The acquire(blocking) method of the new lock object is used to force threads to run synchronously. The optional blocking parameter enables you to control whether the thread waits to acquire the lock.

If blocking is set to 0, the thread returns immediately with a 0 value if the lock cannot be acquired and with a 1 if the lock was acquired. If blocking is set to 1, the thread blocks and wait for the lock to be released.

The release() method of the new lock object is used to release the lock when it is no longer required.

Example

!/usr/bin/python

import threading import time

class myThread (threading.Thread): def ${}_{i}nit_{(self,threadID,name,counter):threading.Thread.{}_{i}nit_{(self)self.threadID=threadI}}$

def print$_{t}ime(threadName, delay, counter) : whilecounter : time.sleep(delay)print''counter- =$ 1

threadLock = threading.Lock() threads = []

Create new threads thread1 = myThread(1, "Thread-1", 1) thread2 = myThread(2, "Thread-2", 2)

Start new Threads thread1.start() thread2.start()

Add threads to thread list threads.append(thread1) threads.append(thread2)

Wait for all threads to complete for t in threads: t.join() print "Exiting Main Thread" When the above code is executed, it produces the following result

Starting Thread-1 Starting Thread-2 Starting Thread-3 Thread-1 processing One Thread-2 processing Two Thread-3 processing Three Thread-1 processing Four Thread-2 processing Five Exiting Thread-3 Exiting Thread-1 Exiting Thread-2 Exiting Main Thread Related Readings "Python Multithreaded Programming". www.tutorialspoint.com. N.p., 2016. Web. 13 Dec. 2016. "An Introduction To Python Concurrency". dabeaz.com. N.p., 2016. Web. 14 Dec. 2016.

3.12 Event Based Programming

Introduction: pydispatcher 1 2 PyDispatcher provides the Python programmer with a multiple-producer-multiple-consumer signal-registration and routing infrastructure for use in multiple contexts. The mechanism of PyDispatcher started life as a highly rated recipe in the Python Cookbook. The project aims to include various enhancements to the recipe developed during use in various applications. It is primarily maintained by Mike Fletcher. A derivative of the project provides the Django web framework's "signal" system.

Used by Django community

Usage 1 To set up a function to receive signals: from pydispatch import dispatcher

SIGNAL = 'my-first-signal'

def handle$_{e}vent(sender) : """Simpleeventhandler"""print'Signalwassentby', sender$

dispatcher.connect(handle$_{e}vent, signal = SIGNAL, sender = dispatcher.Any)$

The use of the Any object allows the handler to listen for messages from any Sender or to listen to Any message being sent. To send messages: first$_{s}ender =$

$object()second_sender =$

def main(): dispatcher.send(signal=SIGNAL, sender=first$_sender)dispatcher.send(signal = SIGNAL, sender = second_sender)$

Which causes the following to be printed:

Signal was sent by <object object at 0x196a090> Signal was sent by Messaging Conda link Docker link Github - pubSubService Github - pubSubClient Pypi link

Python Publish - Subscribe Pattern Implementation:

Step by Step to run PubSub: Step 1: Pull pubsub image from docker hub run it: docker pull hunguyen/pubsub:latest docker run -d -p 8000:8000 hunguyen/pubsub Step 2: To run client first install pyconfiguration from conda conda install -c rain1024 pyconfiguration Step 3: Install pubSubClient package from conda conda install -c hunguyen pubsubclient Step 4: Create config.json file "PUBLISH$_SUBSCRIBE_SERVICE" : "http : //api.service.com" Step5 : Runpubsubclientcreateandregisterorsyncapublisherpublisher = Publisher('P1')createanewtopictopic = Topic('A')createaneventofatopicevent = Event(topic)publisherpublishesaneventpublisher.publish(event)c$
$Subscriber('S1')subscribersubscribestoatopicsubscriber.subscribe(topic)subscribergetallneweventsbytimest$
$subscriber.get_events()pydispatcher$

stackoverflow, Recommended Python publish/subscribe/dispatch module?

3.13 Web Development

Django 1 Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It's free and open source.

Project Folder Structure

project$_folder/your_project_name/your_project_name/static/models.pyserializers.pysettings.pyurls.pyvie$
$InstalldependenciespipinstalldjangopipinstalldjangorestframeworkpipinstallmarkdownMarkdownsuppo$
$filterFilteringsupportpipinstalldjango-cors-headersCORSsupportStep2 :$
$Createprojectdjango-adminstartprojectyour_project_nameStep3 : Configapps3Add'your_project_name',' res$

INSTALLED$_APPS = (...'your_project_name''rest_framework',)Step4 : Model, View, Route6Step4.1 : CreatemodelandserializerYoucangotoDjango : Modelfieldreferencepageformorefields.$

Step 4.1.1: Create Task class in your$_project_name/models.pyfilefromdjango.dbimportmodels$

class Task(models.Model): content = models.CharField(max$_length = 30)status = models.CharField(max_length = 30)Step4.1.2 : CreateTaskSerializerclassinyour_project_name/serializers$

class TaskSerializer(serializers.HyperlinkedModelSerializer): class Meta: model = Task fields = ('id', 'content', 'status') Step 4.1.3: Create table in database 4 python manage.py syncdb With django 1.9

python manage.py makemigrations your$_project_namepythonmanage.pymigrateStep4.2 : CreateTaskViewSetclassinyour_project_name/views.pyfilefromyour_project_name.modelsimportTaskfrom$

class TaskViewSet(viewsets.ModelViewSet): queryset = Task.objects.all() serializer$_class = TaskSerializerStep4.3 : Configroute5Changeyour_project_name/urls.pyfile$

from django.conf.urls import include, url from django.contrib import admin from rest$_frameworkimportroutersfromyour_project_name.viewsimportTaskViewSet$

router = routers.DefaultRouter() router.register(r'api/tasks', TaskViewSet) admin.autodiscover()

urlpatterns = [url(r'$^admin/', include(admin.site.urls)), url(r'', include(router.urls)), url(r'^api-auth/', include('rest_framework.urls', namespace=' rest_framework'))]Step5 :$

$RunServerpythonmanage.pyrunserverStep6.UseAPIStep6.1 : Createanewtaskcurl-i - XPOST - H"Content - Type : application/json"http : //localhost : 8000/api/tasks-d'"content" : "a", "status" : "INIT"'Step6.2 : Listalltaskscurlhttp : //localhost : 8000/api/tasksStep6.3 : Getdetailoftask1curlhttp : //localhost : 8000/api/tasks/1Step6.4 : Deletetask1curl-i-XDELETEhttp : //localhost : 8000/api/tasks/1Step7 : CORSKnownError : No'Access-Control-Allow-Origin'headerispresentontherequestedresource.Origin'null'isthereforenotallowedaccess.$

Step 7.1: Install corsheader app Add module corsheaders to your$_project_name/settings.py$

INSTALLED$_APPS = (...'corsheaders', ...)Step7.2AddmiddlewareclassesAddmiddleware_classestoyour_p$

MIDDLEWARE$_CLASSES = (...'corsheaders.middleware.CorsMiddleware', 'django.middleware.comm$
$AllowAll$

Add this line to your$_project_name/settings.py$

CORS$_ORIGIN_ALLOW_ALL : TrueStep8 : httpsYoucanusehttps : //github.com/teddziuba/django-sslserver$

Unicode REST$_FRAMEWORK = 'DEFAULT_RENDERER_CLASSES' : ('rest_framework.renderers.$
$PagingAddthismodulesettingtoyour_project_name/settings.py$

REST$_FRAMEWORK = 'DEFAULT_PAGINATION_CLASS' :' rest_framework.pagination.LimitOf$

API:

GET <>/?limit=<limit>offset=<offset>

Step 10: Search by field in import this to your viewsets.py

from rest$_frameworkimportfilters$

add this to your viewsets class

filter$_backends = (filters.SearchFilter,)search_fields = ('< field >', '< field >',)$

One-to-Many Relationship 7 from django.db import models

class User(models.Model): name = models.TextField()

def $_{str_{(self):return"-".format(str(self.id),self.name)}}$

class Task(models.Model): name = models.TextField() assign = models.ForeignKey(User, on$_delete = models.CASCADE)StartingwithMysqlAddthisdatabasesettingstoyour_project_name/settings.p$

DATABASES = 'default': 'ENGINE': 'django.db.backends.mysql', 'NAME': '[DB$_NAME]', 'USER' :' [DB_USER]', 'PASSWORD' :' [PASSWORD]', 'HOST' :' [HOST]', OranIPAddressthatyourDBishostedon'PORT' :'$ 3306',

Install this module to your virtual environment

conda install mysql-python if you are using virtual environment

pip install mysql-python if you using are root environment

Custom View 8 from rest$_frameworkimportmixins$

class CreateModelMixin(object): """ Create a model instance. """ def create(self, request, *args, **kwargs): event = request.data try: event['time'] = int(time.time()) except Exception, e: print 'Set Time Error' serializer = self.get$_serializer(data = request.data)serializer.is_valid(raise_exception = True)self.perform_create(serializer)headers = self.get_success_headers(serializer.data)returnResponse(serializer.data, status = status.HTTP_201_CREATED, headers = headers)$

def perform$_create(self, serializer) : serializer.save()$

def get$_success_headers(self, data) : try : return'Location' : data[api_settings.URL_FIELD_NAME]except($
$return$

class YourViewSet(CreateModelMixin, mixins.RetrieveModelMixin, mixins.UpdateModelMixin, mixins.DestroyModelMixin, mixins.ListModelMixin, GenericViewSet): queryset = YourModel.objects.all() serializer$_class = YourModelSerializerLoggingsettingsHereisanexample, putthi$

LOGGING = 'version': 1, 'disable$_e$$xisting_loggers' : False,' formatters' :$
$'verbose' : 'format' :',' simple' : 'format' :',,' filters' : 'special' : '()' :' project.logging.SpecialFilter',' foo'$
$'console' : 'level' :' INFO',' filters' : ['require_debug_true'],' class' :' logging.StreamHandler',' formatter' :'$
$'django' : 'handlers' : ['console'],' propagate' : True,,' django.request' : 'handlers' : ['mail_admins'],' level' :'$

Python: Build Python API Client package Step 1: Write document on Swagger Editor1 Step 2: Genenrate Client –> Python –> save python-client.zip Step 3: Extract zip Step 4: Open project in Pycharm rename project directory, project name, swagger$_c$$lientpackageStep5 : 2mkdircondacdcondagitclonehttps : //github.com/hunguyen1702/conda$ $rf.gitREADME.mdStep6 : Editmeta.yamlfileinyour_package folder6.1Followinstructioninsidemeta.yaml$ $build : -python - setuptoolsrun : -pythonwith : requirements : build : -python - setuptools - six - certifi - python - dateutilrun : -python - six - certifi - python - dateutilStep7 : cd..condabuildyour_packageStep8 : mkdirchannelcdchannelcondaconvert--platformall /anaconda/conda-bld/linux-$ $64/your_package_0.1.0-py27_0.tar.bz2Step9 : Createvirtual-envname : your_env_namedependencies :$ $-certifi = 2016.2.28 = py27_0 - openssl = 1.0.2h = 0 - pip = 8.1.2 = py27_0 - python = 2.7.11 = 0 - python - dateutil = 2.5.3 = py27_0 - readline = 6.2 = 2 - setuptools = 20.7.0 = py27_0 - six = 1.10.0 = py27_0 - tk = 8.5.18 = 0 - wheel = 0.29.0 = py27_0 - zlib = 1.2.8 = 0 - pip : -urllib3 == 1.15.1Step10 :$ $Install : condainstall - -use - localyour_packageDjango$

Writing your first Django app, part 1

Django REST framework: Installation

Django: Migrations

Building a Simple REST API for Mobile Applications

Django: Models

How to show object details in Django Rest Framework browseable API?

rest$_f$$ramework : mixins$

3.14 Logging

logging 1 2 3 levels, attributes references

The logging library takes a modular approach and offers several categories of components: loggers, handlers, filters, and formatters.

Loggers expose the interface that application code directly uses. Handlers send the log records (created by loggers) to the appropriate destination. Filters provide a finer grained facility for determining which log records to output. Formatters specify the layout of log records in the final output. Step 0: Project structure

code/ main.py config logging.conf logs app.log Step 1: Create file logging.conf

[loggers] keys=root

[handlers] keys=consoleHandler,fileHandler

[formatters] keys=formatter

[logger$_r$$oot]level = DEBUGhandlers = consoleHandler, fileHandler$

[handler$_c$$onsoleHandler]class = StreamHandlerlevel = DEBUGformatter = formatterargs = (sys.stdout,)$

[handler$_f$$ileHandler]class = FileHandlerlevel = DEBUGformatter = formatterargs = ('logs/app.log',' a')$

[formatter$_f$$ormatter]format = datefmt = Step2 : Loadconfigandcreatelogger$

In main.py

import logging.config

load logging config logging.config.fileConfig('config/logging.conf') Step 3: In your application code

logging.getLogger().debug('debug message') logging.getLogger().info('info message') logging.getLogger().warn('warn message') logging.getLogger().error('error message') logging.getLogger().critical('critical message') More Resources

Introduction to Logging Quick and simple usage of python log Python: Logging module

Python: Logging cookbook

Python: Logging guide

3.15 Configuration

pyconfiguration

Installation conda install -c rain1024 pyconfiguration Usage Step 1: Create config.json file

"SERVICE$_U RL$" : "$http : //api.service.com$" $Step2 : Addthesecodetomain.pyfile$

from pyconfiguration import Configuration Configuration.load('config.json') print Configuration.SERVICE$_U RL$

$>$ http://api.service.com References: What's the best practice using a settings file 1

What's the best practice using a settings file in Python?

3.16 Command Line

Command Line Arguments There are the following modules in the standard library:

The getopt module is similar to GNU getopt. The optparse module offers object-oriented command line option parsing. Here is an example that uses the latter from the docs:

from optparse import OptionParser

parser = OptionParser() parser.add$_o ption("-f", "--file", dest = "filename", help = "writereporttoFILE", metavar = "FILE")parser.add_o ption("-q", "--quiet", action = "store_f alse", dest = "verbose", default = True, help = "don'tprintstatusmessagestostdout")$

(options, args) = parser.parse$_a rgs()optparsesupports(amongotherthings) :$

Multiple options in any order. Short and long options. Default values. Generation of a usage help message. Suggest Reading Command Line Arguments In Python

3.17 Testing

Testing your code is very important.

Getting used to writing testing code and running this code in parallel is now considered a good habit. Used wisely, this method helps you define more precisely your code's intent and have a more decoupled architecture.

Unittest unittest is the batteries-included test module in the Python standard library. Its API will be familiar to anyone who has used any of the JUnit/-nUnit/CppUnit series of tools.

The Basics Creating test cases is accomplished by subclassing unittest.TestCase.

import unittest

def fun(x): return x + 1

class MyTest(unittest.TestCase): def test(self): self.assertEqual(fun(3), 4) Skipping tests Unittest supports skipping individual test methods and even whole classes of tests. In addition, it supports marking a test as an "expected failure," a test that is broken and will fail, but shouldn't be counted as a failure on a .code TestResult.

Skipping a test is simply a matter of using the skip() decorator or one of its conditional variants.

import sys import unittest

class MyTestCase(unittest.TestCase):

@unittest.skip("demonstrating skipping") def test$_nothing(self): self.fail("shouldn't happen")$

@unittest.skipIf(mylib.$_{version_{<(1,3),"not supported in this library version") def test_format(self): Tests that work for only a certain versi}}$

@unittest.skipUnless(sys.platform.startswith("win"), "requires Windows") def test$_windows_support(self): windows specific testing code pass Tox tox aims to automate and standardize testing$

Tox is a generic virtualenv management and test command line tool you can use for:

checking your package installs correctly with different Python versions and interpreters running your tests in each of the environments, configuring your test tool of choice acting as a frontend to Continuous Integration servers, greatly reducing boilerplate and merging CI and shell-based testing. Installation

You can install tox with pip using the following command

$pip install tox Setup default environment in Windows with conda$

$conda create -p C: 27 python = 2.7$ conda create -p C:34 python=3.4 Related Readings Testing Your Code, The Hitchhiker's Guide to Python unittest — Unit testing framework, docs.python.org Is it possible to use tox with conda-based Python installations?, stackoverflow

3.18 IDE Debugging

Today, I write some notes about my favorite Python IDE - PyCharm. I believe it's a good one for developing python, which supports git, vim, etc. This list below contains my favorite features.

Pycharm Features Intelligent Editor Navigation Graphical Debugger Refactorings Code Inspections Version Control Integration Scientific Tools Intelligent Editor PyCharm provides smart code completion, code inspections, on-the-fly error highlighting and quick-fixes, along with automated code refactorings and rich navigation capabilities.

Syntax Highlighting

Read your code easier with customizable colors for Python code and Django templates. Choose from several predefined color themes.

Auto-Identation and code formating

Automatic indents are inserted on new line. Indent verification and code re-formatting are compliant with project code-style settings.

Configurable code styles

Select a predefined coding style to apply to your code style configuration for various supported languages.

Code completion

Code completion for keywords, classes, variables, etc. as you type or via Ctrl+Space. Editor suggestions are context-aware and offer the most appropriate options.

Keyboard shortcuts: Tab, Alt+Enter

Code selection and comments

Select a block of code and expand it to an expression, to a line, to a logical block of code, and so on with shortcuts. Single keystroke to comment/uncomment the current line or selection.

Code formatter

Code formatter with code style configuration and other features help you write neat code that's easy to support. PyCharm contains built-in PEP-8 for Python and other standards compliant code formatting for supported languages.

Code snippets and templates

Save time using advanced customizable and parametrized live code templates and snippets.

Keyboard shortcuts check.if ENTER

if check: type$_{s}omethingCodefolding$

Code folding, auto-insertion of braces, brackets quotes, matching brace/bracket highlighting, etc.

On-the-fly error highlighting

Errors are shown as you type. The integrated spell-checker verifies your identifiers and comments for misspellings.

Multiple carets and selections

With multiple carets, you can edit several locations in your file at the same time.

Keyboard shortcuts: SHIFT + F6

Code analysis

Numerous code inspections verify Python code as you type and also allow inspecting the whole project for possible errors or code smells.

Quick-fixes

Quick-fixes for most inspections make it easy to fix or improve the code instantly. Alt+Enter shows appropriate options for each inspection.

Keyboard shortcuts: F2

Duplicated code detector

Smart duplicated code detector analyzes your code and searches for copy/-pasted code. You'll be presented with a list of candidates for refactoring—and with the help of refactorings it's easy to keep your code dry.

Configurable language injections

Natively edit non-Python code embedded into string literals, with code completion, error-highlighting, and other coding assistance features.

Code auto generation

Code auto-generation from usage with quick-fixes; docstrings and the code matching verification, plus autoupdate on refactoring. Automatic generation of a docstring stub (reStructuredText, Epytext, Google, and NumPy).

Intention actions

Intention actions help you apply automated changes to code that is correct, to improve it or to make your coding routine easier.

Searching

Keyboard shortcuts: Double Shift (search everywhere)

Navigation Shortcuts

Keyboard shortcuts: ALT + SHIFT + UP/DOWN (move line up and down)

Graphical Debugger PyCharm provides extensive options for debugging your Python/Django and JavaScript code:

Set breakpoints right inside the editor and define hit conditions Inspect context-relevant local variables and user-defined watches, including arrays and complex objects, and edit values on the fly Set up remote debugging using remote interpreters Evaluate an expression in runtime and collect run-time type statistics for better autocompletion and code inspections Attach to a running process Debug Django templates

Inline Debugger

With an inline debugger, all live debugging data are shown directly in the editor, with variable values integrated into the editor's look-and-feel. Variable values can be viewed in the source code, right next to their usages.

Step into My Code

Use Step into My Code to stay focused on your code: the debugger will only step through your code bypassing any library sources.

Multi-process debugging

PyCharm can debug applications that spawn multiple Python processes, such as Django applications that don't run in –no-reload mode, or applications using many other Web frameworks that use a similar approach to code auto-reloading.

Run/Debug configurations

Every script/test or debugger execution creates a special 'Run/Debug Configuration' that can be edited and used later. Run/Debug Configurations can be shared with project settings for use by the whole team.

Workspace Custom Scheme Go to File - Settings... then Editor - Colors Fonts

Now you can change your scheme, I like Darcular

https://confluence.jetbrains.com/download/attachments/51945983/appearance3.png?version=1modificat

IPython Support PyCharm supports usage of IPython magic commands.

http://i.stack.imgur.com/aTEW2.png

Vim Support You can configure PyCharm to work as a Vim editor

https://confluence.jetbrains.com/download/attachments/51946537/vim4.png?version=1modificationDat

Keyboard Shortcuts: Ctrl+Shift+V (paste)

3.19 Package Manager

py2exe py2exe is a Python Distutils extension which converts Python scripts into executable Windows programs, able to run without requiring a Python installation.Spice

Installation py2exe conda install -c https://conda.anaconda.org/clinicalgraphics cg-py2exe Build 1 python setup.py py2exe build PyQT python setup.py py2exe –includes sip Known Issues Error: Microsoft Visual C++ 10.0 is required (Unable to find vcvarsall.bat) (link)

How to fix

Step 1: Install Visual Studio 2015

Step 2:

set VS100COMNTOOLS=

3.20 Environment

Environment Management Similar to pip, conda is an open source package and environment management system 1. Anaconda is a data science platform that comes with a lot of packages. It uses conda at the core. Unlike Anaconda, Miniconda doesn't come with any installed packages by default. Note that for miniconda, everytime you open up a terminal, conda won't automatically be available. Run the command below to use conda within miniconda.

Conda Let's first start by checking if conda is installed.

```
$ conda --version

conda 4.2.12
To see the full documentation for any command, type the command
    ↪ followed by --help. For example, to learn about the conda
    ↪ update command:

$ conda update --help
Once it has been confirmed that conda has been installed, we will
    ↪  now make sure that it is up to date.

$ conda update conda

Using Anaconda Cloud api site https://api.anaconda.org
Fetching package metadata: ....
.Solving package specifications: .........

Package plan for installation in environment //anaconda:

The following packages will be downloaded:

    package | build
    ---------------------------|-----------------
    conda-env-2.6.0 | 0 601 B
    ruamel_yaml-0.11.14 | py27_0 184 KB
    conda-4.2.12 | py27_0 376 KB
    ------------------------------------------------------------
                                           Total: 560 KB

The following NEW packages will be INSTALLED:

    ruamel_yaml: 0.11.14-py27_0

The following packages will be UPDATED:

    conda: 4.0.7-py27_0 --> 4.2.12-py27_0
    conda-env: 2.4.5-py27_0 --> 2.6.0-0
    python: 2.7.11-0 --> 2.7.12-1
    sqlite: 3.9.2-0 --> 3.13.0-0
```

```
Proceed ([y]/n)? y

Fetching packages ...
conda-env-2.6. 100% |################################| Time:
    ↪ 0:00:00 360.78 kB/s
ruamel_yaml-0. 100% |################################| Time:
    ↪ 0:00:00 5.53 MB/s
conda-4.2.12-p 100% |################################| Time:
    ↪ 0:00:00 5.84 MB/s
Extracting packages ...
[ COMPLETE ]|###################################################|
    ↪  100%
Unlinking packages ...
[ COMPLETE ]|###################################################|
    ↪  100%
Linking packages ...
[ COMPLETE ]|###################################################|
    ↪  100%
Environments
Create
In order to manage environments, we need to create at least two
    ↪ so you can move or switch between them. To create a new
    ↪ environment, use the conda create command, followed by any
    ↪  name you wish to call it:

# create new environment
conda create -n <your_environment> python=2.7.11
Clone
Make an exact copy of an environment by creating a clone of it.
    ↪ Here we will clone snowflakes to create an exact copy
    ↪ named flowers:

conda create --name flowers --clone snowflakes
List
List all environments

Now you can use conda to see which environments you have
    ↪ installed so far. Use the conda environment info command
    ↪ to find out

$ conda info -e

conda environments:
snowflakes /home/username/miniconda/envs/snowflakes
bunnies /home/username/miniconda/envs/bunnies
Verify current environment

Which environment are you using right now  snowflakes or bunnies?
    ↪  To find out, type the command:
```

```
conda info --envs
Remove
If you didnt really want an environment named flowers, just remove it as follows:

conda remove --name flowers --all
Share
You may want to share your environment with another person, for example, so they can re-create a test that you have done. To allow them to quickly reproduce your environment, with all of its packages and versions, you can give them a copy  of your environment.yml file.

Export the environment file

To enable another person to create an exact copy of your environment, you will export the active environment file.

conda env export > environment.yml
Use environment from file

Create a copy of another developers environment from their environment.yml file:

conda env create -f environment.yml
# remove environment
conda remove -n <your_environemnt> --all
```

3.21 Module

Create Public Module conda, pypi, github

Step 0/4: Check your package name Go to https://pypi.python.org/pypi/your$_package_nametoseeyourpack$

Step 1/4: Make your module 1 1.1 pip install cookiecutter

1.2 cookiecutter https://github.com/audreyr/cookiecutter-pypackage.git

1.3 Fill all necessary information

full$_name[AudreyRoyGreenfeld] : email[aroy@alum.mit.edu] : github_username[audreyr] : project_name[PythonBoilerplate] : project_slug[] : project_short_description : release_date[] : pypi_username[] : year[2016] : version[0.1.0] : use_pypi_deployment_with_travis[y] : Itwillcreateadirectory$

|- LICENSE |- README.md |- TODO.md |- docs | |– conf.py | |– generated | |– index.rst | |– installation.rst | |– modules.rst | |– quickstart.rst | |– sandman.rst |- requirements.txt |- your$_package||--_{init.py||--your_package.py||--test||--models.py||--test_your_package.py|-setup.pyStep2/4}$

2. Create a .pypirc configuration file in $HOMEdirectory$

[distutils] index-servers = pypi

[pypi] repository=https://pypi.python.org/pypi username=your$_usernamepassword = your_password3.ChangeyourMANIFEST.in$

recursive-include project$_folder * 4.UploadyourpackagetoPyPI$

python setup.py register -r pypi python setup.py sdist upload -r pypi Step 4/4: Conda 2 1. Install conda tools conda install conda-build conda install anaconda-client 2. Build a simple package with conda skeleton pypi

cd your$_{p}ackage_{f}oldermkdircondacdcondacondaskeletonpypiyour_{p}ackageThiscreatesadirectorynamedyo$

|- your$_{p}ackage||--bld.bat||--meta.yaml||--build.sh3.Buildyourpackage$

conda build your$_{p}ackage$

convert to all platform conda convert -f –platform all C:-bld-64$_{p}ackage-$ $0.1.1-py27_{0}.tar.bz24.UploadpackagestoAnaconda$

anaconda login anaconda upload linux-32/your$_{p}ackage.tar.bz2anacondauploadlinux-$ $64/your_{p}ackage.tar.bz2anacondauploadwin-32/your_{p}ackage.tar.bz2anacondauploadwin-$ $64/your_{p}ackage.tar.bz2CreatePrivateModuleStep1:Makeyourmodule11.1pipinstallcookiecutter$

1.2 cookiecutter https://github.com/audreyr/cookiecutter-pypackage.git

1.3 Fill all necessary information

full$_{n}ame[AudreyRoyGreenfeld]:email[aroy@alum.mit.edu]:github_{u}sername[audreyr]:$ $project_{n}ame[PythonBoilerplate]:project_{s}lug[]:project_{s}hort_{d}escription:$ $release_{d}ate[]:pypi_{u}sername[]:year[2016]:version[0.1.0]:use_{p}ypi_{d}eployment_{w}ith_{t}ravis[y]:$ $Step2:BuildyourmoduleChangeyourMANIFEST.in$

recursive-include project$_{f}older*Buildyourmodulewithsetup.py$

cd your$_{p}roject_{f}older$

build local python setup.py build > It will create a new folder in $>PYTHON_{H}OME/Lib/sites-$ $packages/your_{p}roject_{n}ame-0.1.0-py2.7.egg$

build distribution python setup.py sdist > It will create a zip file in $PROJECT_{F}OLDER/distStep3:$ $UsageyourmoduleInthesamemachine$

import your$_{p}roject_{n}ameInothermachine$

Python: Build Install Local Package with Conda Here is a step by step tutorial about building a local module package install it from a custom channel 1

Step 1: Make a setup folder for your package with cookkiecutter on terminal:

mkdir build cd build pip install cookiecutter cookiecutter https://github.com/audreyr/cookiecutter-pypackage.git

Fill all necessary information

full$_{n}ame[AudreyRoyGreenfeld]:email[aroy@alum.mit.edu]:github_{u}sername[audreyr]:$ $project_{n}ame[PythonBoilerplate]:project_{s}lug[]:project_{s}hort_{d}escription:$ $release_{d}ate[]:pypi_{u}sername[]:year[2016]:version[0.1.0]:use_{p}ypi_{d}eployment_{w}ith_{t}ravis[y]:$ $Itwillcreateadirectory$

|- LICENSE |- README.md |- TODO.md |- docs | |– conf.py | |– generated | |– index.rst | |– installation.rst | |– modules.rst | |– quickstart.rst | |– sandman.rst |- requirements.txt |- your$_{p}ackage||--_{init.py||--your_{p}ackage.py||--test||--models.py||--test_{y}our_{p}ackage.py|-setup.pyCopyyou}$

Add this line to MANIFEST.in

recursive-include project$_{f}older*Step2:Buildcondapackagemkdircondacdcondamkdirchannelgitclonehtt$ $//github.com/hunguyen1702/condaBuildLocalTemplate.gitmvcondaBuildLocalTemplateyour_{p}ackage_{n}am$ $rf.gitREADME.mdEditthefilemeta.yamlwiththeinstructioninsideitcd..condabuildyour_{p}ackage_{n}ameStep$ $Createcustomchannelandinstallfromlocalpackage Createachanneldirectory$

cd channel Convert your$_{p}ackageyou'vebuilttoallplatform$

conda convert –platform all /anaconda/conda-bld/linux-64/your$_{p}ackage_{0}.1.0-$ $py27_{0}.tar.bz2andthiswillcreate:$

channel/ linux-64/ package-1.0-0.tar.bz2 linux-32/ package-1.0-0.tar.bz2 osx-64/ package-1.0-0.tar.bz2 win-64/ package-1.0-0.tar.bz2 win-32/ package-1.0-0.tar.bz2 Register your package to your new channel

cd .. conda index channel/linux-64 channel/osx-64 channel/win-64 Veriy your new channel conda search -c file://path/to/channel/ –override-channels If you see your$_{p}ackage'sappearance, soit'swork$

After that if you want to install that package from local, run this command:

conda install –use-local your$_{p}ackage$

and when you want to create environment with local package from file, you just have export environment to .yml file and add this channels section before the dependencies section:

channels: - file://path/to/your/channel/

3.22 Production

Production with docker Base Image: magizbox/conda2.7/

Docker Folder

your$_{a}pp/appconfigmain.pyDockerfilerun.shDockerfile$

FROM magizbox/conda2.7:4.0

ADD ./app /app ADD ./run.sh /run.sh

RUN conda env create -f environment.yml run.sh

source activate your$_{e}nvironment$

cd /app

python main.py Compose

service: build: ./service-app command: 'bash run.sh' Note: an other python conda with lower version (such as 3.5), will occur error when install requests package

3.23 Quản lý gói với Anaconda

Cài đặt package tại một branch của một project trên github

```
$ pip install git+https://github.com/tangentlabs/django-oscar-
    ↪ paypal.git@issue/34/oscar-0.6#egg=django-oscar-paypal
```

Trích xuất danh sách package

```
$ pip freeze > requirements.txt
```

Chạy ipython trong environment anaconda
Chạy đống lệnh này

```
conda install nb_conda
source activate my_env
python -m IPython kernelspec install-self --user
ipython notebook
```

Interactive programming với ipython
Trích xuất ipython ra slide (không hiểu sao default '–to slides' không work nữa, lại phải thêm tham số '–reveal-prefix' [[1]]

```
jupyter nbconvert "file.ipynb"
  --to slides
  --reveal-prefix "https://cdnjs.cloudflare.com/ajax/libs/reveal.
      ↪ js/3.1.0"
```

Tham khảo thêm

* https://stackoverflow.com/questions/37085665/in-which-conda-environment-is-jupyter-executing * https://github.com/jupyter/notebook/issues/541issuecomment-146387578 * https://stackoverflow.com/a/20101940/772391

python 3.4 hay 3.5

Có lẽ 3.5 là lựa chọn tốt hơn (phải có của tensorflow, pytorch, hỗ trợ mock)

Quản lý môi trường phát triển với conda

Chạy lệnh 'remove' để xóa một môi trường

```
conda remove --name flowers --all
```

3.24 Test với python

Sử dụng những loại test nào?

Hiện tại mình đang viết unittest với default class của python là Unittest. Thực ra toàn sử dụng 'assertEqual' là chính!

Ngoài ra mình cũng đang sử dụng tox để chạy test trên nhiều phiên bản python (python 2.7, 3.5). Điều hay của tox là mình có thể thiết kế toàn bộ cài đặt project và các dependencies package trong file 'tox.ini'

Chạy test trên nhiều phiên bản python với tox

Pycharm hỗ trợ debug tox (quá tuyệt!), chỉ với thao tác đơn giản là nhấn chuột phải vào file tox.ini của project.

3.25 Xây dựng docs với readthedocs và sphinx

20/12/2017: Tự nhiên hôm nay tất cả các class có khai báo kế thừa ở project languageflow không thể index được. Vãi thật. Làm thằng đệ không biết đâu mà build model.

Thử build lại chục lần, thay đổi file conf.py và package_reference.rst chán chê không được. Giả thiết đầu tiên là do hai nguyên nhân (1) docstring ghi sai, (2) nội dung trong package_reference.rst bị sai. Sửa chán chê cũng vẫn thế, thử checkout các commit của git. Không hoạt động!

Mất khoảng vài tiếng mới để ý thằng readthedocs có phần log cho từng build một. Lần mò vào build gần nhất và build (mình nhớ là) thành công cách đây 2 ngày

Log build gần nhất

```
Running Sphinx v1.6.5
making output directory...
loading translations [en]... done
loading intersphinx inventory from https://docs.python.org/
    ↪ objects.inv...
intersphinx inventory has moved: https://docs.python.org/objects.
    ↪ inv -> https://docs.python.org/2/objects.inv
loading intersphinx inventory from http://docs.scipy.org/doc/
    ↪ numpy/objects.inv...
intersphinx inventory has moved: http://docs.scipy.org/doc/numpy/
    ↪ objects.inv -> https://docs.scipy.org/doc/numpy/objects.
    ↪ inv
```

```
building [mo]: targets for 0 po files that are out of date
building [readthedocsdirhtml]: targets for 8 source files that
    ↪ are out of date
updating environment: 8 added, 0 changed, 0 removed
reading sources... [ 12%] authors
reading sources... [ 25%] contributing
reading sources... [ 37%] history
reading sources... [ 50%] index
reading sources... [ 62%] installation
reading sources... [ 75%] package_reference
reading sources... [ 87%] readme
reading sources... [100%] usage

looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [ 12%] authors
writing output... [ 25%] contributing
writing output... [ 37%] history
writing output... [ 50%] index
writing output... [ 62%] installation
writing output... [ 75%] package_reference
writing output... [ 87%] readme
writing output... [100%] usage
```

Log build hồi trước

```
Running Sphinx v1.5.6
making output directory...
loading translations [en]... done
loading intersphinx inventory from https://docs.python.org/
    ↪ objects.inv...
intersphinx inventory has moved: https://docs.python.org/objects.
    ↪ inv -> https://docs.python.org/2/objects.inv
loading intersphinx inventory from http://docs.scipy.org/doc/
    ↪ numpy/objects.inv...
intersphinx inventory has moved: http://docs.scipy.org/doc/numpy/
    ↪ objects.inv -> https://docs.scipy.org/doc/numpy/objects.
    ↪ inv
building [mo]: targets for 0 po files that are out of date
building [readthedocs]: targets for 8 source files that are out
    ↪ of date
updating environment: 8 added, 0 changed, 0 removed
reading sources... [ 12%] authors
reading sources... [ 25%] contributing
reading sources... [ 37%] history
reading sources... [ 50%] index
reading sources... [ 62%] installation
reading sources... [ 75%] package_reference
reading sources... [ 87%] readme
```

```
reading sources... [100%] usage

/home/docs/checkouts/readthedocs.org/user_builds/languageflow/
    ↪ checkouts/develop/languageflow/transformer/count.py:
    ↪ docstring of languageflow.transformer.count.
    ↪ CountVectorizer:106: WARNING: Definition list ends without
    ↪  a blank line; unexpected unindent.
/home/docs/checkouts/readthedocs.org/user_builds/languageflow/
    ↪ checkouts/develop/languageflow/transformer/tfidf.py:
    ↪ docstring of languageflow.transformer.tfidf.
    ↪ TfidfVectorizer:113: WARNING: Definition list ends without
    ↪  a blank line; unexpected unindent.
../README.rst:7: WARNING: nonlocal image URI found: https://img.
    ↪ shields.io/badge/latest-1.1.6-brightgreen.svg
looking for now-outdated files... none found
pickling environment... done
checking consistency... done
preparing documents... done
writing output... [ 12%] authors
writing output... [ 25%] contributing
writing output... [ 37%] history
writing output... [ 50%] index
writing output... [ 62%] installation
writing output... [ 75%] package_reference
writing output... [ 87%] readme
writing output... [100%] usage
```

Đập vào mắt là sự khác biệt giữa documentation type
Lỗi

```
building [readthedocsdirhtml]: targets for 8 source files that
    ↪ are out of date
```

Chạy

```
building [readthedocs]: targets for 8 source files that are out
    ↪ of date
```

Hí ha hí hửng. Chắc trong cơn bất loạn sửa lại settings đây mà. Sửa lại nó trong phần Settings (Admin gt; Settings gt; Documentation type)
`![](https://magizbox.files.wordpress.com/2017/10/screenshot-from-2017-12-20-09-54-23.png)`
Khi chạy nó đã cho ra log đúng

```
building [readthedocsdirhtml]: targets for 8 source files that
    ↪ are out of date
```

Nhưng vẫn lỗi. Vãi!!! Sau khoảng 20 phút tiếp tục bấn loạn, chửi bới readthedocs các kiểu. Thì để ý dòng này
Lỗi

```
Running Sphinx v1.6.5
```

Chạy

```
Running Sphinx v1.5.6
```

Ngay dòng đầu tiên mà không để ý, ngu thật. Aha, Hóa ra là thằng readthedocs nó tự động update phiên bản sphinx lên 1.6.5. Mình là mình chúa ghét thay đổi phiên bản (code đã mệt rồi, lại còn phải tương thích với nhiều phiên bản nữa thì ăn c** à). Đầu tiên search với Pycharm thấy dòng này trong ‘conf.py‘

```
# If your documentation needs a minimal Sphinx version, state it
    ↪ here.
# needs_sphinx = '1.0'
```

Đổi thành

```
# If your documentation needs a minimal Sphinx version, state it
    ↪ here.
needs_sphinx = '1.5.6'
```

Vẫn vậy (holy sh*t). Thử sâu một tẹo (thực sự là rất nhiều tẹo). Thấy cái này trong trang Settings

!\[\](https://magizbox.files.wordpress.com/2017/10/screenshot-from-2017-12-20-10-01-39.png)

Ờ há. Thằng đần này cho phép trỏ đường dẫn tới một file trong project để cấu hình dependency. Haha. Tạo thêm một file ‘requirements‘ trong thư mục ‘docs‘ với nội dung

```
sphinx==1.5.6
```

Sau đó cấu hình nó trên giao diện web của readthedocs

!\[\](https://magizbox.files.wordpress.com/2017/10/screenshot-from-2017-12-20-10-04-49.png)

Build thử. Build thử thôi. Cảm giác đúng lắm rồi đấy. Và... nó chạy. Ahihi

!\[\](https://magizbox.files.wordpress.com/2017/10/screenshot-from-2017-12-20-10-06-32.png)

Kinh nghiệm

* Khi không biết làm gì, hãy làm 3 việc. Đọc LOG. Phân tích LOG. Và cố gắng để LOG thay đổi theo ý mình.

PS: Trong quá trình này, cũng không thèm build thằng PDF với Epub nữa. Tiết kiệm được bao nhiêu thời gian.

3.26 Pycharm Pycharm

01/2018: Pycharm là trình duyệt ưa thích của mình trong suốt 3 năm vừa rồi.

Hôm nay tự nhiên lại gặp lỗi không tự nhận unittest, không resolve được package import bởi relative path. Vụ không tự nhận unittest sửa bằng cách xóa file .idea là xong. Còn vụ không resolve được package import bởi relative path thì vẫn chịu rồi. Nhìn code cứ đỏ lòm khó chịu thật.

3.27 Vì sao lại code python?

01/11/2017 Thích python vì nó quá đơn giản (và quá đẹp).

[1] : $https : //github.com/jupyter/nbconvert/issues/91issuecomment -$ 283736634

Chương 4

C++

C++ is a general-purpose programming language. It has imperative, object-oriented and generic programming features, while also providing facilities for low-level memory manipulation. It was designed with a bias toward system programming and embedded, resource-constrained and large systems, with performance, efficiency and flexibility of use as its design highlights. C++ has also been found useful in many other contexts, with key strengths being software infrastructure and resource-constrained applications, including desktop applications, servers (e.g. e-commerce, web search or SQL servers), and performance-critical applications (e.g. telephone switches or space probes). C++ is a compiled language, with implementations of it available on many platforms and provided by various organizations, including the Free Software Foundation (FSF's GCC), LLVM, Microsoft, Intel and IBM.

View online http://magizbox.com/training/cpp/site/

4.1 Get Started

What do I need to start with CLion? In general to develop in C/C++ with CLion you need:

CMake, 2.8.11+ (Check JetBrains guide for updates) GCC/G++/Clang (Linux) or MinGW 3. or MinGW—w64 3.-4. or Cygwin 1.7.32 (minimum required) up to 2.0. (Windows) Downloading and Installing CMake Downloading and installing CMake is pretty simple, just go to the website, download and install by following the recommended guide there or the on Desktop Wizard.

Download and install file cmake-3.9.0-win64-x65.msi > cmake Usage

cmake [options] <path-to-source> cmake [options] <path-to-existing-build>

Specify a source directory to (re-)generate a build system for it in the current working directory. Specify an existing build directory to re-generate its build system.

Run 'cmake –help' for more information. Downloading and Getting Cygwin Cygwin is a large collection of GNU and Open Source tools which provide functionality similar to a Linux distribution on Windows

Download file setup-x86$_6 4.exefromthewebsitehttps : //cygwin.com/install.html$

Install setup-x86$_6 4.exefile$

This is the root directory where Cygwin will be located, usually the recommended C: works

Choose where to install LOCAL DOWNLOAD PACKAGES: This is not the same as root directory, but rather where packages (ie. extra C libraries and tools) you download using Cygwin will be located

Follow the recommended instructions until you get to packages screen:

Once you get to the packages screen, this is where you customize what libraries or tools you will install. From here on I followed the above guide but here's the gist:

From this window, choose the Cygwin applications to install. For our purposes, you will select certain GNU C/C++ packages.

Click the + sign next to the Devel category to expand it.

You will see a long list of possible packages that can be downloaded. Scroll the list to see more packages.

Pick each of the following packages by clicking its corresponding "Skip" marker.

gcc-core: C compiler subpackage gcc-g++: C++ subpackage libgcc1: C runtime library gdb: The GNU Debugger make: The GNU version of the 'make' utility libmpfr4 : A library for multiple-precision floating-point arithmetic with exact rounding Download and install CLion Download file CLion-2017.2.exe from website https://www.jetbrains.com/clion/download/section=windows

Config environment File > Settings... > Build, Execution, Deployment

Choose Cygwin home: C:64 Choose CMake executable: Bundled CMake 3.8.2 Run your first C++ program with CLion

4.2 Basic Syntax

C/C++ Hello World include <iostream> using namespace std;

int main() cout « "hello world"; Convention Naming variable$_{n}ame_{l}ike_{t}hisclass_{d}ata_{m}emeber_{n}ame_{l}ike_{t}his$
$//GargantuanTableIterator * iter = table- > NewIterator(); //for(iter- >$
$Seek("foo"); !iter- > done(); iter- > Next())//process(iter- > key(), iter- > value()); ////deleteiter; clo$
$Usea"*"hereforconcatenationoperator.//TODO(Zeke)changethistouserelations.$

4.3 Cấu trúc dữ liệu

Data Structure Number C++ offer the programmer a rich assortment of built-in as well as user defined data types. Following table lists down seven basic C++ data types:

Boolean - bool Character - char Integer - int Floating point - float Double floating point - double Valueless - void Wide character - wchar$_{t}Severalofthebasictypescanbemodifiedusingon$
$signed, unsigned, short, long$

Following is the example, which will produce correct size of various data types on your computer.

include <iostream> using namespace std;

int main() cout « "Size of char : " « sizeof(char) « endl; cout « "Size of int : " « sizeof(int) « endl; cout « "Size of short int : " « sizeof(short int) « endl; cout « "Size of long int : " « sizeof(long int) « endl; cout « "Size of float : " «

sizeof(float) « endl; cout « "Size of double : " « sizeof(double) « endl; cout « "Size of wchar$_t$: " $<< sizeof(wchar_t) << endl; return 0; StringStringBasic$

include <iostream> include <string> using namespace std ;

// assign a string string s1 = "www.java2s.com"; cout « s1;

// input a string string s2; cin » s2;

// concatenate two strings string s$_c = s1 + s2$;

// compare strings s1 == s2; Collection Pointer A pointer is a variable whose value is the address of another variable. Like any variable or constant, you must declare a pointer before you can work with it.

The general form of a pointer variable declaration is:

type *variable$_name; //example int*ip; //pointer to an integer double*dp; //pointer to a double float* fp; //pointer to a float char * ch; //pointer to character PointerLab$

include <iostream> using namespace std;

/* * Look at these lines */ int* a; a = new int[3]; a[0] = 10; a[1] = 2; cout « "Address of pointer a: a = " « a « endl; cout « "Value of pointer a: a = " « a « endl « endl; cout « "Address of a[0]: a[0] = " « a[0] « endl; cout « "Value of a[0]: a[0] = " « a[0] « endl; cout « "Value of a[0]: *a = " « *a « endl « endl; cout « "Address of a[1]: a[1] = " « a[1] « endl; cout « "Value of a[1]: a[1] = " « a[1] « endl; cout « "Value of a[1]: *(a+1)= " « *(a+1)« endl « endl; cout « "Address of a[2]: a[2] = " « a[2] « endl; cout « "Value of a[2]: a[2] = " « a[2] « endl; cout « "Value of a[2]: *(a+2)= " « *(a+2)« endl « endl; Result:

Address of pointer a: a = 008FF770 Value of pointer a: a = 00C66ED0

Address of a[0]: a[0] = 00C66ED0 Value of a[0]: a[0] = 10 Value of a[0]: *a = 10

Address of a[1]: a[1] = 00C66ED4 Value of a[1]: a[1] = 2 Value of a[1]: *(a+1)= 2

Address of a[2]: a[2] = 00C66ED8 Value of a[2]: a[2] = -842150451 Value of a[2]: *(a+2)= -842150451 Stack, Queue, Linked List, Array, Deque, List, Map, Set

Datetime The C++ standard library does not provide a proper date type. C++ inherits the structs and functions for date and time manipulation from C. To access date and time related functions and structures, you would need to include header file in your C++ program.

There are four time-related types: clock$_t, time_t, size_t, and tm. The types clock_t, size_t and time_t are capable of r$

The structure type tm holds the date and time in the form of a C structure having the following elements:

struct tm int tm$_sec; //seconds of minutes from 0 to 61 int tm_min; //minutes of hour from 0 to 59 int tm_hour; /$

Consider you want to retrieve the current system date and time, either as a local time or as a Coordinated Universal Time (UTC). Following is the example to achieve the same:

include <iostream> include <ctime>

using namespace std;

int main( ) // current date/time based on current system time$_t now = time(0)$;

// convert now to string form char* dt = ctime(now);

cout « "The local date and time is: " « dt « endl;

// convert now to tm struct for UTC tm *gmtm = gmtime(now); dt = asctime(gmtm); cout « "The UTC date and time is:"« dt « endl; When the above code is compiled and executed, it produces the following result:

The local date and time is: Sat Jan 8 20:07:41 2011

The UTC date and time is:Sun Jan 9 03:07:41 2011

4.4 Lập trình hướng đối tượng

Object Oriented Programming Classes and Objects include <iostream> using namespace std;

class Pacman

private: int x; int y; public: Pacman(int x, int y); void show(); ;

Pacman::Pacman(int x, int y) this->x = x; this->y = y;

void Pacman::show() std::cout « "(" « this->x « ", " « this->y « ")";

int main() // your code goes here Pacman p = Pacman(2, 3); p.show(); return 0; Template Function Template

include <iostream> include <string>

using namespace std;

template <typename T>

T Max(T a, T b) return a < b ? b : a;

int main()

int i = 39; int j = 20; cout « Max(i, j) « endl;

double f1 = 13.5; double f2 = 20.7; cout « Max(f1, f2) « endl;

string s1 = "Hello"; string s2 = "World"; cout « Max(s1, s2) « endl;

double n1 = 20.3; float n2 = 20.4; // it will show an error // Error: no instance of function template "Max" matches the argument list // arguments types are: (double, float) cout « Max(n1, n2) « endl; return 0;

4.5 Cơ sở dữ liệu

Database Sqlite with Visual Studio 2013 Step 1: Create new project 1.1 Create a new C++ Win32 Console application.

Step 2: Download Sqlite DLL

2.1. Download the native SQLite DLL from: http://sqlite.org/sqlite-dll-win32-x86-3070400.zip 2.2. Unzip the DLL and DEF files and place the contents in your project's source folder (an easy way to find this is to right click on the tab and click the "Open Containing Folder" menu item.

Step 3: Build LIB file

3.1. Open a "Developer Command Prompt" and navigate to your source folder. (If you can't find this tool, follow this post in stackoverflow Where is Developer Command Prompt for VS2013? to create it) 3.2. Create an import library using the following command line: LIB /DEF:sqlite3.def

Step 4: Add Dependencies

4.1. Add the library (i.e. sqlite3.lib) to your Project Properties -> Configuration Properties -> Linker -> Input -> Additional Dependencies. 4.2. Download http://sqlite.org/sqlite-amalgamation-3070400.zip 4.3. Unzip the sqlite3.h header file and place into your source directory. 4.4. Include the the sqlite3.h header file in your source code. 4.5. You will need to include the sqlite3.dll in the same directory as your program (or in a System Folder).

Step 5: Run test code

include "stdafx.h" include <ios> include <iostream> include "sqlite3.h"

using namespace std;

int $_tmain(int argc, _TCHAR* argv[]) int rc; char * error;$

// Open Database cout « "Opening MyDb.db ..." « endl; sqlite3 *db; rc = sqlite3$_open("MyDb.db", db); if (rc) cerr << "Error opening SQLite3 database : " << sqlite3_errmsg(db) << e$

// Execute SQL cout « "Creating MyTable ..." « endl; const char *sqlCreateTable = "CREATE TABLE MyTable (id INTEGER PRIMARY KEY, value STRING);"; rc = sqlite3$_exec(db, sqlCreateTable, NULL, NULL, error); if (rc) cerr << "Error executing SQ$

// Execute SQL cout « "Inserting a value into MyTable ..." « endl; const char *sqlInsert = "INSERT INTO MyTable VALUES(NULL, 'A Value');"; rc = sqlite3$_exec(db, sqlInsert, NULL, NULL, error); if (rc) cerr << "Error executing SQLite3 statement : " <<$

// Display MyTable cout « "Retrieving values in MyTable ..." « endl; const char *sqlSelect = "SELECT * FROM MyTable;"; char **results = NULL; int rows, columns; sqlite3$_get_table(db, sqlSelect, results, rows, columns, error); if (rc) cerr << "Error executing S$

// Display Cell Value cout.width(12); cout.setf(ios::left); cout « results[cellPosition] « " ";

// End Line cout « endl;

// Display Separator For Header if (0 == rowCtr) for (int colCtr = 0; colCtr < columns; ++colCtr) cout.width(12); cout.setf(ios::left); cout « " "; cout « endl; sqlite3$_free_table(results);$

// Close Database cout « "Closing MyDb.db ..." « endl; sqlite3$_close(db); cout << "Closed MyDb.db" << endl << endl;$

// Wait For User To Close Program cout « "Please press any key to exit the program ..." « endl; cin.get();

return 0;

4.6 Testing

Create Unit Test in Visual Studio 2013 Step 1. Create TDDLab Solution 1.1 Open Visual Studio 2013

1.2 File -> New Project... ->

Click Visual C++ -> Win32

Choose Win32 Console Application

Fill to Name input text: TDDLab

Click OK -> Next

1.3 In project settings, remove options:

Precompiled Header Securirty Develoment Lifecyde(SQL) check 1.4 Click Finish

Step 2. Create Counter Class 2.1 Right-click TDDLab -> Add -> Class...

2.2 Choose Visual C++ -> C++ Class -> Add

2.3 Fill in Class name box Counter -> Finish

2.4 In Counter.h file, add this below function

int add(int a, int b); 2.5 In Counter.cpp, add this below function

int Counter::add(int a, int b) return a+b; Your Counter class should look like this

Step 3. Create TDDLabTest Project 3.1 Right-click Solution 'TDDLab' -> Add -> New Project...

3.2 Choose Visual C++ -> Test

3.3 Choose Native Unit Test Project

3.4 Fill to Name input text: TDDLabTest

Step 4. Write unit test 4.1 In unittest1.cpp, add header of Counter class include "../TDDLab/Counter.h" 4.2 In TEST$_METHODfunction$

Counter counter; Assert::AreEqual(2, counter.add(1, 1)); 4.3 Click TEST in menu bar -> Run -> ‘All Test (Ctrl + R, A)

Step 5. Fix error LNK 2019: unresolved external symbol 5.1 Change Configuration Type of TDDLab project

Right click TDDLab project -> Properties General -> Configuration Type -> Static library (.lib) -> OK 5.2 Add Reference to TDDLabTest project

Right click TDDLabTest solution -> Properties -> Common Properties -> Add New Reference Choose TDDLab -> OK -> OK Step 6. Run Tests Click TEST in menu bar -> Run -> ‘All Test (Ctrl + R, A)

Test should be passed.

4.7 IDE Debugging

Visual Studio 2013 Install Extension

VsVim

googletest guide

Folder Structure with VS 2013

solution README.md |project1 | file011.txt | file012.txt | |project2 | file011.txt | file012.txt | Auto Format

Ctrl + K, Ctrl + D Git in Visual Studio

https://git-scm.com/book/en/v2/Git-in-Other-Environments-Git-in-Visual-Studio

Online IDE codechef ide

Chương 5

Javascript

View online http://magizbox.com/training/java/site/

What is Javascript? JavaScript is a high-level, dynamic, untyped, and interpreted programming language. It has been standardized in the ECMAScript language specification. Alongside HTML and CSS, it is one of the three core technologies of World Wide Web content production; the majority of websites employ it and it is supported by all modern Web browsers without plug-ins. JavaScript is prototype-based with first-class functions, making it a multi-paradigm language, supporting object-oriented, imperative, and functional programming styles. It has an API for working with text, arrays, dates and regular expressions, but does not include any I/O, such as networking, storage, or graphics facilities, relying for these upon the host environment in which it is embedded.

5.1 Installation

Google Chrome Pycharm

5.2 IDE

Google Chrome Developer Tools

The Chrome Developer Tools (DevTools for short), are a set of web authoring and debugging tools built into Google Chrome. The DevTools provide web developers deep access into the internals of the browser and their web application. Use the DevTools to efficiently track down layout issues, set JavaScript breakpoints, and get insights for code optimization.

5.3 Basic Syntax

1. Code Formatting Indent with 2 spaces

// Object initializer. var inset = top: 10, right: 20, bottom: 15, left: 12 ;

// Array initializer. this.rows$_{=}$[$'"Slartibartfast" < fjordmaster@magrathea.com >'$
$,'"ZaphodBeeblebrox" < theprez@universe.gov >','"FordPrefect" < ford@theguide.com >'$

$,' "ArthurDent" < has.no.tea@gmail.com >',' "MarvintheParanoidAndroid" < marv@googlemail.com >',' the.mice@magrathea.com']$;

// Used in a method call. goog.dom.createDom(goog.dom.TagName.DIV, id: 'foo', className: 'some-css-class', style: 'display:none' , 'Hello, world!'); 2. Naming functionNamesLikeThis variableNamesLikeThis ClassNamesLikeThis EnumNamesLikeThis methodNamesLikeThis CONSTANT$_V ALUES_L IKE_T HIS foo.namespaceNamesLikeThis.$

3.1 Class Comment /** * Class making something fun and easy. * @param string arg1 An argument that makes this more interesting. * @param Array.<number> arg2 List of numbers to be processed. * @constructor * @extends goog.Disposable */ project.MyClass = function(arg1, arg2) // ... ; goog.inherits(project.MyClass, goog.Disposable); 3.2 Method Comment /** * Operates on an instance of MyClass and returns something. * @param project.MyClass obj Instance of MyClass which leads to a long * comment that needs to be wrapped to two lines. * @return boolean Whether something occurred. */ function PR$_s omeMethod(obj)//...4.ExpressionandStatem$

22 "this is an epression" (5 > 6) ? false : true Statements The Simplest kind of stagement is an expression with a semi colon

!false; 5 + 6; 5. Loop and iteration while var number = 0; while (number <= 12) console.log(number); number = number + 2; do..while do var yourName = prompt("Who are you?"); while (!yourName); console.log(yourName); for for (var i = 0; i < 10; i++) console.log(i); 6. Function 6.1 Defining a Function var square = function(x) return x * x; ; square(5); 6.2 Scope Scope is the area where contains all variable or function are living. Scope has some rules: Child Scope can access all variable and function in parent Scope. (E.g: Local Scope can access Global Scope) function saveName(firstName) var temp = "temp"; function capitalizeName() temp = temp + " here"; return firstName.toUpperCase(); var capitalized = capitalizeName(); return capitalized; alert(saveName("Robert")); But parent Scope can access variable and function inside children scope (E.g: Global Scope cannot acces to local Scope) function talkDirty () var saying = "Oh, you little VB lover, you"; return alert(saying); alert(saying); //->Error 6.3 Call Stack The storage where computer stores context is called CALL STACK.

// CALL STACK function greet(who) console.log("Hello " + who); ask("How are you?"); console.log("I'm fine"); ;

function ask(question) console.log("well, " + question); ;

greet("Harry"); console.log("Bye"); Out of Call Stack

function chicken() return egg();

function egg() return chicken(); console.log(chicken() + " came first"); 6.4. Optional Argument We can pass too many or too few arguments to the function without any SyntaxError. If we pass too much arguments, the extra ones are ignored If we pass to few arguments, the missing ones get value undefined function power(base, exponent) if (exponent == undefined) exponent = 2; var result = 1; for (var count = 0; count < exponent; count++) result = result * base; return result; console.log(power(4)); console.log(power(4,3)); upside: flexible downside: hard to control the error

6.5 Closure Look at this example:

function sayHello(name) var text = 'Hello' + name; var say = function() console.log(text); return say; var say2 = sayHello("ahaha"); say2(); if in C program, does say2() work? The answer is nope! Because in C program, when a function returns, the Stack-flame will be destroyed, and all the local variable such as text will undefinded. So, when say2() is called, there is no text anymore, and the error, will be shown! But, in JavaScript, This code works!! Because, it provides for us an Object called Closure! Closure is borned when we define a function in another function, it keep all the live local variable. So, when say2() is called, the closure will give all the value of local variable outside it, and text will be identity.!

var globalVariable = 10; function func() var name = "xxx"; function getName() return name; function speak() var sound = "alo"; function scream() console.log(globalVariable); console.log(name); return "aaaaaaaaaa!"; function talk() var voice = getName() + " speak " + sound; console.log(voice); return voice; scream(); talk(); speak(); func(); 6.6. Recursion Recursion is function can call itself, as long as it is not overflow

function power(base, exponent) if (exponent == 0) return 1; else return base * power(base, exponent -1); console.log(power(2,3));

function FindSolution(target) function Find(start, history) if (start == target) return history; else if (start $>$ target) return null; else return Find(start + 5, "(" + history + " + 5 ") || Find(start * 3, "(" + history + " * 3)"); return Find(1, "1"); console.log(FindSolution(25)); 6.7. Arguments object The arguments object contains all parameters you pass to a function.

function argumentCounter() console.log("you gave me", arguments.length, "argument."); argumentCounter("Straw man", "Tautology", "Ad hominem"); 6.8. Higher-Order Function Filter array var ancestry = JSON.parse(ANCESTRY$_{F}ILE);console.log(ancestry$

function filter(array, test) var passed = []; for (var i = 0; i $<$ array.length; i++) if (test(array[i])) passed.push(array[i]); return passed; console.log(filter(ancestry, function(person) return person.born $>$ 1900 person.born $<$ 1925;));

TRANSFORMING WITH A MAP function map(array, transform) var mapped = []; for (var i = 0; i $<$ array.length; i++) mapped.push(transform(array[i])); return mapped;

var overNinety = ancestry.filter(function(person) return person.died - person.born $>$ 90;); console.log(map(overNinety, function(person) return person.name;));

REDUCE function reduce(array, combine, start) var current = start; for (var i = 0; i $<$ array.length; i++) current = combine(current, array[i]); return current; console.log(reduce([1, 2, 3, 4], function(a, b) return a + b; , 0)); Problem: using map and reduce, transform [1,2,3,4] to [1,2],[3,4]

var a = [1, 2, 3, 4] a = $_.map(a, function(i)if(ireturn[[],[i]]elsereturn[[i],[]]); a = _.reduce(a, function(x, y)return[x[0].concat(y[0]), x[1].concat(y[1])])$

BINDING FUNCTION var theSet = ["Carel Haverbeke", "Maria van Brussel", "Donald Duck"]; function isInSet(set, person) return set.indexOf(person.name) $>$ -1;

console.log(ancestry.filter(function(person) return isInSet(theSet, person);)); console.log(ancestry.filter(isInSet.bind(null, theSet))); What's the cleanest way to write a multiline string in JavaScript? [duplicate]

Google JavaScript Style Guide

5.4 Data Structure

5.4.1 Number

Some example of number: 10, 1.234, 1.99e9, NaN, Infinity, -Infinity console.log(2.99e9); console.log(0 /0); console.log(1 /0); console.log(-1 /0); Automatic Conversion

console.log(8 * null); // -> 0 console.log("5" - 1); // -> 4 console.log("5" + 1); //-> 51 console.log(false == 0) //-> true

5.4.2 String

sprintf In index.html

<script src="cdnjs.cloudflare.com/ajax/libs/sprintf/1.0.3/sprintf.js"/>

In script.js

// arguments sprintf(" hello sprintf

// object var user = name: "Dolly" sprintf("Hello Hello Dolly

// array of object var users = [name: "Dolly", name: "Molly"] sprintf("Hello Hello Dolly and Molly Multiline String str = " line 1 line 2 line 3"; Regular Expression in JavaScript This lab is based on Chapter9: EloquentJavaScript

Creating a regular expression There are 2 ways:

var re1 = new RegExp("abc"); var re2 = /abc/ there are some special characters such as question mark, or plus sign. If you want to use them, you have to use backslash. Like this:

var eighteen = /eighteen/; var question = /question/; Testing for match Regular Express has a number of method. Simplest is test

console.log(/abc/.test("abcd")); console.log(/abc/.test("abxde")); Matching a set of character []: Put a set of characters between 2 square bracket

console.log(/[0123456789]/.test("1245")); console.log(/[0-9]/.test("1"); console.log(/[0-9]/.test("acd"); console.log(/[0-9]/.test("aaascacas1")); There are some special character: Any digit character (Like [0-9])

var datetime = /--:/; console.log(datetime.test("16-06-2016 14:09")); console.log(dateTime.test("30-jan-2003 15:20")); An alphanumeric character ("word character")

var word = //; console.log(word.test("@@")); Any whitespace character (space, tab, newline, and similar)

var space = /+abc/; console.log(space.test("1. abd")); console.log(space.test("1. abd")); console.log(space.test("1.abd")); A character that is not a digit

var notDigit = //; console.log(notDigit.test("ww")); console.log(notDigit.test("1a")); console.log(notDigit.test("1124")); A nonalphanumeric character

var nonAlphanumbericChar = //; console.log(nonAlphanumbericChar.test("abc12231")); console.log(nonAlphanumbericChar.test("!@§A nonwhitespace character

var nonWhiteSpace = /§/; console.log(nonWhiteSpace.test("abc123")); console.log(nonWhiteSpace.test("1. abcd")); console.log(nonWhiteSpace.test(" ")); "." Any character except for newline

var anyThing = /.../; console.log(anyThing.test("abc.")); console.log(anyThing.test("acbacd.")); console.log(anyThing.test("acba")); ""$Usingcaretcharactertomatchanyexcepttheones$

var notBinary = /[^{0}1]/; $console.log(notBinary.test($"01101011100"$));$ $console.log(notBinary.test($"010210

"$Matchoneormore$" $*$ "$Matchzeroormore$

console.log(/+/.test(1234)); console.log(/+/.test());

console.log(/*/.test(1234)); console.log(/*/.test()) "?" Question mark test a character exist or not is still oke

var ball = /bal?l/; console.log(ball.test("ball")); console.log(ball.test("bal"));

a,b the character before exist from a to b times. Check datetime:

var datetime = /1,2-1,2-4 1,2:1,2/; console.log(datetime.test("20-12-2015 14:09")); var checkTimes = /waz3,5up/; console.log(checkTimes.test("wazzzzzup")); console.log(checkTimes.test("wazzzup")); console.log(checkTimes.test("wazup")); Grouping Subexpressions () using prentheses to make whole group like one character

var cartoonCrying = /boo+(hoo+)+/i; //i to match all Captalize or normal text console.log(cartoonCrying.test("Boohoooohoohooo")); console.log(cartoonCrying.test("boohoooohooO

Matches and group Test is a simplest method, and it only return true or false. exec (execute) is anther method in regex. It returns null if no match, and object if match.

var match = /+/.exec("one two 100"); console.log(match); console.log(match.input); console.log(match.index); if in the expression has a group subexpression, then it will return the text contain this subexpress, and the text match this subexpress:

var quotedText = /'([']*)'/;console.log(quotedText.exec("shesaid'hello'"));andifthesubexpressionappearsonemoretimes,th

console.log(/bad(ly)?/.exec("bad")); console.log(/()+/.exec("123")); The date type create new Date(). return the current time

var date = new Date(); console.log(new Date(2009, 11, 9); console.log(new Date(2009, 11, 9, 23, 59, 61)); <!–TimeStamp–> console.log(new Date(2009, 11, 9, 23, 59, 61).getTime()); console.log(new Date(1260378001000)); <!–getFullYear, getMonth,...–> var date = new Date(); console.log(date.getFullYear()); console.log(date.getMonth()); console.log(date.getDate()); console.log(date.getHours()); console.log(date.getMinutes()); console.log(date.getSeconds()); Word and string boundaries console.log(/cat/.test("concatenate")); console.log(/cat/.test("con123cat-129e0enate")); console.log(//.test("concatenate")); console.log(//.test("con123cat-129e0enate")); Choice pattern Only one in the list beween the "|" match

var animalCount = /+ (pig|cow|chicken)s?/; console.log(animalCount.test("15 pigs")); console.log(animalCount.test("15 pigchickens")); Replace Replace will find the first match and replace.if we want to replace all matches, using "g" behind the expresssion

console.log("papa".replace("p", "m")); console.log("Borobudur".replace(/[ou]/, "a")); console.log("Borobudur".replace(/[ou]/g, "a")); Replace can refer back to the matched, and using them

console.log("Le, Khanh, Hung, Bach".replace(/([]+), ([]+)/g, "12")); Greed function stripComments(code) return code.replace(/.*|[]*/g, "");console.log(stripComments("1+ /*2*/3"));//1+3console.log(stripComments("x = 10;//ten!"));//x = 10;console.log(stripComments("1/ a*/+/*b*/1"));//11SearchmethodSearchmethodreturnthefirstindexiftheregularexpressionmatch.Andre 1ifnotfound

console.log(" word".search(/§/)); // 2 console.log(" ".search(/§/)); // -1 The last index property In the regular expression has a property is lastIndex. And when this Regex do some method, it will start from the lastIndex. And after doing something, the lastIndex will update to the behind the index of the match exec.

var pattern = /y/g; pattern.lastIndex = 3; //lastIndex update to 3 var match = pattern.exec("xyzzy"); //lastIndex update to 5 console.log(pattern.lastIndex);

match = pattern.exec("xyzzyxxx"); //Not match any "y" from index 5 console.log(match.index); console.log(pattern.lastIndex); Looping Over the Line Applying the hepoloris of lastIndex, we can using while to do something like this:

var input = "A string with 3 numbers in it... 42 and 88."; var number = /(+)/g; var match; while (match = number.exec(input)) console.log("Found", match[1], "at", match.index);

5.4.3 Collection

Some useful methods with array push and pop var a = [1,2,3,4]; console.log(a.pop(), a); console.log(a.push(3), a); shift and unshift console.log(a.shift(), a); console.log(a.unshift(1), a); indexOf and lastIndexOf var b = [1,2,3,4,2,3,1]; console.log(b.indexOf(1)); console.log(b.lastIndexOf(1)); slice console.log([0,1,2,3,4].slice(2,4)); console.log([0,1,2,3,4].slice(2)); concat var a = [1,2,3]; var b = [4,5,6]; a.concat(b); console.log(a);

5.4.4 Datetime

Current Time moment().format('MMMM Do YYYY, h:mm:ss a'); Moment.js

5.4.5 Boolean

Boolean has only 2 values: true and false

console.log("Abc" < "Abcd") // -> true console.log("abc" < "Abcd") // -> false console.log("123" == "123") // -> true console.log(NaN == NaN) // -> false what is the different?

console.log("5" == 5); console.log("5" === 5);

5.4.6 Object

Object Define an object var object = number: 10, string: "string", array: [1,2,3], object: a: 1, b: 2 Add new property to object object.newProperty = "value"; object['key'] = 'value'; delete property delete object.newProperty; Window object (global object) The Global scope is stored in an object which called window

function test() var local = 10; console.log("local" in window); console.log(window.local); test(); var global = 10; console.log("global" in window); console.log(window.global);

5.5 OOP

1. Classes and Objects Constructor function Ball(position) this.position = position; this.display = function() console.log(this.position[0], ", ", this.position[1]);

ball = new Ball([2, 3]); ball.display(); 2. Inheritance Person = function (name, birthday, job) this.name = name; this.birthday = birthday; this.job = job; ;

Person.prototype.display = function () console.log(this.name, ""); console.log(this.birthday, ""); console.log(this.job, ""); ;

Politician = function (name, birthday) Person.call(this, name, birthday, "Politician"); ; Politician.prototype = Object.create(Person.prototype); Politician.prototype.constructor = Politician;

var person1 = new Person("Barack Obama", "04/08/1961", "Politician"); var person2 = new Politician("David Cameron", "09/10/1966"); person1.display(); person2.display();

Object-Oriented Programming var rabbit = ; rabbit.speak = function(line) console.log("The rabit says:'" + line + "'"); ; rabbit.speak("I'm alive");

function speak(line) console.log("The "+ this.type + " rabbit says '" + line + "'");

var whiteRabbit = type: "white", speak: speak; var fatRabbit = type: "fat", speak: speak; whiteRabbit.speak("Oh my ears and whiskers, " + "how late it's getting!"); fatRabbit.speak("I could sure use a carrot right now");

// Prototype // Prototype is another object that is used as a fallback source of properties // When object request a property that it does not have, its prototype will be searched for the property var empty = ; console.log(empty.toString); console.log(empty.toString);

// Get prototype of an object 2 ways: console.log(Object.getPrototypeOf() == Object.prototype); console.log(Object.getPrototypeOf(Object.prototype));

// Using Object.create to create an object with an specific prototype var protoRabbit = speak: function(line) console.log("The " + this.type + " rabbit says '" + line + "'"); ;

var killerRabbit = Object.create(protoRabbit); killerRabbit.type = "Killer"; killerRabbit.speak("Skreeee!");

// Constructor function Rabbit(type) this.type = type; var killerRabbit = new Rabbit("Killer"); var blackRabbit = new Rabbit("black"); console.log(blackRabbit.type);

// using prototype to add a new method Rabbit.prototype.speak = function(line) console.log("The " + this.type + " rabit says '" + line + "'"); ; blackRabbit.speak("Doom...");

// OVERRIDING DERIVED PROPERTIES Rabbit.prototype.teeth = "small"; console.log(killerRabbit.teeth);

killerRabbit.teeth = "Long, sharp, and bloody"; console.log(killerRabbit.teeth); console.log(blackRabbit.teeth); console.log(Rabbit.prototype.teeth);

// PROTOTYPE INTERFERENCE // A prototype can be used at any time to add methods, properties // to all objects based on it Rabbit.prototype.dance = function () console.log("The " + this.type + " rabbit dances a jig"); ; killerRabbit.dance(); // but there is a problem: var map = ; function storePhi(event, phi) map[event] = phi;

storePhi("pizza", 0.069); storePhi("touched tree", -0.081); console.log(map);

Object.prototype.nonsense = "hi"; for (var name in map) console.log(name); console.log("nonsense" in map); console.log("toString" in map); delete Object.prototype.nonsense; // we can use Object.defineProperty to solve it Object.defineProperty(Object.prototype, "hiddenNonsense", enumerable: false, value: "hi");

for (var name in map) console.log(name); console.log(map.hiddenNonsense); // but there still has a problem console.log("toString" in map); console.log(map.hasOwnProperty("toString"))

// PROTOTYPE-LESS OBJECTS // if we only want to create an fresh object, without prototype then we tranform null to create var map = Object.create(null); map["pizza"] = 0.09; console.log("toString" in map); console.log("pizza" in map);

// POLYMORPHISM // laying out a table: example for polymorphism function rowHeights(rows) return rows.map(function(row) return row.reduce(function(max, cell) return Math.max(max, cell.minHeight()); , 0););

function colWidths(rows) return rows[0].map(function($_, i) return rows.reduce(function(max, row) return$

function drawTable(rows) var heights = rowHeights(rows); var widths = colWidths(rows);

function drawLine(blocks, lineNo) return blocks.map(function(block) return block[lineNo];).join(" ");

function drawRow(row, rowNum) var blocks = row.map(function(cell, colNum) return cell.draw(widths[colNum], heights[rowNum]);); return blocks[0].map(function($_, lineNo) return d$

return rows.map(drawRow).join("");

function repeat(string, times) var result = ""; for (var i = 0; i < times; i++) result += string; return result;

function TextCell(text) this.text = text.split(""); TextCell.prototype.minWidth = function() return this.text.reduce(function(width, line) return Math.max(width, line.lenght); , 0); ; TextCell.prototype.minHeight = function() return this.text.length; TextCell.prototype.minHeight = function() return this.text.lenght; TextCell.prototype.minHeight = function() return this.text.length; TextCell.prototype.draw = function(width, height) var result = []; for (var i = 0; i < height; i++) var line = this.text[i] || ""; result.push(line + repeat(" ", width - line.length)); return result;

var rows = []; for (var i = 0; i < 5; i++) var row = []; for (var j = 0; j < 5; j++) if ((i + j) row.push(new TextCell("1234")); else row.push(new TextCell("5")); rows.push(row); console.log(drawTable(rows));

// // GETTERS AND SETTERS // var pile = // elements: ["eggshell", "orange peel", "worm"], // get height() // return this.elements.length; // , // set height(value) // console.log("Ignoring attemp to set high to ", value); // // ;

// console.log(pile.height); // pile.height = 100; // console.log(pile.height);

[1]: Introduction to Object-Oriented JavaScript [2]: How to call parent constructor?

5.6 Networking

POST $.ajax(type : "POST", url : "http : //service.com/items", data : JSON.stringify("name" : "newiten$

5.7 Logging

Javascript Logging Having a fancy JavaScript debugger is great, but sometimes the fastest way to find bugs is just to dump as much information to the console as you can.

console.log console.assert console.error

5.8 Documentation

Components jsdoc (with docdash template)

JSDoc is an API documentation generator for JavaScript, similar to JavaDoc or PHPDoc. You add documentation comments directly to your source code, right along side the code itself. The JSDoc Tool will scan your source code, and generate a complete HTML documentation website for you.

gulp, PyCharm

Usage Step 1. Install gulp-jsdoc npm install –save-dev gulp gulp-jsdoc docdash Step 2. Create documentation task Create documentation task in gulpfile.js file

var template = "path": "./node$_modules/docdash$";

gulp.task('docs', function() return gulp.src("./src/*.js") .pipe(jsdoc('./docs', template));); Step 3. Refresh Gulp tasks In pycharm, click to refresh button in gulp window.

Step 4. Add comment to your code Add comment to your code, You can see an example: should.js

/** * Simple utility function for a bit more easier should assertion * extension * @param Function f So called plugin function. It should accept * 2 arguments: ‘should‘ function and ‘Assertion‘ constructor * @memberOf should * @returns Function Returns ‘should‘ function * @static * @example * * should.use(function(should, Assertion) * Assertion.add(’asset’, function() * this.params = operator: ’to be asset’ ; * * this.obj.should.have.property(’id’).which.is.a.Number(); * this.obj.should.have.property(’path’); *) *) */ should.use = function(f) f(should, should.Assertion); return this; ;

Types: boolean, string, number, Array (see more)

Step 5. Run docs task In pycharm, click to docs task in gulp window.

5.9 Error Handling

In javascript bugs may be displayed is NaN or underfined and program still run but after that, the wrong value can cause some mistake when we use it So, finding bugs and fix them is the quiet hard work in javascript But we can do, and this job is called debugging

STRICT MODE This is the way to find errors that javascript ignores. Example is using an undefined variable. if we dont use strick mode, then everything will be ok, but if using, the error will be shown

function SpotProblem() // "use strict"; for (counter = 0; counter < 10; counter++) console.log("Good!"); SpotProblem(); console.log(counter); strick mode can find error when using this in local, but it is still in global. Example: When we forget to declare the key word "new" when create an new Object

"use strict"; function Person(name) this.name = name; var john = Person("John"); console.log(name); And there are another cases, that trick mode is not allowed: Delete an object is not allowed

"use strict"; var x = 3.14; delete x;

"use strict"; var obj = v1: 3, v2: 4; delete pbj;

"use strict"; var func = function(); delete func; Duplicate parameter is not allowed

"use strict"; var func = function(a1, a1) console.log(a1); Reserve Word is not allowed to name variable

"use strict"; var arguments = 5; var eval = 6; console.log(arguments); console.log(eval); TESTING Testing makes sure that the program working well, and if there are any changes, testing will automatic show us the error, thus, we know where need to fix

function Vector(x, y) this.x = x; this.y = y; Vector.prototype.plus = function(other) return new Vector(this.x + other.x, this.y + other.y);

function TestVector() var p1 = new Vector(10, 20); var p2 = new Vector(-10, 5); var p3 = p1.plus(p2);

if (p1.x !== 10) return "fail: x property"; if (p1.y !== 20) return "fail: y property"; if (p2.x !== -10) return "fail: nagative x property"; if (p2.y !== 5) return "fail: y property"; if (p3.x !== 0) return "fail: x property from plus"; if (p3.y !== 25) return "fail: y property from plus"; return "Vector is Oke"; TestVector(); DEBUGGING when the testing is fail, we have to debug to find the bugs. The first we should guess the bug. And then we put break point in the line, we assume it make bug If that is the exactly bug we want to find, then we fix it, and write more test for this case In this example code below, the function convert the number in the decima to another. we run and see the result is wrong, so we guess that the error may be caused by the result variable, then we put break point in the line contains result variable.

function ConvertNumber(n, base) var result = "", sign = ""; if (n $<$ 0) sign = "-"; n = -n; do result = String(n n /= base; //-> n = Math.floor(n / base); while (n $>$ 0); return sign + result; console.log(ConvertNumber(13, 10)); console.log(ConvertNumber(14, 2)); ERROR PROPAGATION Sometime our code is working well with normal input. But with special one, they can cause error. So, we have to consider all situation can make Flaws, and handling them. This example code below has an if..else to handle the wrong input if user types not a number in the prompt input

function promptNumber(question) var result = Number(prompt(question, "")); if (isNaN(result)) return null; else return result; console.log(promptNumber("How many trees do you see?")); EXCEPTION In the Error Propagation, we can control the errors if we know them. But what will happen if we don't know the error? For solving this problem, javascript provides for us an try...catch.. to control error we dont know or not sure

try throw new Error("Invalid defination"); catch (error) console.log(error);

function promtDirection(question) var result = prompt(question, ""); if (result.toLowerCase() == "left") return "L"; if (result.toLowerCase() == "right") return "R"; throw new Error("Invalid direction: " + result);

function look() if (promtDirection("Which way?") == "L") return "a house"; else return "two angry bears";

try console.log("you see", look()); catch (error) console.log("Something went wrong: " + error); CLEAN UP AFTER EXCEPTIONS We have a block of code below:

var context = null; function withContext(newContext, body) var oldContext = context; context = newContext; var result = body(); context = oldContext; return result; withContext("new", function() var a = b/0; return a;); What would happend with context? It cannot be excute the last line code, because in withContext function, it will throw off the stack by an exception. So javascript provides a try...finally...

var context = null; function withContext(newContext, body) var oldContext = context; context = newContext; try return body(); finally context = oldContext; withContext("new", function() var a = b/0; return a;); SELECTIVE CATCHING There are some errors cannot handle by environment. So, if we let the error go through, it can cause broken program. For examnple, the Error() in environment cannot catch the infinitive loop in the try block, if we dont catch this problem, the programm will crash soon

for (;;) try var dir = promtDirection("Where?"); console.log("You chose ", dir); break; catch (e) console.log("Not a valid direction. Try again."); The loop will break out if the promptDirection() can excute. But it doesn't. Because it is not defined before, so the environment catch it and go through the catch to show error The circle again and again will make the program crash. So we will create a special Exception.

function InputError(message) this.message = message; this.stack = (new Error()).stack; InputError.prototype = Object.create(Error.prototype); InputError.prototype.name = "InputError"; Error: has an property is stack. it contains all exception, which environment can catch. Then, we have the promptDirection function to return the result if Enter valid format, or an exception if invalid function promptDirection(question) var result = prompt(question, ""); if (result.toLowerCase() == "left") return "L"; if (result.toLowerCase() == "right") return "R"; throw new InputError("Invalid direction: " + result); Finally, we can catch all exception we want

for (;;) try var dir = promptDirection("Where?"); console.log("You choose ", dir); break; catch(e) if (e instanceof InputError) console.log("Not a valid direction. Try again. "); else throw e; ASSERTIONS function AssertionFailed(message) this.message = message; AssertionFailed.prototype = Object.create(Error.prototype);

function assert(test, message) if (!test) throw new AssertionFailed(message);

function lastElement(array) assert(array.length > 0, "empty array in lastElement"); return array[array.length - 1];

5.10 Testing

Mocha Mocha is a feature-rich JavaScript test framework running on Node.js and the browser, making asynchronous testing simple and fun. Mocha tests run serially, allowing for flexible and accurate reporting, while mapping uncaught exceptions to the correct test cases.

Installation bower install -D mocha chai Usage Step 1. Make index.html

<!DOCTYPE html> <html> <head> <meta charset="utf-8"> <title>Tests</title> <link rel="stylesheet" media="all" href="mocha.css"> </head> <body> <div id="mocha"></div> <script src="mocha.js"></script> <script src="chai.js"></script> <script src="functions.js"></script> <script>mocha.setup('bdd'); chai.should();</script> <script src="tests.js"></script> <script>mocha.run();</script> </body> </html> Step 2. Edit functions.js

function sum(a, b) return a + b;

function asynchronusSum(a, b) return new Promise(function(fulfill, reject) fulfill(a + b);); Step 3. Edit tests.js

describe('Calculator', function() this.timeout(5000); describe('sum()', function() it('should return sum of two number', function() sum(2, 3).should.equal(5)););

describe('asynchronusSum()', function() it('should return sum of two number', function(done) asynchronusSum(2, 3).then(function(output) output.should.equal(5); done();));););

5.11 Package Manager

Bower A package manager for the web

Web sites are made of lots of things — frameworks, libraries, assets, utilities, and rainbows. Bower manages all these things for you.

Bower works by fetching and installing packages from all over, taking care of hunting, finding, downloading, and saving the stuff you're looking for. Bower keeps track of these packages in a manifest file, bower.json. How you use packages is up to you. Bower provides hooks to facilitate using packages in your tools and workflows.

Bower is optimized for the front-end. Bower uses a flat dependency tree, requiring only one version for each package, reducing page load to a minimum.

http://bower.io/

[code] bower install jquery underscore moment sprintf -S [/code]

HTML <bower based>

<script src="./bower$_c$*omponents/jquery/dist/jquery.js*” >< /*script* >< *scriptsrc* = ”*./bower*$_c$*omponents/moment/moment.js*” >< /*script* >< *scriptsrc* = ”*./bower*$_c$*omponents/underscore/underscore.js*” >< /*script* >< *scriptsrc* = ”*./bower*$_c$*omponents/sprintf/src/sprintf.js*” >< /*script* > *HTML* < *cdnbased* >

<script src="//cdnjs.cloudflare.com/ajax/libs/jquery/3.0.0-beta1/jquery.js"></script> <script src="//cdnjs.cloudflare.com/ajax/libs/underscore.js/1.8.3/underscore.js"></script> <script src="//cdnjs.cloudflare.com/ajax/libs/sprintf/1.0.3/sprintf.js"></script>

5.12 Build Tool

Gulp

Automate and enhance your workflow

Here's some of the sweet stuff you try out with this repo.

Compile CoffeeScript (with source maps!) Compile Handlebars Templates Compile SASS with Compass LiveReload require non-CommonJS code, with dependencies Set up module aliases Run a static Node server (with logging) Pop open your app in a Browser Report Errors through Notification Center Image processing Installation npm install -S gulp gulp-concat Usage Watch

var gulp = require('gulp'); var concat = require('gulp-concat'); var uglify = require('gulp-uglify'); var jsdoc = require("gulp-jsdoc");

var third$_p$*arties* = [”*bower*$_c$*omponents/jquery/dist/jquery.js*”*,*”*bower*$_c$*omponents/bootstrap/dist/js/b*

var modules = ["modules/your$_s$*cript.js*”];

gulp.watch(third$_p$*arties*, [′*js*$_t$*hirdparty*′]); *gulp.watch*(*modules*, [′*js*$_m$*odules*′]);

gulp.task('js$_t$*hirdparty*′, *function*()*returngulp.src*(*third*$_p$*arties*)*.pipe*(*concat*(′*third*$_p$*arty.uglify.js*′))*.pip*

gulp.task('js$_m$*odules*′, *function*()*returngulp.src*(*modules*)*.pipe*(*concat*(′*modules.uglify.js*′))//*.pipe*(*ugli*

gulp.task('documentation', function () return gulp .src("./modules/*/*.js") .pipe(jsdoc('./documentation')););

gulp.task('default', ['js$_t$*hirdparty*′,′ *js*$_m$*odules*′]); *http* : //*gulpjs.com/*

Deprecated grunt

5.13 Make Module

Make Module sample modules: underscore, momentjs

Folder Structure |- docs |- test |- src | |– your$_m$*odule.js*| − *.gitignore*| − *bower.json*

Chương 6

Java

01/11/2017: Java đơn giản là gay nhé. Không chơi. Viết java chỉ viết thế này thôi. Không viết hơn. Thề!

View online http://magizbox.com/training/java/site/

Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere" (WORA), meaning that compiled Java code can run on all platforms that support Java without the need for recompilation. Java applications are typically compiled to bytecode that can run on any Java virtual machine (JVM) regardless of computer architecture. As of 2016, Java is one of the most popular programming languages in use, particularly for client-server web applications, with a reported 9 million developers. Java was originally developed by James Gosling at Sun Microsystems (which has since been acquired by Oracle Corporation) and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++, but it has fewer low-level facilities than either of them.

6.1 Get Started

Installation Ubuntu Step 1. Download sdk

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html Step 2. Create folder jvm

sudo mkdir /usr/lib/jvm/ Step 3. cd to folder downloads jdk and run command

sudo mv jdk1.7.0$_x//usr/lib/jvm/jdk1.7.0_xRuninstalljavasudoupdate-alternatives--install/usr/bin/javajava/usr/lib/jvm/jdk1.7.0_x/jre/bin/java0Addpathjdk: /usr/lib/jvm/jdk1.7.0_x$

su - nano /etc/environment

6.2 Basic Syntax

Variable Types Although Java is object oriented, not all types are objects. It is built on top of basic variable types called primitives.

Here is a list of all primitives in Java:

byte (number, 1 byte) short (number, 2 bytes) int (number, 4 bytes) long (number, 8 bytes) float (float number, 4 bytes) double (float number, 8 bytes) char (a character, 2 bytes) boolean (true or false, 1 byte) Java is a strong typed language, which means variables need to be defined before we use them. Numbers To declare and assign a number use the following syntax:

int myNumber; myNumber = 5; Or you can combine them:

int myNumber = 5; To define a double floating point number, use the following syntax:

double d = 4.5; d = 3.0; If you want to use float, you will have to cast:

float f = (float) 4.5; Or, You can use this:

float f = 4.5f (f is a shorter way of casting float) Characters and Strings In Java, a character is it's own type and it's not simply a number, so it's not common to put an ascii value in it, there is a special syntax for chars:

char c = 'g'; String is not a primitive. It's a real type, but Java has special treatment for String.

Here are some ways to use a string:

// Create a string with a constructor String s1 = new String("Who let the dogs out?"); // Just using "" creates a string, so no need to write it the previous way. String s2 = "Who who who who!"; // Java defined the operator + on strings to concatenate: String s3 = s1 + s2; There is no operator overloading in Java! The operator + is only defined for strings, you will never see it with other objects, only primitives.

You can also concat string to primitives:

int num = 5; String s = "I have " + num + " cookies"; //Be sure not to use "" with primitives. boolean Every comparison operator in java will return the type boolean that not like other languages can only accept two special values: true or false.

boolean b = false; b = true;

boolean toBe = false; b = toBe || !toBe; if (b) System.out.println(toBe);

int children = 0; b = children; // Will not work if (children) // Will not work // Will not work Operators Java provides a rich set of operators to manipulate variables. We can divide all the Java operators into the following groups:

Arithmetic Operators Relational Operators Bitwise Operators Logical Operators Assignment Operators Misc Operators The Arithmetic Operators Arithmetic operators are used in mathematical expressions in the same way that they are used in algebra.

The following table lists the arithmetic operators:

Operator Description Example + (Addition) Adds values on either side of the operator 10 + 20 -> 30 - (Subtraction) Subtracts right hand operand from left hand operand 10 - 20 -> -10 * (Multiplication) Multiplies values on either side of the operator 10 * 20 -> 200 / (Division) Divides left hand operand by right hand operand 20 / 10 -> 2 ++ (Increment) Increases the value of operand by 1 a = 20

a++ -> 21

– (Decrement) Decreases the value of operand by 1 a = 20

a– -> 19

The Relational Operators There are following relational operators supported by Java language

== (equal to) Checks if the values of two operands are equal or not, if yes then condition becomes true.

Example: (A == B) is not true. 2 != (not equal to) Checks if the values of two operands are equal or not, if values are not equal then condition becomes true.

Example: (A != B) is true.

3 > (greater than) Checks if the value of left operand is greater than the value of right operand, if yes then condition becomes true.

Example: (A > B) is not true. 4 < (less than) Checks if the value of left operand is less than the value of right operand, if yes then condition becomes true.

Example: (A < B) is true. 5 >= (greater than or equal to) Checks if the value of left operand is greater than or equal to the value of right operand, if yes then condition becomes true.

Example (A >= B) is not true. 6 <= (less than or equal to) Checks if the value of left operand is less than or equal to the value of right operand, if yes then condition becomes true.

example(A <= B) is true.

The Bitwise Operators Java defines several bitwise operators, which can be applied to the integer types, long, int, short, char, and byte.

Bitwise operator works on bits and performs bit-by-bit operation. Assume if a = 60; and b = 13; now in binary format they will be as follows:

a = 0011 1100

b = 0000 1101

ab = 0000 1100

a|b = 0011 1101

a^b = 00110001

a = 1100 0011

The following table lists the bitwise operators:

Assume integer variable A holds 60 and variable B holds 13 then:

(bitwise and) Binary AND Operator copies a bit to the result if it exists in both operands.

Example: (A B) will give 12 which is 0000 1100 2 | (bitwise or) Binary OR Operator copies a bit if it exists in either operand.

Example: (A | B) will give 61 which is 0011 1101 3 $^(bitwise XOR) Binary XOR Operator copies the bit if it is s$

Example: (A $^B)$ will give 49 which is 00110001 4 $(bitwise compliment) Binary Ones Complement Operator is u$

Example: (A) will give -61 which is 1100 0011 in 2's complement form due to a signed binary number. 5 « (left shift) Binary Left Shift Operator. The left operands value is moved left by the number of bits specified by the right operand

Example: A « 2 will give 240 which is 1111 0000 6 » (right shift) Binary Right Shift Operator. The left operands value is moved right by the number of bits specified by the right operand.

Example: A » 2 will give 15 which is 1111 7 »> (zero fill right shift) Shift right zero fill operator. The left operands value is moved right by the number of bits specified by the right operand and shifted values are filled up with zeros.

Example: A »>2 will give 15 which is 0000 1111

The Logical Operators The following table lists the logical operators:

Assume Boolean variables A holds true and variable B holds false, then:

(logical and) Called Logical AND operator. If both the operands are non-zero, then the condition becomes true.

Example (A B) is false. 2 || (logical or) Called Logical OR Operator. If any of the two operands are non-zero, then the condition becomes true.

Example (A || B) is true. 3 ! (logical not) Called Logical NOT Operator. Use to reverses the logical state of its operand. If a condition is true then Logical NOT operator will make false.

Example !(A B) is true.

The Assignment Operators There are following assignment operators supported by Java language:

Show Examples

SR.NO Operator and Description 1 = Simple assignment operator, Assigns values from right side operands to left side operand.

Example: C = A + B will assign value of A + B into C 2 += Add AND assignment operator, It adds right operand to the left operand and assign the result to left operand.

Example: C += A is equivalent to C = C + A 3 -= Subtract AND assignment operator, It subtracts right operand from the left operand and assign the result to left operand.

Example:C -= A is equivalent to C = C - A 4 *= Multiply AND assignment operator, It multiplies right operand with the left operand and assign the result to left operand.

Example: C *= A is equivalent to C = C * A 5 /= Divide AND assignment operator, It divides left operand with the right operand and assign the result to left operand

ExampleC /= A is equivalent to C = C / A 6

Example: C

ExampleC «= 2 is same as C = C « 2 8 »= Right shift AND assignment operator

Example C »= 2 is same as C = C » 2 9 = Bitwise AND assignment operator.

Example: C = 2 is same as C = C 2 10 $^=bitwiseexclusiveORandassignmentoperator.$

Example: C $^=2issameasC = C^2 11| = bitwiseinclusiveORandassignmentoperator.$

Example: C |= 2 is same as C = C | 2

Miscellaneous Operators There are few other operators supported by Java Language.

Conditional Operator (? :) Conditional operator is also known as the ternary operator. This operator consists of three operands and is used to evaluate Boolean expressions. The goal of the operator is to decide which value should be assigned to the variable. The operator is written as:

variable x = (expression) ? value if true : value if false Following is the example:

public class Test

public static void main(String args[]) int a, b; a = 10; b = (a == 1) ? 20: 30; System.out.println("Value of b is : " + b);

b = (a == 10) ? 20: 30; System.out.println("Value of b is : " + b); This would produce the following result ?

Value of b is : 30 Value of b is : 20 Precedence of Operators Operator precedence determines the grouping of terms in an expression. This affects how an expression is evaluated. Certain operators have higher precedence than others; for example, the multiplication operator has higher precedence than the addition operator:

For example, x = 7 + 3 * 2; here x is assigned 13, not 20 because operator * has higher precedence than +, so it first gets multiplied with 3*2 and then adds into 7.

Here, operators with the highest precedence appear at the top of the table, those with the lowest appear at the bottom. Within an expression, higher precedence operators will be evaluated first.

Category Operator Associativity Postfix () [] . (dot operator) Left toright Unary ++ - - ! Right to left Multiplicative * /

Conditional Java uses boolean variables to evaluate conditions. The boolean values true and false are returned when an expression is compared or evaluated. For example:

int a = 4; boolean b = a == 4;

if (b) System.out.println("It's true!"); Of course we don't normally assign a conditional expression to a boolean, we just use the short version:

int a = 4;

if (a == 4) System.out.println("Ohhh! So a is 4!"); Boolean operators There aren't that many operators to use in conditional statements and most of them are pretty strait forward:

int a = 4; int b = 5; boolean result; result = a < b; // true result = a > b; // false result = a <= 4 // a smaller or equal to 4 - true result = b >= 6 // b bigger or equal to 6 - false result = a == b // a equal to b - false result = a != b // a is not equal to b - true result = a > b || a < b // Logical or - true result = 3 < a a < 6 // Logical and - true result = !result // Logical not - false if - else and between The if, else statement in java is pretty simple.

if (a == b) // a and b are equal, let's do something cool And we can also add an else statement after an if, to do something if the condition is not true

if (a == b) // We already know this part else // a and b are not equal... :/ The if - else statements doesn't have to be in several lines with , if can be used in one line, or without the , for a single line statment.

if (a == b) System.out.println("Another line Wow!"); else System.out.println("Double rainbow!"); Although this method might be useful for making your code shorter by using fewer lines, we strongly recommend for beginners not to use this short version of statements and always use the full version with . This goes to every statement that can be shorted to a single line (for, while, etc).

The ugly side of if There is a another way to write a one line if - else statement by using the operator ? :

int a = 4; int result = a == 4 ? 1 : 8;

// result will be 1 // This is equivalent to int result;

if (a == 4) result = 1; else result = 8; Again, we strongly recommend for beginners not to use this version of if.

== and equals The operator == works a bit different on objects than on primitives. When we are using objects and want to check if they are equal, the operator == will say if they are the same, if you want to check if they are logically equal, you should use the equals method on the object. For example:

String a = new String("Wow"); String b = new String("Wow"); String sameA = a;

boolean r1 = a == b; // This is false, since a and b are not the same object boolean r2 = a.equals(b); // This is true, since a and b are logically equals boolean r3 = a == sameA; // This is true, since a and sameA are really the same object

6.3 Data Structure

Data Structure Number, String Convert number to string

String.valueOf(1000) Make a random

// create a random number from 0 to 99 (new Random()).nextInt(100) Collection Arrays Arrays in Java are also objects. They need to be declared and then created. In order to declare a variable that will hold an array of integers, we use the following syntax:

int[] arr; Notice there is no size, since we didn't create the array yet.

arr = new int[10]; This will create a new array with the size of 10. We can check the size by printing the array's length:

System.out.println(arr.length); We can access the array and set values:

arr[0] = 4; arr[1] = arr[0] + 5; Java arrays are 0 based, which means the first element in an array is accessed at index 0 (e.g: arr[0], which accesses the first element). Also, as an example, an array of size 5 will only go up to index 4 due to it being 0 based.

int[] arr = new int[5] //accesses and sets the first element arr[0] = 4; We can also create an array with values in the same line:

int[] arr = 1, 2, 3, 4, 5; Don't try to print the array without a loop, it will print something nasty like [I@f7e6a96.

Set

import java.util.HashSet; import java.util.Set;

public class HelloWorld

public static void main(String []args) Set<Dog> dogs = new HashSet<Dog>(); Dog dog1 = new Dog("a", 1); Dog dog2 = new Dog("a", 2); Dog dog3 = new Dog("a", 1); Dog dog4 = new Dog("b", 1); dogs.add(dog1); dogs.add(dog2); dogs.add(dog3); dogs.add(dog4); System.out.println(dogs.size());

// 3 public class Dog public String name; public int age; public int value; public Dog(String name, int age) this.name = name; this.age = age; value = (this.name + String.valueOf(this.age)).hashCode();

@Override public int hashCode() return value;

@Override public boolean equals(Object obj) return (obj instanceof Dog ((Dog) obj).value == this.value); List List<String> places = Arrays.asList("Buenos Aires", "Córdoba", "La Plata"); Datetime Calendar c = Calendar.getInstance(); Suggest Readings Initialization of an ArrayList in one line How to convert from int to String?

6.4 OOP

6.4.1 Classes

Java is an Object-Oriented Language. As a language that has the Object-Oriented feature, Java supports the following fundamental concepts

Classes and Objects Encapsulation Inheritance Polymorphism Abstraction Instance Method Message Parsing In this chapter, we will look into the concepts - Classes and Objects.

Object Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating. An object is an instance of a class. Class A class can be defined as a template/blueprint that describes the behavior/state that the object of its type support. Objects Let us now look deep into what are objects. If we consider the real-world, we can find many objects around us, cars, dogs, humans, etc. All these objects have a state and a behavior.

If we consider a dog, then its state is - name, breed, color, and the behavior is - barking, wagging the tail, running.

If you compare the software object with a real-world object, they have very similar characteristics.

Software objects also have a state and a behavior. A software object's state is stored in fields and behavior is shown via methods.

So in software development, methods operate on the internal state of an object and the object-to-object communication is done via methods.

Classes A class is a blueprint from which individual objects are created.

Following is a sample of a class.

Example

public class Dog String breed; int ageC String color;

void barking()

void hungry()

void sleeping() A class can contain any of the following variable types.

Local variables Variables defined inside methods, constructors or blocks are called local variables. The variable will be declared and initialized within the method and the variable will be destroyed when the method has completed. Instance variables Instance variables are variables within a class but outside any method. These variables are initialized when the class is instantiated. Instance variables can be accessed from inside any method, constructor or blocks of that particular class. Class variables Class variables are variables declared within a class, outside any method, with the static keyword. A class can have any number of methods to access the value of various kinds of methods. In the above example, barking(), hungry() and sleeping() are methods.

Following are some of the important topics that need to be discussed when looking into classes of the Java Language.

Constructors When discussing about classes, one of the most important sub topic would be constructors. Every class has a constructor. If we do not explicitly write a constructor for a class, the Java compiler builds a default constructor for that class.

Each time a new object is created, at least one constructor will be invoked. The main rule of constructors is that they should have the same name as the class. A class can have more than one constructor.

Following is an example of a constructor

Example

public class Puppy public Puppy()

public Puppy(String name) // This constructor has one parameter, name. Java also supports Singleton Classes where you would be able to create only one instance of a class.

Note We have two different types of constructors. We are going to discuss constructors in detail in the subsequent chapters.

Creating an Object As mentioned previously, a class provides the blueprints for objects. So basically, an object is created from a class. In Java, the new keyword is used to create new objects.

There are three steps when creating an object from a class

Declaration A variable declaration with a variable name with an object type. Instantiation The 'new' keyword is used to create the object. Initialization The 'new' keyword is followed by a call to a constructor. This call initializes the new object. Following is an example of creating an object

Example

public class Puppy public Puppy(String name) // This constructor has one parameter, name. System.out.println("Passed Name is :" + name);

public static void main(String []args) // Following statement would create an object myPuppy Puppy myPuppy = new Puppy("tommy"); If we compile and run the above program, then it will produce the following result

Passed Name is :tommy Accessing Instance Variables and Methods Instance variables and methods are accessed via created objects. To access an instance variable, following is the fully qualified path

/* First create an object */ ObjectReference = new Constructor();

/* Now call a variable as follows */ ObjectReference.variableName;

/* Now you can call a class method as follows */ ObjectReference.MethodName(); Example

This example explains how to access instance variables and methods of a class.

public class Puppy int puppyAge;

public Puppy(String name) // This constructor has one parameter, name. System.out.println("Name chosen is :" + name);

public void setAge(int age) puppyAge = age;

public int getAge( ) System.out.println("Puppy's age is :" + puppyAge); return puppyAge;

public static void main(String []args) /* Object creation */ Puppy myPuppy = new Puppy("tommy");

/* Call class method to set puppy's age */ myPuppy.setAge(2);

/* Call another class method to get puppy's age */ myPuppy.getAge( );

/* You can access instance variable as follows as well */ System.out.println("Variable Value :" + myPuppy.puppyAge); If we compile and run the above program, then it will produce the following result

Output

Name chosen is :tommy Puppy's age is :2 Variable Value :2 Source File Declaration Rules As the last part of this section, let's now look into the source file declaration rules. These rules are essential when declaring classes, import statements and package statements in a source file.

There can be only one public class per source file. A source file can have multiple non-public classes. The public class name should be the name of the source file as well which should be appended by .java at the end. For example: the class name is public class Employee then the source file should be as Employee.java. If the class is defined inside a package, then the package statement should be the first statement in the source file. If import statements are present, then they must be written between the package statement and the class declaration. If there are no package statements, then the import statement should be the first line in the source file. Import and package statements will imply to all the classes present in the source file. It is not possible to declare different import and/or package statements to different classes in the source file. Classes have several access levels and there are different types of classes; abstract classes, final classes, etc. We will be explaining about all these in the access modifiers chapter.

Apart from the above mentioned types of classes, Java also has some special classes called Inner classes and Anonymous classes.

Java Package In simple words, it is a way of categorizing the classes and interfaces. When developing applications in Java, hundreds of classes and interfaces will be written, therefore categorizing these classes is a must as well as makes life much easier.

Import Statements In Java if a fully qualified name, which includes the package and the class name is given, then the compiler can easily locate the source code or classes. Import statement is a way of giving the proper location for the compiler to find that particular class.

For example, the following line would ask the compiler to load all the classes available in directory java$_installation/java/io$

import java.io.*; A Simple Case Study For our case study, we will be creating two classes. They are Employee and EmployeeTest.

First open notepad and add the following code. Remember this is the Employee class and the class is a public class. Now, save this source file with the name Employee.java.

The Employee class has four instance variables - name, age, designation and salary. The class has one explicitly defined constructor, which takes a parameter.

Example

import java.io.*; public class Employee

String name; int age; String designation; double salary;

// This is the constructor of the class Employee public Employee(String name) this.name = name;

// Assign the age of the Employee to the variable age. public void empAge(int empAge) age = empAge;

/* Assign the designation to the variable designation.*/ public void empDesignation(String empDesig) designation = empDesig;

/* Assign the salary to the variable salary.*/ public void empSalary(double empSalary) salary = empSalary;

/* Print the Employee details */ public void printEmployee() System.out.println("Name:"+ name); System.out.println("Age:" + age); System.out.println("Designation:" + designation); System.out.println("Salary:" + salary); As mentioned previously in this tutorial, processing starts from the main method. Therefore, in order for us to run this Employee class there should be a main method and objects should be created. We will be creating a separate class for these tasks.

Following is the EmployeeTest class, which creates two instances of the class Employee and invokes the methods for each object to assign values for each variable.

Save the following code in EmployeeTest.java file.

import java.io.*; public class EmployeeTest

public static void main(String args[]) /* Create two objects using constructor */ Employee empOne = new Employee("James Smith"); Employee empTwo = new Employee("Mary Anne");

// Invoking methods for each object created empOne.empAge(26); empOne.empDesignation("Senior Software Engineer"); empOne.empSalary(1000); empOne.printEmployee();

empTwo.empAge(21); empTwo.empDesignation("Software Engineer"); empTwo.empSalary(500);

empTwo.printEmployee(); Now, compile both the classes and then run EmployeeTest to see the result as follows

Output

C:$javacEmployee.javaC : javacEmployeeTest.javaC : javaEmployeeTestName : JamesSmithAge : 26Designation : SeniorSoftwareEngineerSalary$: 1000.0$Name : MaryAnneAge$: 21$Designation : SoftwareEngineerSalary$: 500.0

6.4.2 Encapsulation

Encapsulation is one of the four fundamental OOP concepts. The other three are inheritance, polymorphism, and abstraction.

Encapsulation in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes, and can be accessed only through the methods of their current class. Therefore, it is also known as data hiding.

Implementation To achieve encapsulation in Java

Declare the variables of a class as private. Provide public setter and getter methods to modify and view the variables values. Example Following is an example that demonstrates how to achieve Encapsulation in Java

/* File name : EncapTest.java */ public class EncapTest private String name; private String idNum; private int age;

public int getAge() return age;

public String getName() return name;

public String getIdNum() return idNum;

public void setAge(int newAge) age = newAge;

public void setName(String newName) name = newName;

public void setIdNum(String newId) idNum = newId; The public setXXX() and getXXX() methods are the access points of the instance variables of the EncapTest class. Normally, these methods are referred as getters and setters. Therefore, any class that wants to access the variables should access them through these getters and setters.

The variables of the EncapTest class can be accessed using the following program

/* File name : RunEncap.java */ public class RunEncap

public static void main(String args[]) EncapTest encap = new EncapTest(); encap.setName("James"); encap.setAge(20); encap.setIdNum("12343ms");

System.out.print("Name : " + encap.getName() + " Age : " + encap.getAge()); This will produce the following result

Name : James Age : 20 Benefits The fields of a class can be made read-only or write-only. A class can have total control over what is stored in its fields. The users of a class do not know how the class stores its data. A class can change the data type of a field and users of the class do not need to change any of their code. Related Readings "Java Inheritance". www.tutorialspoint.com. N.p., 2016. Web. 10 Dec. 2016.

6.4.3 Inheritance

In the preceding lessons, you have seen inheritance mentioned several times. In the Java language, classes can be derived from other classes, thereby inheriting fields and methods from those classes.

The idea of inheritance is simple but powerful: When you want to create a new class and there is already a class that includes some of the code that you want, you can derive your new class from the existing class. In doing this, you can reuse the fields and methods of the existing class without having to write (and debug!) them yourself.

A subclass inherits all the members (fields, methods, and nested classes) from its superclass. Constructors are not members, so they are not inherited by subclasses, but the constructor of the superclass can be invoked from the subclass.

Class Hierarchy The Object class, defined in the java.lang package, defines and implements behavior common to all classes—including the ones that you write. In the Java platform, many classes derive directly from Object, other classes derive from some of those classes, and so on, forming a hierarchy of classes.

At the top of the hierarchy, Object is the most general of all classes. Classes near the bottom of the hierarchy provide more specialized behavior.

An Example Here is the sample code for a possible implementation of a Bicycle class that was presented in the Classes and Objects lesson:

public class Bicycle

// the Bicycle class has three fields public int cadence; public int gear; public int speed;

// the Bicycle class has one constructor public Bicycle(int startCadence, int startSpeed, int startGear) gear = startGear; cadence = startCadence; speed = startSpeed;

// the Bicycle class has four methods public void setCadence(int newValue) cadence = newValue;

public void setGear(int newValue) gear = newValue;

public void applyBrake(int decrement) speed -= decrement;

public void speedUp(int increment) speed += increment;

A class declaration for a MountainBike class that is a subclass of Bicycle might look like this:

public class MountainBike extends Bicycle

// the MountainBike subclass adds one field public int seatHeight;

// the MountainBike subclass has one constructor public MountainBike(int startHeight, int startCadence, int startSpeed, int startGear) super(startCadence, startSpeed, startGear); seatHeight = startHeight;

// the MountainBike subclass adds one method public void setHeight(int newValue) seatHeight = newValue; MountainBike inherits all the fields and methods of Bicycle and adds the field seatHeight and a method to set it. Except for the constructor, it is as if you had written a new MountainBike class entirely from scratch, with four fields and five methods. However, you didn't have to do all the work. This would be especially valuable if the methods in the Bicycle class were complex and had taken substantial time to debug.

What You Can Do in a Subclass A subclass inherits all of the public and protected members of its parent, no matter what package the subclass is in. If the subclass is in the same package as its parent, it also inherits the package-private members of the parent. You can use the inherited members as is, replace them, hide them, or supplement them with new members:

The inherited fields can be used directly, just like any other fields. You can declare a field in the subclass with the same name as the one in the superclass, thus hiding it (not * recommended). You can declare new fields in the subclass that are not in the superclass. The inherited methods can be used directly as they are. You can write a new instance method in the subclass that has the same signature as the one in the superclass, thus overriding it. You can write a new static method in the subclass that has the same signature as the one in the superclass, thus hiding it. You can declare new methods in the subclass that are not in the superclass. You can write a subclass constructor that invokes the constructor of the superclass, either implicitly or by using the keyword super. The following sections in this lesson will expand on these topics.

Private Members in a Superclass A subclass does not inherit the private members of its parent class. However, if the superclass has public or protected methods for accessing its private fields, these can also be used by the subclass.

A nested class has access to all the private members of its enclosing class—both fields and methods. Therefore, a public or protected nested class inherited by a subclass has indirect access to all of the private members of the superclass.

Casting Objects We have seen that an object is of the data type of the class from which it was instantiated. For example, if we write

public MountainBike myBike = new MountainBike(); then myBike is of type MountainBike.

MountainBike is descended from Bicycle and Object. Therefore, a MountainBike is a Bicycle and is also an Object, and it can be used wherever Bicycle or Object objects are called for.

The reverse is not necessarily true: a Bicycle may be a MountainBike, but it isn't necessarily. Similarly, an Object may be a Bicycle or a MountainBike, but it isn't necessarily.

Casting shows the use of an object of one type in place of another type, among the objects permitted by inheritance and implementations. For example, if we write

Object obj = new MountainBike(); then obj is both an Object and a MountainBike (until such time as obj is assigned another object that is not a MountainBike). This is called implicit casting.

If, on the other hand, we write

MountainBike myBike = obj; we would get a compile-time error because obj is not known to the compiler to be a MountainBike. However, we can tell the compiler that we promise to assign a MountainBike to obj by explicit casting:

MountainBike myBike = (MountainBike)obj; This cast inserts a runtime check that obj is assigned a MountainBike so that the compiler can safely assume that obj is a MountainBike. If obj is not a MountainBike at runtime, an exception will be thrown.

Related Readings "Inheritance". docs.oracle.com. N.p., 2016. Web. 8 Dec. 2016. "Java Inheritance". www.tutorialspoint.com. N.p., 2016. Web. 8 Dec. 2016. Friesen, Jeff. "Java 101: Inheritance In Java, Part 1". JavaWorld. N.p., 2016. Web. 8 Dec. 2016.

6.4.4 Polymorphism

Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object.

Any Java object that can pass more than one IS-A test is considered to be polymorphic. In Java, all Java objects are polymorphic since any object will pass the IS-A test for their own type and for the class Object.

It is important to know that the only possible way to access an object is through a reference variable. A reference variable can be of only one type. Once declared, the type of a reference variable cannot be changed.

The reference variable can be reassigned to other objects provided that it is not declared final. The type of the reference variable would determine the methods that it can invoke on the object.

A reference variable can refer to any object of its declared type or any subtype of its declared type. A reference variable can be declared as a class or interface type.

Example Let us look at an example.

public interface Vegetarian public class Animal public class Deer extends Animal implements Vegetarian Now, the Deer class is considered to be polymorphic since this has multiple inheritance. Following are true for the above examples

A Deer IS-A Animal A Deer IS-A Vegetarian A Deer IS-A Deer A Deer IS-A Object When we apply the reference variable facts to a Deer object reference, the following declarations are legal

Deer d = new Deer(); Animal a = d; Vegetarian v = d; Object o = d; All the reference variables d, a, v, o refer to the same Deer object in the heap.

Virtual Methods In this section, I will show you how the behavior of overridden methods in Java allows you to take advantage of polymorphism when designing your classes.

We already have discussed method overriding, where a child class can override a method in its parent. An overridden method is essentially hidden in the parent class, and is not invoked unless the child class uses the super keyword within the overriding method.

/* File name : Employee.java */ public class Employee private String name; private String address; private int number;

public Employee(String name, String address, int number) System.out.println("Constructing an Employee"); this.name = name; this.address = address; this.number = number;

public void mailCheck() System.out.println("Mailing a check to " + this.name + " " + this.address);

public String toString() return name + " " + address + " " + number;

public String getName() return name;

public String getAddress() return address;

public void setAddress(String newAddress) address = newAddress;

public int getNumber() return number; Now suppose we extend Employee class as follows

/* File name : Salary.java */ public class Salary extends Employee private double salary; // Annual salary

public Salary(String name, String address, int number, double salary) super(name, address, number); setSalary(salary);

public void mailCheck() System.out.println("Within mailCheck of Salary class "); System.out.println("Mailing check to " + getName() + " with salary " + salary);

public double getSalary() return salary;

public void setSalary(double newSalary) if(newSalary >= 0.0) salary = newSalary;

public double computePay() System.out.println("Computing salary pay for " + getName()); return salary/52; Now, you study the following program carefully and try to determine its output

/* File name : VirtualDemo.java */ public class VirtualDemo

public static void main(String [] args) Salary s = new Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00); Employee e = new Salary("John Adams", "Boston, MA", 2, 2400.00); System.out.println("Call mailCheck using Salary reference –"); s.mailCheck(); System.out.println("Call mailCheck using Employee reference–"); e.mailCheck(); This will produce the following result

Constructing an Employee Constructing an Employee

Call mailCheck using Salary reference – Within mailCheck of Salary class Mailing check to Mohd Mohtashim with salary 3600.0

Call mailCheck using Employee reference– Within mailCheck of Salary class Mailing check to John Adams with salary 2400.0 Here, we instantiate two Salary objects. One using a Salary reference s, and the other using an Employee reference e.

While invoking s.mailCheck(), the compiler sees mailCheck() in the Salary class at compile time, and the JVM invokes mailCheck() in the Salary class at run time.

mailCheck() on e is quite different because e is an Employee reference. When the compiler sees e.mailCheck(), the compiler sees the mailCheck() method in the Employee class.

Here, at compile time, the compiler used mailCheck() in Employee to validate this statement. At run time, however, the JVM invokes mailCheck() in the Salary class.

This behavior is referred to as virtual method invocation, and these methods are referred to as virtual methods. An overridden method is invoked at run time, no matter what data type the reference is that was used in the source code at compile time.

Related Readings "Java Polymorphism". www.tutorialspoint.com. N.p., 2016. Web. 10 Dec. 2016.

6.4.5 Abstraction

As per dictionary, abstraction is the quality of dealing with ideas rather than events. For example, when you consider the case of e-mail, complex details such as what happens as soon as you send an e-mail, the protocol your e-mail server uses are hidden from the user. Therefore, to send an e-mail you just need to type the content, mention the address of the receiver, and click send.

Likewise in Object-oriented programming, abstraction is a process of hiding the implementation details from the user, only the functionality will be provided to the user. In other words, the user will have the information on what the object does instead of how it does it.

In Java, abstraction is achieved using Abstract classes and interfaces.

Abstract Class A class which contains the abstract keyword in its declaration is known as abstract class.

Abstract classes may or may not contain abstract methods, i.e., methods without body (public void get();) But, if a class has at least one abstract method, then the class must be declared abstract. If a class is declared abstract, it cannot be instantiated. To use an abstract class, you have to inherit it from another class, provide implementations to the abstract methods in it. If you inherit an abstract class, you have to provide implementations to all the abstract methods in it. Example

This section provides you an example of the abstract class. To create an abstract class, just use the abstract keyword before the class keyword, in the class declaration.

/* File name : Employee.java */ public abstract class Employee private String name; private String address; private int number;

public Employee(String name, String address, int number) System.out.println("Constructing an Employee"); this.name = name; this.address = address; this.number = number;

public double computePay() System.out.println("Inside Employee computePay"); return 0.0;

public void mailCheck() System.out.println("Mailing a check to " + this.name + " " + this.address);

public String toString() return name + " " + address + " " + number;

public String getName() return name;

public String getAddress() return address;

public void setAddress(String newAddress) address = newAddress;

public int getNumber() return number; You can observe that except abstract methods the Employee class is same as normal class in Java. The class is now abstract, but it still has three fields, seven methods, and one constructor.

Now you can try to instantiate the Employee class in the following way

/* File name : AbstractDemo.java */ public class AbstractDemo

public static void main(String [] args) /* Following is not allowed and would raise error */ Employee e = new Employee("George W.", "Houston, TX", 43); System.out.println("Call mailCheck using Employee reference–"); e.mailCheck(); When you compile the above class, it gives you the following error

Employee.java:46: Employee is abstract; cannot be instantiated Employee e = new Employee("George W.", "Houston, TX", 43); $^1 errorInheritingtheAbstractClassWecaninherittheprop$

/* File name : Salary.java */ public class Salary extends Employee private double salary; // Annual salary

public Salary(String name, String address, int number, double salary) super(name, address, number); setSalary(salary);

public void mailCheck() System.out.println("Within mailCheck of Salary class "); System.out.println("Mailing check to " + getName() + " with salary " + salary);

public double getSalary() return salary;

public void setSalary(double newSalary) if(newSalary $>=$ 0.0) salary = newSalary;

public double computePay() System.out.println("Computing salary pay for " + getName()); return salary/52; Here, you cannot instantiate the Employee class, but you can instantiate the Salary Class, and using this instance you can access all the three fields and seven methods of Employee class as shown below.

/* File name : AbstractDemo.java */ public class AbstractDemo public static void main(String [] args) Salary s = new Salary("Mohd Mohtashim", "Ambehta, UP", 3, 3600.00); Employee e = new Salary("John Adams", "Boston, MA", 2, 2400.00); System.out.println("Call mailCheck using Salary reference –"); s.mailCheck(); System.out.println("mailCheck using Employee reference–"); e.mailCheck(); This produces the following result

Constructing an Employee Constructing an Employee Call mailCheck using Salary reference – Within mailCheck of Salary class Mailing check to Mohd Mohtashim with salary 3600.0

Call mailCheck using Employee reference– Within mailCheck of Salary class Mailing check to John Adams with salary 2400.0 Abstract Methods If you want a class to contain a particular method but you want the actual implementation of that method to be determined by child classes, you can declare the method in the parent class as an abstract.

abstract keyword is used to declare the method as abstract. You have to place the abstract keyword before the method name in the method declaration. An abstract method contains a method signature, but no method body. Instead of curly braces, an abstract method will have a semoi colon (;) at the end. Following is an example of the abstract method.

public abstract class Employee private String name; private String address; private int number;

public abstract double computePay(); // Remainder of class definition Declaring a method as abstract has two consequences

The class containing it must be declared as abstract. Any class inheriting the current class must either override the abstract method or declare itself as abstract. Note Eventually, a descendant class has to implement the abstract method; otherwise, you would have a hierarchy of abstract classes that cannot be instantiated.

Suppose Salary class inherits the Employee class, then it should implement the computePay() method as shown below

/* File name : Salary.java */ public class Salary extends Employee private double salary; // Annual salary

public double computePay() System.out.println("Computing salary pay for " + getName()); return salary/52; // Remainder of class definition Related Readings "Java Abstraction". www.tutorialspoint.com. N.p., 2016. Web. 10 Dec. 2016.

6.5 File System IO

The java.io package contains nearly every class you might ever need to perform input and output (I/O) in Java. All these streams represent an input source and an output destination. The stream in the java.io package supports many data such as primitives, object, localized characters, etc.

Stream A stream can be defined as a sequence of data. There are two kinds of Streams

InPutStream The InputStream is used to read data from a source. OutPutStream The OutputStream is used for writing data to a destination.

Java provides strong but flexible support for I/O related to files and networks but this tutorial covers very basic functionality related to streams and I/O. We will see the most commonly used examples one by one

Byte Streams Java byte streams are used to perform input and output of 8-bit bytes. Though there are many classes related to byte streams but the most frequently used classes are, FileInputStream and FileOutputStream. Following is an example which makes use of these two classes to copy an input file into an output file

Example

import java.io.*; public class CopyFile

public static void main(String args[]) throws IOException FileInputStream in = null; FileOutputStream out = null;

try in = new FileInputStream("input.txt"); out = new FileOutputStream("output.txt");

int c; while ((c = in.read()) != -1) out.write(c); finally if (in != null) in.close(); if (out != null) out.close(); Now let's have a file input.txt with the following content

This is test for copy file. As a next step, compile the above program and execute it, which will result in creating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following

*javacCopyFile.java*java CopyFile Character Streams Java Byte streams are used to perform input and output of 8-bit bytes, whereas Java Character streams are used to perform input and output for 16-bit unicode. Though there are many classes related to character streams but the most frequently used classes are, FileReader and FileWriter. Though internally FileReader uses FileInputStream and FileWriter uses FileOutputStream but here the major difference is that FileReader reads two bytes at a time and FileWriter writes two bytes at a time.

We can re-write the above example, which makes the use of these two classes to copy an input file (having unicode characters) into an output file

Example

import java.io.*; public class CopyFile

public static void main(String args[]) throws IOException FileReader in = null; FileWriter out = null;

try in = new FileReader("input.txt"); out = new FileWriter("output.txt");

int c; while ((c = in.read()) != -1) out.write(c); finally if (in != null) in.close(); if (out != null) out.close(); Now let's have a file input.txt with the following content

This is test for copy file. As a next step, compile the above program and execute it, which will result in creating output.txt file with the same content as we have in input.txt. So let's put the above code in CopyFile.java file and do the following

*javacCopyFile.java*java CopyFile Standard Streams All the programming languages provide support for standard I/O where the user's program can take input from a keyboard and then produce an output on the computer screen. If you are aware of C or C++ programming languages, then you must be aware of three standard devices STDIN, STDOUT and STDERR. Similarly, Java provides the following three standard streams

Standard Input This is used to feed the data to user's program and usually a keyboard is used as standard input stream and represented as System.in. Standard Output This is used to output the data produced by the user's program and usually a computer screen is used for standard output stream and represented as System.out. Standard Error This is used to output the error data produced by the user's program and usually a computer screen is used for standard error stream and represented as System.err. Following is a simple program, which creates InputStreamReader to read standard input stream until the user types a "q"

Example

import java.io.*; public class ReadConsole

public static void main(String args[]) throws IOException InputStreamReader cin = null;

try cin = new InputStreamReader(System.in); System.out.println("Enter characters, 'q' to quit."); char c; do c = (char) cin.read(); System.out.print(c); while(c != 'q'); finally if (cin != null) cin.close(); Let's keep the above code in ReadConsole.java file and try to compile and execute it as shown in the following program. This program continues to read and output the same character until we press 'q'

$javacReadConsole.java$java ReadConsole Enter characters, 'q' to quit. 1 1 e e q q Reading and Writing Files As described earlier, a stream can be defined as a sequence of data. The InputStream is used to read data from a source and the OutputStream is used for writing data to a destination.

Here is a hierarchy of classes to deal with Input and Output streams.

The two important streams are FileInputStream and FileOutputStream, which would be discussed in this tutorial.

FileInputStream This stream is used for reading data from the files. Objects can be created using the keyword new and there are several types of constructors available.

Following constructor takes a file name as a string to create an input stream object to read the file

InputStream f = new FileInputStream("C:/java/hello"); Following constructor takes a file object to create an input stream object to read the file. First we create a file object using File() method as follows

File f = new File("C:/java/hello"); InputStream f = new FileInputStream(f); Once you have InputStream object in hand, then there is a list of helper methods which can be used to read to stream or to do other operations on the stream.

Method Description 1 public void close() throws IOException

This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.

2 protected void finalize()throws IOException

This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.

3 public int read(int r)throws IOException

This method reads the specified byte of data from the InputStream. Returns an int. Returns the next byte of data and -1 will be returned if it's the end of the file.

4 public int read(byte[] r) throws IOException

This method reads r.length bytes from the input stream into an array. Returns the total number of bytes read. If it is the end of the file, -1 will be returned.

5 public int available() throws IOException

Gives the number of bytes that can be read from this file input stream. Returns an int.

There are other important input streams available, for more detail you can refer to the following links

ByteArrayInputStream DataInputStream FileOutputStream FileOutputStream is used to create a file and write data into it. The stream would create a file, if it doesn't already exist, before opening it for output.

Here are two constructors which can be used to create a FileOutputStream object.

Following constructor takes a file name as a string to create an input stream object to write the file

OutputStream f = new FileOutputStream("C:/java/hello") Following constructor takes a file object to create an output stream object to write the file. First, we create a file object using File() method as follows

File f = new File("C:/java/hello"); OutputStream f = new FileOutputStream(f); Once you have OutputStream object in hand, then there is a list of helper methods, which can be used to write to stream or to do other operations on the stream.

Method Description 1 public void close() throws IOException

This method closes the file output stream. Releases any system resources associated with the file. Throws an IOException.

2 protected void finalize()throws IOException

This method cleans up the connection to the file. Ensures that the close method of this file output stream is called when there are no more references to this stream. Throws an IOException.

3 public void write(int w)throws IOException

This methods writes the specified byte to the output stream.

4 public void write(byte[] w)

Writes w.length bytes from the mentioned byte array to the OutputStream.

There are other important output streams available, for more detail you can refer to the following links

ByteArrayOutputStream DataOutputStream Example

Following is the example to demonstrate InputStream and OutputStream

import java.io.*; public class fileStreamTest

public static void main(String args[])

try byte bWrite [] = 11,21,3,40,5; OutputStream os = new FileOutputStream("test.txt"); for(int x = 0; x < bWrite.length ; x++) os.write(bWrite[x]); // writes the bytes os.close();

InputStream is = new FileInputStream("test.txt"); int size = is.available();

for(int i = 0; i < size; i++) System.out.print((char)is.read() + " "); is.close(); catch(IOException e) System.out.print("Exception"); The above code would create file test.txt and would write given numbers in binary format. Same would be the output on the stdout screen.

File Navigation and I/O There are several other classes that we would be going through to get to know the basics of File Navigation and I/O.

File Class FileReader Class FileWriter Class Directories in Java A directory is a File which can contain a list of other files and directories. You use File object to create directories, to list down files available in a directory. For complete detail, check a list of all the methods which you can call on File object and what are related to directories.

Creating Directories There are two useful File utility methods, which can be used to create directories

The mkdir( ) method creates a directory, returning true on success and false on failure. Failure indicates that the path specified in the File object already exists, or that the directory cannot be created because the entire path does not exist yet.

The mkdirs() method creates both a directory and all the parents of the directory.

Following example creates "/tmp/user/java/bin" directory

Example

import java.io.File; public class CreateDir

public static void main(String args[]) String dirname = "/tmp/user/java/bin"; File d = new File(dirname);

// Create directory now. d.mkdirs(); Compile and execute the above code to create "/tmp/user/java/bin".

Note Java automatically takes care of path separators on UNIX and Windows as per conventions. If you use a forward slash (/) on a Windows version of Java, the path will still resolve correctly.

Listing Directories You can use list( ) method provided by File object to list down all the files and directories available in a directory as follows

Example

import java.io.File; public class ReadDir

public static void main(String[] args) File file = null; String[] paths;

try // create new file object file = new File("/tmp");

// array of files and directory paths = file.list();

// for each name in the path array for(String path:paths) // prints filename and directory name System.out.println(path); catch(Exception e) // if any error occurs e.printStackTrace(); This will produce the following result based on the directories and files available in your /tmp directory

test1.txt test2.txt ReadDir.java ReadDir.class Related Readings "Java Files And I/O". www.tutorialspoint.com. N.p., 2016. Web. 15 Dec. 2016.

6.6 Error Handling

An exception (or exceptional event) is a problem that arises during the execution of a program. When an Exception occurs the normal flow of the program is disrupted and the program/Application terminates abnormally, which is not recommended, therefore, these exceptions are to be handled.

An exception can occur for many different reasons. Following are some scenarios where an exception occurs.

A user has entered an invalid data. A file that needs to be opened cannot be found. A network connection has been lost in the middle of communications or the JVM has run out of memory. Some of these exceptions are caused by user error, others by programmer error, and others by physical resources that have failed in some manner.

Based on these, we have three categories of Exceptions. You need to understand them to know how exception handling works in Java.

Type of exceptions Checked Exception

A checked exception is an exception that occurs at the compile time, these are also called as compile time exceptions. These exceptions cannot simply be ignored at the time of compilation, the programmer should take care of (handle) these exceptions.

For example, if you use FileReader class in your program to read data from a file, if the file specified in its constructor doesn't exist, then a FileNotFoundException occurs, and the compiler prompts the programmer to handle the exception.

import java.io.File; import java.io.FileReader;

public class FilenotFound$_{D}emo$

public static void main(String args[]) File file = new File("E://file.txt"); FileReader fr = new FileReader(file); If you try to compile the above program, you will get the following exceptions.

C: $javacFilenotFound_{D}emo.javaFilenotFound_{D}emo.java : 8 : error : unreportedexceptionFileNotFoundException; mustbecaughtordeclaredtobethrownFileReaderfr = newFileReader(file);^{1} errorNoteSincethemethodsread()andclose()ofFileReaderclassthrowsIOException$

Unchecked exceptions

An unchecked exception is an exception that occurs at the time of execution. These are also called as Runtime Exceptions. These include programming bugs, such as logic errors or improper use of an API. Runtime exceptions are ignored at the time of compilation.

For example, if you have declared an array of size 5 in your program, and trying to call the 6th element of the array then an ArrayIndexOutOfBoundsExceptionexception occurs.

public class Unchecked$_{D}emo$

public static void main(String args[]) int num[] = 1, 2, 3, 4; System.out.println(num[5]); If you compile and execute the above program, you will get the following exception.

Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5 at Exceptions.Unchecked$_{D}emo.main(Unchecked_{D}emo.java : 8)Errors$

These are not exceptions at all, but problems that arise beyond the control of the user or the programmer. Errors are typically ignored in your code because you can rarely do anything about an error. For example, if a stack overflow occurs, an error will arise. They are also ignored at the time of compilation.

Exception Hierarchy All exception classes are subtypes of the java.lang.Exception class. The exception class is a subclass of the Throwable class. Other than the exception class there is another subclass called Error which is derived from the Throwable class.

Errors are abnormal conditions that happen in case of severe failures, these are not handled by the Java programs. Errors are generated to indicate errors generated by the runtime environment. Example: JVM is out of memory. Normally, programs cannot recover from errors.

The Exception class has two main subclasses: IOException class and RuntimeException Class.

Following is a list of most common checked and unchecked Java's Built-in Exceptions

Exceptions Methods Following is the list of important methods available in the Throwable class.

1 public String getMessage() Returns a detailed message about the exception that has occurred. This message is initialized in the Throwable constructor. 2 public Throwable getCause() Returns the cause of the exception as represented by a Throwable object. 3 public String toString() Returns the name of the class

concatenated with the result of getMessage(). 4 public void printStackTrace() Prints the result of toString() along with the stack trace to System.err, the error output stream. 5 public StackTraceElement [] getStackTrace() Returns an array containing each element on the stack trace. The element at index 0 represents the top of the call stack, and the last element in the array represents the method at the bottom of the call stack. 6 public Throwable fillInStackTrace() Fills the stack trace of this Throwable object with the current stack trace, adding to any previous information in the stack trace. Catching Exceptions A method catches an exception using a combination of the try and catch keywords. A try/catch block is placed around the code that might generate an exception. Code within a try/catch block is referred to as protected code, and the syntax for using try/catch looks like the following

Syntax

try // Protected code catch(ExceptionName e1) // Catch block The code which is prone to exceptions is placed in the try block. When an exception occurs, that exception occurred is handled by catch block associated with it. Every try block should be immediately followed either by a catch block or finally block.

A catch statement involves declaring the type of exception you are trying to catch. If an exception occurs in protected code, the catch block (or blocks) that follows the try is checked. If the type of exception that occurred is listed in a catch block, the exception is passed to the catch block much as an argument is passed into a method parameter.

Example

The following is an array declared with 2 elements. Then the code tries to access the 3rd element of the array which throws an exception.

// File Name : ExcepTest.java import java.io.*;

public class ExcepTest

public static void main(String args[]) try int a[] = new int[2]; System.out.println("Access element three :" + a[3]); catch(ArrayIndexOutOfBoundsException e) System.out.println("Exception thrown :" + e); System.out.println("Out of the block"); This will produce the following result

Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 3 Out of the block Multiple Catch Blocks A try block can be followed by multiple catch blocks. The syntax for multiple catch blocks looks like the following

try // Protected code catch(ExceptionType1 e1) // Catch block catch(ExceptionType2 e2) // Catch block catch(ExceptionType3 e3) // Catch block The previous statements demonstrate three catch blocks, but you can have any number of them after a single try. If an exception occurs in the protected code, the exception is thrown to the first catch block in the list. If the data type of the exception thrown matches ExceptionType1, it gets caught there. If not, the exception passes down to the second catch statement. This continues until the exception either is caught or falls through all catches, in which case the current method stops execution and the exception is thrown down to the previous method on the call stack.

Example

Here is code segment showing how to use multiple try/catch statements.

try file = new FileInputStream(fileName); x = (byte) file.read(); catch(IOException i) i.printStackTrace(); return -1; catch(FileNotFoundException f) // Not valid! f.printStackTrace(); return -1; Catching Multiple Type of Exceptions Since

Java 7, you can handle more than one exception using a single catch block, this feature simplifies the code. Here is how you would do it

catch (IOException|FileNotFoundException ex) logger.log(ex); throw ex; The Throws/Throw Keywords If a method does not handle a checked exception, the method must declare it using the throws keyword. The throws keyword appears at the end of a method's signature.

You can throw an exception, either a newly instantiated one or an exception that you just caught, by using the throw keyword.

Try to understand the difference between throws and throw keywords, throws is used to postpone the handling of a checked exception and throw is used to invoke an exception explicitly.

The following method declares that it throws a RemoteException

import java.io.*; public class className

public void deposit(double amount) throws RemoteException // Method implementation throw new RemoteException(); // Remainder of class definition A method can declare that it throws more than one exception, in which case the exceptions are declared in a list separated by commas. For example, the following method declares that it throws a RemoteException and an InsufficientFundsException

import java.io.*; public class className

public void withdraw(double amount) throws RemoteException, InsufficientFundsException // Method implementation // Remainder of class definition The Finally Block The finally block follows a try block or a catch block. A finally block of code always executes, irrespective of occurrence of an Exception.

Using a finally block allows you to run any cleanup-type statements that you want to execute, no matter what happens in the protected code.

A finally block appears at the end of the catch blocks and has the following syntax

Syntax

try // Protected code catch(ExceptionType1 e1) // Catch block catch(ExceptionType2 e2) // Catch block catch(ExceptionType3 e3) // Catch block finally // The finally block always executes.

Example

public class ExcepTest

public static void main(String args[]) int a[] = new int[2]; try System.out.println("Access element three :" + a[3]); catch(ArrayIndexOutOfBoundsException e) System.out.println("Exception thrown :" + e); finally a[0] = 6; System.out.println("First element value: " + a[0]); System.out.println("The finally statement is executed"); This will produce the following result

Exception thrown :java.lang.ArrayIndexOutOfBoundsException: 3 First element value: 6 The finally statement is executed Note the following

A catch clause cannot exist without a try statement. It is not compulsory to have finally clauses whenever a try/catch block is present. The try block cannot be present without either catch clause or finally clause. Any code cannot be present in between the try, catch, finally blocks. The try-with-resources Generally, when we use any resources like streams, connections, etc. we have to close them explicitly using finally block. In the following program, we are reading data from a file using FileReader and we are closing it using finally block.

import java.io.File; import java.io.FileReader; import java.io.IOException;

public class ReadData$_{D}emo$

public static void main(String args[]) FileReader fr = null; try File file = new File("file.txt"); fr = new FileReader(file); char [] a = new char[50]; fr.read(a); // reads the content to the array for(char c : a) System.out.print(c); // prints the characters one by one catch(IOException e) e.printStackTrace(); finally try fr.close(); catch(IOException ex) ex.printStackTrace(); try-with-resources, also referred as automatic resource management, is a new exception handling mechanism that was introduced in Java 7, which automatically closes the resources used within the try catch block.

To use this statement, you simply need to declare the required resources within the parenthesis, and the created resource will be closed automatically at the end of the block. Following is the syntax of try-with-resources statement.

Syntax

try(FileReader fr = new FileReader("file path")) // use the resource catch() // body of catch Following is the program that reads the data in a file using try-with-resources statement.

Example

import java.io.FileReader; import java.io.IOException;

public class Try$_withDemo$

public static void main(String args[]) try(FileReader fr = new FileReader("E://file.txt")) char [] a = new char[50]; fr.read(a); // reads the contentto the array for(char c : a) System.out.print(c); // prints the characters one by one catch(IOException e) e.printStackTrace(); Following points are to be kept in mind while working with try-with-resources statement.

To use a class with try-with-resources statement it should implement AutoCloseable interface and the close() method of it gets invoked automatically at runtime. You can declare more than one class in try-with-resources statement. While you declare multiple classes in the try block of try-with-resources statement these classes are closed in reverse order. Except the declaration of resources within the parenthesis everything is the same as normal try/catch block of a try block. The resource declared in try gets instantiated just before the start of the try-block. The resource declared at the try block is implicitly declared as final. User-defined Exceptions You can create your own exceptions in Java. Keep the following points in mind when writing your own exception classes

All exceptions must be a child of Throwable. If you want to write a checked exception that is automatically enforced by the Handle or Declare Rule, you need to extend the Exception class. If you want to write a runtime exception, you need to extend the RuntimeException class. We can define our own Exception class as below

class MyException extends Exception You just need to extend the predefined Exception class to create your own Exception. These are considered to be checked exceptions. The following InsufficientFundsException class is a user-defined exception that extends the Exception class, making it a checked exception. An exception class is like any other class, containing useful fields and methods.

Example

// File Name InsufficientFundsException.java import java.io.*;

public class InsufficientFundsException extends Exception private double amount;

public InsufficientFundsException(double amount) this.amount = amount;

public double getAmount() return amount; To demonstrate using our user-defined exception, the following CheckingAccount class contains a withdraw() method that throws an InsufficientFundsException.

// File Name CheckingAccount.java import java.io.*;

public class CheckingAccount private double balance; private int number;

public CheckingAccount(int number) this.number = number;

public void deposit(double amount) balance += amount;

public void withdraw(double amount) throws InsufficientFundsException if(amount <= balance) balance -= amount; else double needs = amount - balance; throw new InsufficientFundsException(needs);

public double getBalance() return balance;

public int getNumber() return number; The following BankDemo program demonstrates invoking the deposit() and withdraw() methods of CheckingAccount.

// File Name BankDemo.java public class BankDemo

public static void main(String [] args) CheckingAccount c = new CheckingAccount(101); System.out.println("Depositing 500..."); $c.deposit(500.00);$

try System.out.println("100..."); $c.withdraw(100.00); System.out.println("600...");$ c.withdraw(600.00); catch(InsufficientFundsException e) System.out.println("Sorry, but you are short "$+e.getAmount()); e.printStackTrace(); CompileallTheabovethreefilesandrunBankDemo$

Output

Depositing 500...

Withdrawing 100...

Withdrawing 600...$Sorry, butyouareshort$200.0 InsufficientFundsException at CheckingAccount.withdraw(CheckingAccount.java:25) at BankDemo.main(BankDemo.java:13) Common Exceptions In Java, it is possible to define two catergories of Exceptions and Errors.

JVM Exceptions These are exceptions/errors that are exclusively or logically thrown by the JVM. Examples: NullPointerException, ArrayIndexOutOfBoundsException, ClassCastException. Programmatic Exceptions These exceptions are thrown explicitly by the application or the API programmers. Examples: IllegalArgumentException, IllegalStateException. Suggested Readings "Java Exceptions". 2016. www.Tutorialspoint.Com. https://www.tutorialspoint.com/java/java$_e xceptions.htm.$

6.7 Logging

Log4j log4j is a reliable, fast and flexible logging framework (APIs) written in Java, which is distributed under the Apache Software License. log4j is a popular logging package written in Java. log4j has been ported to the C, C++, C, Perl, Python, Ruby, and Eiffel languages.

log4j is highly configurable through external configuration files at runtime. It views the logging process in terms of levels of priorities and offers mechanisms to direct logging information to a great variety of destinations, such as a database, file, console, UNIX Syslog, etc.

log4j has three main components:

loggers: Responsible for capturing logging information. appenders: Responsible for publishing logging information to various preferred destinations. layouts: Responsible for formatting logging information in different styles. log4j features

It is thread-safe. It is optimized for speed. It is based on a named logger hierarchy. It supports multiple output appenders per logger. It supports internationalization. It is not restricted to a predefined set of facilities. Logging behavior can be set at runtime using a configuration file. It is designed to handle Java Exceptions from the start. It uses multiple levels, namely ALL, TRACE, DEBUG, INFO, WARN, ERROR and FATAL. The format of the log output can be easily changed by extending the Layout class. The target of the log output as well as the writing strategy can be altered by implementations of the Appender interface. It is fail-stop. However, although it certainly strives to ensure delivery, log4j does not guarantee that each log statement will be delivered to its destination. Example Step 1: Add log4j dependency to your build.gradle file

compile group: 'log4j', name: 'log4j', version: '1.2.17' Step 2: Add log configuration in main/resources/log4j.property

Set root logger level to DEBUG and its only appender to A1. log4j.rootLogger=DEBUG, A1

A1 is set to be a ConsoleAppender. log4j.appender.A1=org.apache.log4j.ConsoleAppender

A1 uses PatternLayout. log4j.appender.A1.layout=org.apache.log4j.PatternLayout log4j.appender.A1.layout.ConversionPattern=

Print only messages of level WARN or above in the package com.foo. log4j.logger.com.foo=WARN Here is another configuration file that uses multiple appenders:

log4j.rootLogger=debug, stdout, R

log4j.appender.stdout=org.apache.log4j.ConsoleAppender log4j.appender.stdout.layout=org.apache.log4j.

Pattern to output the caller's file name and line number. log4j.appender.stdout.layout.ConversionPattern=

log4j.appender.R=org.apache.log4j.RollingFileAppender log4j.appender.R.File=example.log

log4j.appender.R.MaxFileSize=100KB Keep one backup file log4j.appender.R.MaxBackupIndex=1

log4j.appender.R.layout=org.apache.log4j.PatternLayout log4j.appender.R.layout.ConversionPattern=Ste 3: Sample log4j program

package logging;

import org.apache.log4j.Logger;

public class LoggingDemo public static void main(String[] args) final Logger logger = Logger.getLogger(LoggingDemo.class); logger.debug("debug statement"); logger.info("info statement"); logger.error("error statement"); Output

DEBUG [main] (LoggingDemo.java:10) - debug statement INFO [main] (LoggingDemo.java:11) - info statement ERROR [main] (LoggingDemo.java:12) - error statement Suggested Readings "Log4j Tutorial". 2016. www.tutorialspoint.com. http://www.tutorialspoint.com/log4j/. "Java Logging". 2016. tutorials.jenkov.com. http://tutorials.jenkov.com/java-logging/index.html.

6.8 IDE

Java: IDE IntellIJ 1. Project Manager 2. Search Replace 3. Navigation 4. Formatting 5. Debugging 6. Build Release 7. Git Integration 1. Project Manager 1.1 Create New Project

1.2 Import Maven Project

https://www.jetbrains.com/help/idea/2016.1/importing-project-from-maven-model.html

2. Search Replace Global Search Shift Shift 3. Navigation Next/Previous Error F2 / Shift + F2 4. Formatting Auto Format Ctrl + Alt + L

6.9 Package Manager

Java: Package Manager Gradle

Create your first project with gradle Step 1: Create new project folder

mkdir gradle$_{s}ampleStep2 : Makefolderstructure$

gradle init –type java-library Step 3: Import to IntelliJ

Open IntelliJ, click File > New... > Project From Existing Sources... Plugins Application plugin Usages

1. Using the application plugin

Add this line in build.gradle

apply plugin: 'application' 2. Configure the application main class

mainClassName = "org.gradle.sample.Main"

6.10 Build Tool

Java: Build Tool Apache Ant

Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. Ant can also be used effectively to build non Java applications, for instance C or C++ applications. More generally, Ant can be used to pilot any type of process which can be described in terms of targets and tasks. 1

Install Ant Download and extract Apache Ant 1.9.6

wget http://mirrors.viethosting.vn/apache//ant/binaries/apache-ant-1.9.6-bin.tar.gz tar -xzf apache-ant-1.9.6-bin.tar.gz Set path to ant folder

Build Ant through proxy Requirement: 1.9.5+

Add the following lines into build.xml

<target name="ivy-init" depends="ivy-proxy, ivy-probe-antlib, ivy-init-antlib" description="–> initialise Ivy settings"> <ivy:settings file="$ivy.dir/ivysettings.xml" / >< /target >< targetname = "ivy-proxy" description = "-- > ProxyIvysettings" >< propertyname = "proxy.host" value = "proxy.com" / >< propertyname = "proxy.port" value = "8080" / >< propertyname = "proxy.user" value = "user" / >< propertyname = "proxy.password" value = "password" / >< setproxyproxyhost =$ "proxy.host" proxyport="$proxy.port" proxyuser =$ "proxy.user" proxypassword="$proxy.password" / >< /target > ApacheAnt^{TM}$

6.11 Production

Java: Production (Docker) Production with java

Base Image: [java]/java

Docker Folder

your-app/ app bin your$_{a}pp.shlibDockerfilerun.shDockerfile$

FROM java:7

COPY run.sh run.sh run.sh

cd /app/bin chmod u+x your$_{a}pp.sh./your_{a}pp.shCompose$

service: build: ./your$_{a}ppcommand :' bashrun.sh'$

Chương 7

PHP

PHP là ngôn ngữ lập trình web dominate tất cả các anh tài khác mà (chắc là) chỉ dịu đi khi mô hình REST xuất hiện. Nhớ lần đầu gặp bạn Laravel mà cảm giác cuộc đời sang trang.

Cuối tuần này lại phải xem làm sao cài được xdebug vào PHPStorm cho thằng em tập tành lập trình. Haizzz

Tương tác với cơ sở dữ liệu

Liệt kê danh sách các bản ghi trong bảng groups

‘‘‘ $sql = "SELECT * FROM`groups`"$;groups = mysqli$_query$(conn, sql); ‘‘‘

Xóa một bản ghi trong bảng groups

‘‘‘ $sql = "DELETEFROM`groups`WHEREid = `5`"; mysqli_query$(conn, sql); ‘‘‘

Cài đặt debug trong PHPStorm

https://www.youtube.com/watch?v=mEJ21RB0F14

(1) XAMPP

- Download XAMPP (cho PHP 7.1.x - do XDebug chưa chính thức hỗ trợ 7.2.0) https://www.apachefriends.org/xampp-files/7.1.12/xampp-win32-7.1.12-0-VC14-installer.exe - Install XAMPP xampp-win32-7.1.12-0-VC14-installer.exe - Truy cập vào địa chỉ http://localhost/dashboard/phpinfo.php để kiểm tra cài đặt đã thành công chưa

(2) Tải và cài đặt PHPStorm

- Download PHPStorm https://download-cf.jetbrains.com/webide/PhpStorm-2017.3.2.exe - Install PHPStorm

(3) Tạo một web project trong PHPStorm - Chọn interpreter trỏ đến PHP trong xampp

(4) Viết một chương trình add.php

‘‘‘php $a = 2;$b = 3; $c =$a + b;

echo c; ‘‘‘

Click vào ‘add.php‘, chọn Debug, PHPStorm sẽ báo chưa cài XDebug

(5) Cài đặt XDebug theo hướng dẫn tại https://gist.github.com/odan/1abe76d373a9cbb15bed

Click vào add.php, chọn Debug

(6) Cài đặt XDebug với PHPStorm Marklets Vào trang https://www.jetbrains.com/phpstorm/marklets/ Trong phần Zend Debugger - chọn cổng 9000 - IP: 127.0.0.1 Nhấn nút Generate

Bookmark các link quot;Start debuggerquot;, quot;Stop debuggerquot; lên trình duyệt

(7) Debug PHP từ trình duyệt

* Vào trang http://localhost/untitled/add.php * Click vào bookmark Start debugger * Trong PHPStorm, nhấn vào biểu tượng quot;Start Listening for PHP Debug Connectionsquot; * Đặt breakpoint tại dòng thứ 5 * Refresh lại trang http://localhost/untitled/add.php, lúc này, breakpoint sẽ dừng ở dòng 5

Chương 8

R

View online http://magizbox.com/training/r/site/

R R is a programming language and software environment for statistical computing and graphics supported by the R Foundation for Statistical Computing. The R language is widely used among statisticians and data miners for developing statistical software and data analysis. Polls, surveys of data miners, and studies of scholarly literature databases show that R's popularity has increased substantially in recent years.

R is a GNU package. The source code for the R software environment is written primarily in C, Fortran, and R. R is freely available under the GNU General Public License, and pre-compiled binary versions are provided for various operating systems. While R has a command line interface, there are several graphical front-ends available.[

R was created by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, and is currently developed by the R Development Core Team, of which Chambers is a member. R is named partly after the first names of the first two R authors and partly as a play on the name of S. The project was conceived in 1992, with an initial version released in 1995 and a stable beta version in 2000.

8.1 R Courses

I'm going to give a course about R, but it's take a lot of time to finish. I will give at least one lesson a week. You can track it here

(next) Data visualization with R Everything you need to know about R Read and Write Data Importing data from JSON into R Manipulate Data Manipulate String and Datetime Actually, beside my works, there are a lot of excellent and free courses in the internet for you

Beginner

tryr from codeschool

tryr is a course for beginners created by codeschool. This course contains R Syntax, Vectors, Matrices, Summary Statistics, Factors, Data Frames and Working With Real-World Data sections.

Introduction to R from datacamp

This course created by datacamp - a "online learning platform that focuses on building the best learning experience for Data Science in specific". Here is the introduction about this course quoted from authors "In this introduction to R, you will master the basics of this beautiful open source language such as factors, lists and data frames. With the knowledge gained in this course, you will be ready to undertake your first very own data analysis." It contains 6 chapters: Intro to basics, Vectors, Matrices, Factors, Data frames and Lists.

Intermediate and Advanced

R Programming of Johns Hopkins University in coursera Learn how to program in R and how to use R for effective data analysis. This is the second course in the Johns Hopkins Data Science Specialization. It's a 4-weeks course, contains: Overview of R, R data types and objects, reading and writing data (week 1), Control structures, functions, scoping rules, dates and times (week 2), Loop functions, debugging tools (week 3) and Simulation, code profiling (week 4)

An Introduction to Statistical Learning with Applications in R of two experts Trevor Hastie and Rob Tibshirani from Standfor Unitiversity

This course was introduced by Kevin Markham in r-blogger in september 2014. "I found it to be an excellent course in statistical learning (also known as "machine learning"), largely due to the high quality of both the textbook and the video lectures. And as an R user, it was extremely helpful that they included R code to demonstrate most of the techniques described in the book." In this course you will learn about Statistical Learning, Linear Regression, Classification, Resampling Methods, Linear Model Selection and Regularization, Moving Beyond Linearity, Tree-Based Methods, Support Vector Machines and Unsupervised Learning

Cheatsheet – Python R codes for common Machine Learning Algorithms

8.2 Everything you need to know about R

In this post I maintain all useful references for someone want to write nice R code.

Google's R Style Guide at google R is a high-level programming language used primarily for statistical computing and graphics. The goal of the R Programming Style Guide is to make our R code easier to read, share, and verify. The rules below were designed in collaboration with the entire R user community at Google.

Installing R packages at r-bloggers https://www.r-bloggers.com/installing-r-packages/

This is a short post giving steps on how to actually install R packages.

Managing your projects in a reproducible fashion at nicercode https://nicercode.github.io/blog/2013-04-05-projects/

Managing your projects in a reproducible fashion doesn't just make your science reproducible, it makes your life easier.

Creating R Packages http://cran.r-project.org/doc/contrib/Leisch-CreatingPackages.pdf

This tutorial gives a practical introduction to creating R packages. We discuss how object oriented programming and S formulas can be used to give R code the usual look and feel, how to start a package from a collection of R functions, and how to test the code once the package has been created. As running example we use functions for standard linear regression analysis which are developed from scratch

How to write trycatch in R http://stackoverflow.com/questions/12193779/how-to-write-trycatch-in-r

Welcome to the R world

Debugging with RStudio https://support.rstudio.com/hc/en-us/articles/200713843-Debugging-with-RStudio

RStudio includes a visual debugger that can help you understand code and find bugs.

Optimising code http://adv-r.had.co.nz/Profiling.htmlperformance-profiling

Optimising code to make it run faster is an iterative process:

Find the biggest bottleneck (the slowest part of your code). Try to eliminate it (you may not succeed but that's ok). Repeat until your code is "fast enough." This sounds easy, but it's not.

Chương 9

Scala

View online http://magizbox.com/training/scala/site/

Scala is a programming language for general software applications. Scala has full support for functional programming and a very strong static type system. This allows programs written in Scala to be very concise and thus smaller in size than other general-purpose programming languages. Many of Scala's design decisions were inspired by criticism of the shortcomings of Java.

9.1 Installation

Windows Step 1. Download scala from http://www.scala-lang.org/downloads

Step 2. Run installer

Step 3. Verify

Open terminal and check which version of scala

$scala - version$

Scala code runner version 2.11.5 – Copyright 2002-2013, LAMP/EPFL

9.2 IDE

I use IntelliJ IDEA 2016.2 as scala IDE

IntelliJ IDEA Installation Guide Online IDE You can use tryscala as an online IDE

http://www.tryscala.com/

9.3 Basic Syntax

Print print > println("Hello, Scala!");

Hello, Scala! Conditional if Statement

if statement consists of a Boolean expression followed by one or more statements.

var x = 10; if(x < 20) println("This is if statement"); if-else Statement

var x = 30 if(x < 20) println("This is if statement"); else println("This is else statement"); if-else if-else Statement

var x = 30; if(x == 10) println("Value of X is 10"); else if(x == 20) println("Value of X is 20"); else if(x == 30) println("Value of X is 30"); else println("This is else statement"); Coding Convention 1 Keep It Simple Don't pack two much in one expression /* * It's amazing what you can get done in a single statement * But that does not mean you have to do it. */ jp.getRawClasspath.filter($.getEntryKind == IClasspathEntry.CPE_SOURCE).iterator.flatMap(entry =$
$flatten(ResourcesPlugin.getWorkspace.getRoot.findMember(entry.getPath)))RefactorThere'salotofval$
$jp.getRawClasspath.filter(.getEntryKind == IClasspathEntry.CPE_SOURCE)defworkspaceRoot =$
$ResourcesPlugin.getWorkspace.getRootdeffilesOfEntry(entry : Set[File]) =$
$flatten(worspaceRoot.findMember(entry.getPath)sources.iteratorflatMapfilesOfEntryPreferFunctio$

use vals, not vars use recursions or combinators, not loops use immutable collections concentrate on transformations, not CRUD When to deviate from the default - sometimes, mutable gives better performance. - sometimes (but not that often!) it adds convenience

But don't diablolize local state Why does mutable state lead to complexity?

It interacts with different program parts in ways that are hard to track.

=> Local state is less harmful than global state.

"Var" Shortcuts var interfaces = parseClassHeader()... if (isAnnotation) interfaces += ClassFileAnnotation Refactor

val parsedIfaces = parseClassHeader() val interfaces = if (isAnnotation) parsedIfaces + ClassFileAnnotation else parsedIfaces Martin Odersky - Scala with Style

Chương 10

NodeJS

View online http://magizbox.com/training/nodejs/site/

Node.js is an open-source, cross-platform JavaScript runtime environment for developing a diverse variety of tools and applications. Although Node.js is not a JavaScript framework, many of its basic modules are written in JavaScript, and developers can write new modules in JavaScript. The runtime environment interprets JavaScript using Google's V8 JavaScript engine. Node.js has an event-driven architecture capable of asynchronous I/O. These design choices aim to optimize throughput and scalability in Web applications with many input/output operations, as well as for real-time Web applications (e.g., real-time communication programs and browser games). Node.js was originally written in 2009 by Ryan Dahl. The initial release supported only Linux. Its development and maintenance was led by Dahl and later sponsored by Joyent.

10.1 Get Started

Installation Windows In this section I will show you how to Install Node.js® and NPM on Windows

Prerequisites Node isn't a program that you simply launch like Word or Photoshop: you won't find it pinned to the taskbar or in your list of Apps. To use Node you must type command-line instructions, so you need to be comfortable with (or at least know how to start) a command-line tool like the Windows Command Prompt, PowerShell, Cygwin, or the Git shell (which is installed along with Github for Windows).

Installation Overview Installing Node and NPM is pretty straightforward using the installer package available from the Node.js® web site.

Installation Steps 1. Download the Windows installer from the Nodes.js® web site.

2. Run the installer (the .msi file you downloaded in the previous step.)

3. Follow the prompts in the installer (Accept the license agreement, click the NEXT button a bunch of times and accept the default installation settings).

4. Restart your computer. You won't be able to run Node.js® until you restart your computer.

Ubuntu In this section I will show you how to Install Node.js® and NPM on Ubuntu update os sudo apt-get update install node with apt-get sudo apt-get install nodejs install npm with apt-get sudo apt-get install npm Test Make sure you have Node and NPM installed by running simple commands to see what version of each is installed and to run a simple test program:

> node -v v6.9.5

> npm -v 3.10.10 Suggested Readings How To Install Node.js on an Ubuntu 14.04 server How to Install Node.js® and NPM on Windows

10.2 Basic Syntax

Print console.log("Hello World"); Conditional if(you$_smart)console.log("learnnodejs");elseconsole.log("goau$
$0;count < 10;count{+}{+})console.log(count);Functionfunctionprint_info(arg1, arg2)console.log(arg1);conso$

10.3 File System IO

File System IO Node implements File I/O using simple wrappers around standard POSIX functions. The Node File System (fs) module can be imported using the following syntax

var fs = require("fs") Synchronous vs Asynchronous Every method in the fs module has synchronous as well as asynchronous forms. Asynchronous methods take the last parameter as the completion function callback and the first parameter of the callback function as error. It is better to use an asynchronous method instead of a synchronous method, as the former never blocks a program during its execution, whereas the second one does.

Example

Create a text file named input.txt with the following content

Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!! Let us create a js file named main.js with the following code

var fs = require("fs");

// Asynchronous read fs.readFile('input.txt', function (err, data) if (err) return console.error(err); console.log("Asynchronous read: " + data.toString()););

// Synchronous read var data = fs.readFileSync('input.txt'); console.log("Synchronous read: " + data.toString());

console.log("Program Ended"); Now run the main.js to see the result

$nodemain.jsVerifytheOutput.$

Synchronous read: Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!!

Program Ended Asynchronous read: Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!! The following sections in this chapter provide a set of good examples on major File I/O methods. Open a File Syntax

Following is the syntax of the method to open a file in asynchronous mode

fs.open(path, flags[, mode], callback) Parameters

Here is the description of the parameters used

path This is the string having file name including path. flags Flags indicate the behavior of the file to be opened. All possible values have been mentioned below. mode It sets the file mode (permission and sticky bits), but only if the file was created. It defaults to 0666, readable and writeable. callback This is the callback function which gets two arguments (err, fd). Flags

Flags for read/write operations are

r - Open file for reading. An exception occurs if the file does not exist. r+ - Open file for reading and writing. An exception occurs if the file does not exist. rs - Open file for reading in synchronous mode. rs+ - Open file for reading and writing, asking the OS to open it synchronously. See notes for 'rs' about using this with caution. w - Open file for writing. The file is created (if it does not exist) or truncated (if it exists). wx - Like 'w' but fails if the path exists. w+ - Open file for reading and writing. The file is created (if it does not exist) or truncated (if it exists). wx+ - Like 'w+' but fails if path exists. a - Open file for appending. The file is created if it does not exist. ax - Like 'a' but fails if the path exists. a+ - Open file for reading and appending. The file is created if it does not exist. ax+ - Like 'a+' but fails if the the path exists. Example

Let us create a js file named main.js having the following code to open a file input.txt for reading and writing.

var fs = require("fs");

// Asynchronous - Opening File console.log("Going to open file!"); fs.open('input.txt', 'r+', function(err, fd) if (err) return console.error(err); console.log("File opened successfully!");); Now run the main.js to see the result

$node main.js Verify the Output.$

Going to open file! File opened successfully! Get File Information Syntax

Following is the syntax of the method to get the information about a file

fs.stat(path, callback) Parameters

Here is the description of the parameters used

path This is the string having file name including path. callback This is the callback function which gets two arguments (err, stats) where stats is an object of fs.Stats type which is printed below in the example. Apart from the important attributes which are printed below in the example, there are several useful methods available in fs.Stats class which can be used to check file type. These methods are given in the following table.

Method Description

stats.isFile() - Returns true if file type of a simple file. stats.isDirectory() - Returns true if file type of a directory. stats.isBlockDevice() - Returns true if file type of a block device. stats.isCharacterDevice() - Returns true if file type of a character device. stats.isSymbolicLink() - Returns true if file type of a symbolic link. stats.isFIFO() - Returns true if file type of a FIFO. stats.isSocket() - Returns true if file type of asocket. Example

Let us create a js file named main.js with the following code

var fs = require("fs");

console.log("Going to get file info!"); fs.stat('input.txt', function (err, stats) if (err) return console.error(err); console.log(stats); console.log("Got file info successfully!");

// Check file type console.log("isFile ? " + stats.isFile()); console.log("isDirectory ? " + stats.isDirectory());); Now run the main.js to see the result

$node main.js Verify the Output.$

Going to get file info! dev: 1792, mode: 33188, nlink: 1, uid: 48, gid: 48, rdev: 0, blksize: 4096, ino: 4318127, size: 97, blocks: 8, atime: Sun Mar 22 2015 13:40:00 GMT-0500 (CDT), mtime: Sun Mar 22 2015 13:40:57 GMT-0500 (CDT), ctime:

Sun Mar 22 2015 13:40:57 GMT-0500 (CDT) Got file info successfully! isFile ? true isDirectory ? false Writing a File Syntax

Following is the syntax of one of the methods to write into a file

fs.writeFile(filename, data[, options], callback) This method will over-write the file if the file already exists. If you want to write into an existing file then you should use another method available.

Parameters

Here is the description of the parameters used

path This is the string having the file name including path. data This is the String or Buffer to be written into the file. options The third parameter is an object which will hold encoding, mode, flag. By default. encoding is utf8, mode is octal value 0666. and flag is 'w' callback This is the callback function which gets a single parameter err that returns an error in case of any writing error. Example

Let us create a js file named main.js having the following code

var fs = require("fs");

console.log("Going to write into existing file"); fs.writeFile('input.txt', 'Simply Easy Learning!', function(err) if (err) return console.error(err);

console.log("Data written successfully!"); console.log("Let's read newly written data"); fs.readFile('input.txt', function (err, data) if (err) return console.error(err); console.log("Asynchronous read: " + data.toString()););); Now run the main.js to see the result

$node main.js Verify the Output.$

Going to write into existing file Data written successfully! Let's read newly written data Asynchronous read: Simply Easy Learning! Reading a File Syntax

Following is the syntax of one of the methods to read from a file

fs.read(fd, buffer, offset, length, position, callback) This method will use file descriptor to read the file. If you want to read the file directly using the file name, then you should use another method available.

Parameters

Here is the description of the parameters used

fd This is the file descriptor returned by fs.open(). buffer This is the buffer that the data will be written to. offset This is the offset in the buffer to start writing at. length This is an integer specifying the number of bytes to read. position This is an integer specifying where to begin reading from in the file. * If position is null, data will be read from the current file position. callback This is the callback function which gets the three arguments, (err, bytesRead, buffer). Example

Let us create a js file named main.js with the following code

var fs = require("fs"); var buf = new Buffer(1024);

console.log("Going to open an existing file"); fs.open('input.txt', 'r+', function(err, fd) if (err) return console.error(err); console.log("File opened successfully!"); console.log("Going to read the file"); fs.read(fd, buf, 0, buf.length, 0, function(err, bytes) if (err) console.log(err); console.log(bytes + " bytes read");

// Print only read bytes to avoid junk. if(bytes > 0) console.log(buf.slice(0, bytes).toString()););); Now run the main.js to see the result

$node main.js Verify the Output.$

Going to open an existing file File opened successfully! Going to read the file 97 bytes read Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!! Closing a File Syntax

Following is the syntax to close an opened file

fs.close(fd, callback) Parameters

Here is the description of the parameters used

fd This is the file descriptor returned by file fs.open() method. callback This is the callback function No arguments other than a possible exception are given to the completion callback. Example Let us create a js file named main.js having the following code

var fs = require("fs"); var buf = new Buffer(1024);

console.log("Going to open an existing file"); fs.open('input.txt', 'r+', function(err, fd) if (err) return console.error(err); console.log("File opened successfully!"); console.log("Going to read the file");

fs.read(fd, buf, 0, buf.length, 0, function(err, bytes) if (err) console.log(err);

// Print only read bytes to avoid junk. if(bytes > 0) console.log(buf.slice(0, bytes).toString());

// Close the opened file. fs.close(fd, function(err) if (err) console.log(err); console.log("File closed successfully.");););); Now run the main.js to see the result

$nodemain.jsVerifytheOutput.$

Going to open an existing file File opened successfully! Going to read the file Tutorials Point is giving self learning content to teach the world in simple and easy way!!!!!

File closed successfully. Truncate a File Syntax

Following is the syntax of the method to truncate an opened file

fs.ftruncate(fd, len, callback) Parameters

Here is the description of the parameters used

fd This is the file descriptor returned by fs.open(). len This is the length of the file after which the file will be truncated. callback This is the callback function No arguments other than a possible ekxception are given to the completion callback. Example

Let us create a js file named main.js having the following code

var fs = require("fs"); var buf = new Buffer(1024);

console.log("Going to open an existing file"); fs.open('input.txt', 'r+', function(err, fd) if (err) return console.error(err); console.log("File opened successfully!"); console.log("Going to truncate the file after 10 bytes");

// Truncate the opened file. fs.ftruncate(fd, 10, function(err) if (err) console.log(err); console.log("File truncated successfully."); console.log("Going to read the same file");

fs.read(fd, buf, 0, buf.length, 0, function(err, bytes) if (err) console.log(err);

// Print only read bytes to avoid junk. if(bytes > 0) console.log(buf.slice(0, bytes).toString());

// Close the opened file. fs.close(fd, function(err) if (err) console.log(err); console.log("File closed successfully."););););); Now run the main.js to see the result

$nodemain.jsVerifytheOutput.$

Going to open an existing file File opened successfully! Going to truncate the file after 10 bytes File truncated successfully. Going to read the same file Tutorials File closed successfully. Delete a File Syntax Following is the syntax of the method to delete a file

fs.unlink(path, callback) Parameters

Here is the description of the parameters used path This is the file name including path. callback This is the callback function No arguments other than a possible exception are given to the completion callback. Example

Let us create a js file named main.js having the following code

var fs = require("fs");

console.log("Going to delete an existing file"); fs.unlink('input.txt', function(err) if (err) return console.error(err); console.log("File deleted successfully!");); Now run the main.js to see the result

$nodemain.jsVerifytheOutput.$

Going to delete an existing file File deleted successfully! Create a Directory Syntax

Following is the syntax of the method to create a directory

fs.mkdir(path[, mode], callback) Parameters

Here is the description of the parameters used

path This is the directory name including path. mode This is the directory permission to be set. Defaults to 0777. callback This is the callback function No arguments other than a possible exception are given to the completion callback. Example

Let us create a js file named main.js having the following code

var fs = require("fs");

console.log("Going to create directory /tmp/test"); fs.mkdir('/tmp/test',function(err) if (err) return console.error(err); console.log("Directory created successfully!");); Now run the main.js to see the result

$nodemain.jsVerifytheOutput.$

Going to create directory /tmp/test Directory created successfully! Read a Directory Syntax

Following is the syntax of the method to read a directory

fs.readdir(path, callback) Parameters

Here is the description of the parameters used

path This is the directory name including path. callback This is the callback function which gets two arguments (err, files) where files is an array of the names of the files in the directory excluding '.' and '..'. Example

Let us create a js file named main.js having the following code

var fs = require("fs");

console.log("Going to read directory /tmp"); fs.readdir("/tmp/",function(err, files) if (err) return console.error(err); files.forEach(function (file) console.log(file););); Now run the main.js to see the result

$nodemain.jsVerifytheOutput.$

Going to read directory /tmp ccmzx99o.out ccyCSbkF.out employee.ser hsperfdata$_{a}pachetesttest.txtRem$

Following is the syntax of the method to remove a directory

fs.rmdir(path, callback) Parameters

Here is the description of the parameters used

path This is the directory name including path. callback This is the callback function No argume nts other than a possible exception are given to the completion callback. Example

Let us create a js file named main.js having the following code

var fs = require("fs");

console.log("Going to delete directory /tmp/test"); fs.rmdir("/tmp/test",function(err) if (err) return console.error(err); console.log("Going to read directory /tmp");

fs.readdir("/tmp/",function(err, files) if (err) return console.error(err); files.forEach(function (file) console.log(file);););); Now run the main.js to see the result

nodemain.jsVerifytheOutput.

Going to read directory /tmp ccmzx99o.out ccyCSbkF.out employee.ser hsperfdata$_a$*pachetest.txt*

10.4 Package Manager

Package Manager: NPM Node Package Manager (NPM) provides two main functionalities

Online repositories for node.js packages/modules which are searchable on search.nodejs.org Command line utility to install Node.js packages, do version management and dependency management of Node.js packages. NPM comes bundled with Node.js installables after v0.6.3 version. To verify the same, open console and type the following command and see the result

*npm−−version*2.7.1*IfyouarerunninganoldversionofNPMthenitisquiteeasytoupdateittothelatestversi*

sudonpminstallnpm−g/usr/bin/npm− > */usr/lib/node*$_m$*odules/npm/bin/npm−cli.jsnpm*@2.7.1*/usr/lib/node*$_m$*odules/npmInstallingModulesThereisasimplesyntaxtoinstallanyNode.js*

npminstall < *ModuleName* > *Forexample, followingisthecommandtoinstallafamousNode.jswebfram*

npminstallexpressNowyoucanusethismoduleinyourjsfileasfollowing

var express = require('express'); Global vs Local Installation By default, NPM installs any dependency in the local mode. Here local mode refers to the package installation in node$_m$*odulesdirectorylyinginthefolderwhereNodeapplicationispresent.Locallydeploy*

*ls−ltotal*0*drwxr−xr−x*3*rootroot*20*Mar*1702 : 23*node*$_m$*odulesAlternatively, youcanusenpmlscommandt*

Globally installed packages/dependencies are stored in system directory. Such dependencies can be used in CLI (Command Line Interface) function of any node.js but cannot be imported using require() in Node application directly. Now let's try installing the express module using global installation.

npminstallexpress−gThiswillproduceasimilarresultbutthemodulewillbeinstalledglobally.Here, thefirs

express@4.12.2 /usr/lib/node$_m$*odules/expressmerge−descriptors*@1.0.0*utils−merge*@1.0.0*cookie−signature*@1.0.6*methods*@1.1.1*fresh*@0.2.4*cookie*@0.1.2*escape−html*@1.0.1*range−parser*@1.0.2*content−type*@1.0.1*finalhandler*@0.3.3*vary*@1.0.0*parseurl*@1.3.0*content−disposition*@0.5.0*path−to−regexp*@0.1.3*depd*@1.0.0*qs*@2.3.3*on−finished*@2.2.0(*ee−first*@1.1.0)*etag*@1.5.1(*crc*@3.2.1)*debug*@2.1.3(*ms*@0.7.0)*proxy−addr*@1.0.7(*forwarded*@0.1.0, *ipaddr.js*@0 *static*@1.9.2(*send*@0.12.2)*accepts*@1.2.5(*negotiator*@0.5.1, *mime−types*@2.0.10)*type−is*@1.6.1(*media−typer*@0.3.0, *mime−types*@2.0.10)*Youcanusethefollowingcommandtocheckallthemodules*

npmls−gUsingpackage.jsonpackage.jsonispresentintherootdirectoryofanyNodeapplication/modulean

"name": "express", "description": "Fast, unopinionated, minimalist web framework", "version": "4.11.2", "author":

"name": "TJ Holowaychuk", "email": "tj@vision-media.ca" ,

"contributors": ["name": "Aaron Heckmann", "email": "aaron.heckmann+github@gmail.com"
,

"name": "Ciaran Jessup", "email": "ciaranj@gmail.com" ,

"name": "Douglas Christopher Wilson", "email": "doug@somethingdoug.com"
,

"name": "Guillermo Rauch", "email": "rauchg@gmail.com" ,

"name": "Jonathan Ong", "email": "me@jongleberry.com" ,

"name": "Roman Shtylman", "email": "shtylman+expressjs@gmail.com" ,

"name": "Young Jae Sim", "email": "hanul@hanul.me"], "license": "MIT", "repository": "type": "git", "url": "https://github.com/strongloop/express" ,

"homepage": "https://expressjs.com/", "keywords": ["express", "framework", "sinatra", "web", "rest", "restful", "router", "app", "api"], "dependencies": "accepts": " 1.2.3", "content-disposition": "0.5.0", "cookie-signature": "1.0.5", "debug": " 2.1.1", "depd": " 1.0.0", "escape-html": "1.0.1", "etag": " 1.5.1", "finalhandler": "0.3.3", "fresh": "0.2.4", "media-typer": "0.3.0", "methods": " 1.1.1", "on-finished": " 2.2.0", "parseurl": " 1.3.0", "path-to-regexp": "0.1.3", "proxy-addr": " 1.0.6", "qs": "2.3.3", "range-parser": " 1.0.2", "send": "0.11.1", "serve-static": " 1.8.1", "type-is": " 1.5.6", "vary": " 1.0.0", "cookie": "0.1.2", "merge-descriptors": "0.0.2", "utils-merge": "1.0.0" , "devDependencies": "after": "0.8.1", "ejs": "2.1.4", "istanbul": "0.3.5", "marked": "0.3.3", "mocha": " 2.1.0", "should": " 4.6.2", "supertest": " 0.15.0", "hjs": " 0.0.6", "body-parser": " 1.11.0", "connect-redis": " 2.2.0", "cookie-parser": " 1.3.3", "express-session": " 1.10.2", "jade": " 1.9.1", "method-override": " 2.3.1", "morgan": " 1.5.1", "multiparty": " 4.1.1", "vhost": " 3.0.0" , "engines": "node": ">= 0.10.0" , "files": ["LICENSE", "History.md", "Readme.md", "index.js", "lib/"], "scripts": "test": "mocha –require test/support/env –reporter spec –bail –check-leaks test/ test/acceptance/", "test-cov": "istanbul cover node$_m$odules/mocha/bin/$_m$ocha−
−−−requiretest/support/env−−reporterdot−−check−leakstest/test/acceptance/”,”test−
tap” : ”mocha−−requiretest/support/env−−reportertap−−check−leakstest/test/acceptance/”,”test−
travis” : ”istanbulcovernode$_m$odules/mocha/bin/$_m$ocha − −reportlcovonly −
−−−requiretest/support/env−−reporterspec−−check−leakstest/test/acceptance/”,”gitHead” :
”63ab25579bda70b4927a179b580a9c580b6c7ada”,”bugs” : ”url” : ”https : //github.com/strongloop/express/
”express@4.11.2”,”$_s$hasum” : ”8df3d5a9ac848585f00a0777601823faecd3b148”,”$_f$rom” :
”express@∗”,”$_n$pmVersion” : ”1.4.28”,”$_n$pmUser” : ”name” : ”dougwilson”,”email” : ”doug@somethingd
[”name” : ”tjholowaychuk”,”email” : ”tj@vision − media.ca”,”name” : ”jongleberry”,”email” : ”jonatha
”shasum” : ”8df3d5a9ac848585f00a0777601823faecd3b148”,”tarball” : ”https : //registry.npmjs.org/expr
,”$_r$esolved” : ”https : //registry.npmjs.org/express/−/express−4.11.2.tgz”,”readme” :
”ERROR : NoREADMEdatafound!” AttributesofPackage.jsonnamenameofthepackageversionversiono

npmuninstallexpressOnceNPMuninstallsthepackage, youcanverifyitbylookingatthecontentof/node$_m$

npmlsUpdatingaModuleUpdatepackage.jsonandchangetheversionofthedependencytobeupdatedandrunt

npmupdateexpressSearchaModuleSearchapackagenameusingNPM.

npmsearchexpressCreateaModuleCreatingamodulerequirespackage.jsontobegenerated.Let′sgeneratep

npminit

This utility will walk you through creating a package.json file. It only covers the most common items, and tries to guess sane defaults.

See 'npm help json' for definitive documentation on these fields and exactly what they do.

Use 'npm install <pkg> –save' afterwards to install a package and save it as a dependency in the package.json file.

Press ^{C}atanytimetoquit.name : (webmaster)Youwillneedtoprovidealltherequiredinformationaboutyour
mentionedpackage.jsonfiletounderstandthemeaningsofvariousinformationdemanded.Oncepackage.json

npmadduserUsername : mcmohdPassword : Email : (thisISpublic)mcmohd@gmail.comItistimenowto

npmpublishIfeverythingisfinewithyourmodule, thenitwillbepublishedintherepositoryandwillbeaccessi

10.5 Command Line

Pass command line arguments The arguments are stored in process.argv

Here are the node docs on handling command line args:

process.argv is an array containing the command line arguments. The first element will be 'node', the second element will be the name of the JavaScript file. The next elements will be any additional command line arguments.

// print process.argv process.argv.forEach(function (val, index, array) console.log(index + ': ' + val);); This will generate:

$nodeprocess-2.jsonetwo = threefour0 : node1 : /Users/mjr/work/node/process-2.js2 : one3 : two = three4 : four$

Chương 11

Octave

View online http://magizbox.com/training/octave/site/

GNU Octave is software featuring a high-level programming language, primarily intended for numerical computations. Octave helps in solving linear and nonlinear problems numerically, and for performing other numerical experiments using a language that is mostly compatible with MATLAB. It may also be used as a batch-oriented language. Since it is part of the GNU Project, it is free software under the terms of the GNU General Public License.

Known Issues Plot window not responding

11.1 Matrix

Creating Matrix A = [1, 1, 2; 3, 5, 8; 13, 21, 32] A = 1 1 2 3 5 8 13 21 32

Creating an 1D column vector a = [1; 2; 3] a = 1 2 3

Creating an 1D row vector b = [1 2 3] b = 1 2 3

Creating a random m x n matrix rand(3, 2) ans = 0.13567 0.51230 0.67646 0.19012 0.76147 0.89694

Creating a zero m x n matrix zeros(3, 2) ans = 0 0 0 0 0 0

Creating an m x n matrix of ones ones(3,2) ans = 1 1 1 1 1 1

Creating an identity matrix eye(3) Diagonal Matrix 1 0 0 0 1 0 0 0 1

Creating a diagonal matrix a = [1 2 3] diag(a) Diagonal Matrix 1 0 0 0 2 0 0 0 3 Accessing Matrix Elements Getting the dimension of a matrix A = [1 2 3; 4 5 6] size(A) ans = 2 3

Selecting rows A = [1 2 3; 4 5 6; 7 8 9] A(1, :) ans = 1 2 3 A(1:2, :) ans = 1 2 3 4 5 6

Selecting columns A = [1 2 3; 4 5 6; 7 8 9] A(:, 1) ans = 1 4 7 A(:, 1:2) ans = 1 2 4 5 7 8

Extracting rows and columns by criteria A = [1 2 3; 4 5 9; 7 8 9] A(A(:,3) == 9, :) ans = 4 5 9 7 8 9

Accessing elements A = [1 2 3; 4 5 6; 7 8 9] A(1, 1) ans = 1 A(2, 3) ans = 6 Manipulating Shape and Dimensions Converting a matrix into a row vector (by column) A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A(:) ans = 1 4 7 2 5 8 3 6 9

Converting row to column vectors b = [1 2 3] b = 1 2 3 b' ans = 1 2 3

Reshaping Matrices A = [1 2 3 4; 5 6 7 8; 9 10 11 12] A = 1 2 3 4 5 6 7 8 9 10 11 12 reshape(A,4,3) ans = 1 6 11 5 10 4 9 3 8 2 7 12

Concatenating matrices A = [1 2 3; 4 5 6] A = 1 2 3 4 5 6 B = [7 8 9; 10 11 12] B = 7 8 9 10 11 12 C = [A; B] C = 1 2 3 4 5 6 7 8 9 10 11 12

Stacking vectors and matrices a = [1 2 3] a = 1 2 3 b = [4 5 6] b = 4 5 6 c = [a' b'] c = 1 4 2 5 3 6 Basic Operations Matrix-scalar operations A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A * 2 ans = 2 4 6 8 10 12 14 16 18 A + 2 ans = 3 4 5 6 7 8 9 10 11 A - 2 ans = -1 0 1 2 3 4 5 6 7 A / 2 ans = 0.50000 1.00000 1.50000 2.00000 2.50000 3.00000 3.50000 4.00000 4.50000

Matrix-matrix multiplication A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A * A ans = 30 36 42 66 81 96 102 126 150

Matrix-vector multiplication A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 b = [1; 2; 3] b = 1 2 3 A * b ans = 14 32 50

Element-wise matrix-matrix operations A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A .* A ans = 1 4 9 16 25 36 49 64 81 A .+ A ans = 2 4 6 8 10 12 14 16 18 A .- A ans = 0 0 0 0 0 0 0 0 0 A ./ A ans = 1 1 1 1 1 1 1 1 1

Matrix elements to power n A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A.$^2 ans =$ 149162536496481

Matrix to power n A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A $^2 ans =$ 303642668196102126150

Matrix transpose A = [1 2 3; 4 5 6; 7 8 9] A = 1 2 3 4 5 6 7 8 9 A' ans = 1 4 7 2 5 8 3 6 9

Determinant of a matrix A = [6 1 1; 4 -2 5; 2 8 7] A = 6 1 1 4 -2 5 2 8 7 det(A) ans = -306

Inverse of a matrix A = [4 7; 2 6] A = 4 7 2 6 inv(A) ans = 0.60000 -0.70000 -0.20000 0.40000 Advanced Operations Calculating the covariance matrix of 3 random variables x1 = [4.0000 4.2000 3.9000 4.3000 4.1000]' x1 = 4.0000 4.2000 3.9000 4.3000 4.1000 x2 = [2.0000 2.1000 2.0000 2.1000 2.2000]' x2 = 2.0000 2.1000 2.0000 2.1000 2.2000 x3 = [0.60000 0.59000 0.58000 0.62000 0.63000]' x3 = 0.60000 0.59000 0.58000 0.62000 0.63000 cov([x1,x2,x3]) ans = 2.5000e-002 7.5000e-003 1.7500e-003 7.5000e-003 7.0000e-003 1.3500e-003 1.7500e-003 1.3500e-003 4.3000e-004

Calculating eigenvectors and eigenvalues A = [3 1; 1 3] A = 3 1 1 3 [eig$_vec, eig_val] = eig(A)eig_vec = -0.707110.707110.707110.70711eig_val = DiagonalMatrix$2004

Generating a Gaussian dataset pkg load statistics mean = [0 0] mean = 0 0 cov = [2 0; 0 2] cov = 2 0 0 2 mvnrnd(mean,cov,5) ans = -0.7442485 -0.0099190 -1.7695915 0.0418147 -0.8780206 0.6145333 0.5145315 -0.9834832 -1.4736628 0.4570979

Chương 12

Toolbox

View online http://magizbox.com/training/toolbox/site/

Toolbox by MG The Toolbox contains all the little tools you never know where to find.

Text Editor Vim : Vim is a clone of Bill Joy's vi text editor program for Unix. It was written by Bram Moolenaar based on source for a port of the Stevie editor to the Amiga and first released publicly in 1991. Vim is designed for use both from a command-line interface and as a standalone application in a graphical user interface. Vim is free and open source software and is released under a license that includes some charityware clauses, encouraging users who enjoy the software to consider donating to children in Uganda. The license is compatible with the GNU General Public License. Although it was originally released for the Amiga, Vim has since been developed to be cross-platform, supporting many other platforms. In 2006, it was voted the most popular editor amongst Linux Journal readers; in 2015 the Stack Overflow developer survey found it to be the third most popular text editor; and in 2016 the Stack Overflow developer survey found it to be the fourth most popular development environment.

Virtual Machine VirtualBox : Oracle VM VirtualBox (formerly Sun VirtualBox, Sun xVM VirtualBox and Innotek VirtualBox) is a free and open-source hypervisor for x86 computers currently being developed by Oracle Corporation. Developed initially by Innotek GmbH, it was acquired by Sun Microsystems in 2008 which was in turn acquired by Oracle in 2010. VirtualBox may be installed on a number of host operating systems, including: Linux, macOS, Windows, Solaris, and OpenSolaris. There are also ports to FreeBSD and Genode. It supports the creation and management of guest virtual machines running versions and derivations of Windows, Linux, BSD, OS/2, Solaris, Haiku, OSx86 and others, and limited virtualization of macOS guests on Apple hardware. For some guest operating systems, a "Guest Additions" package of device drivers and system applications is available which typically improves performance, especially of graphics.

VMWare : VMware, Inc. is a subsidiary of Dell Technologies that provides cloud computing and platform virtualization software and services. It was the first commercially successful company to virtualize the x86 architecture. VMware's desktop software runs on Microsoft Windows, Linux, and macOS, while its enterprise software hypervisor for servers, VMware ESXi, is a bare-metal hypervisor that runs directly on server hardware without requiring an additional underlying operating system.

12.1 Vim

Vim Running Vim for the First Time To start Vim, enter this command:

gvim file.txt In UNIX you can type this at any command prompt. If you are running Microsoft Windows, open an MS-DOS prompt window and enter the command. In either case, Vim starts editing a file called file.txt. Because this is a new file, you get a blank window. This is what your screen will look like:

+———————————————+ | | | | | | | | | | | |"file.txt" [New file] | +———————————————+ ('" is the cursor position.) The tilde ( ) lines indicate lines not in the file. In other words, when Vim runs out of file to display, it displays tilde lines. At the bottom of the screen, a message line indicates the file is named file.txt and shows that you are creating a new file. The message information is temporary and other information overwrites it.

THE VIM COMMAND

The gvim command causes the editor to create a new window for editing. If you use this command:

vim file.txt the editing occurs inside your command window. In other words, if you are running inside an xterm, the editor uses your xterm window. If you are using an MS-DOS command prompt window under Microsoft Windows, the editing occurs inside this window. The text in the window will look the same for both versions, but with gvim you have extra features, like a menu bar. More about that later.

Inserting text The Vim editor is a modal editor. That means that the editor behaves differently, depending on which mode you are in. The two basic modes are called Normal mode and Insert mode. In Normal mode the characters you type are commands. In Insert mode the characters are inserted as text. Since you have just started Vim it will be in Normal mode. To start Insert mode you type the "i" command (i for Insert). Then you can enter the text. It will be inserted into the file. Do not worry if you make mistakes; you can correct them later. To enter the following programmer's limerick, this is what you type:

iA very intelligent turtle Found programming UNIX a hurdle After typing "turtle" you press the key to start a new line. Finally you press the key to stop Insert mode and go back to Normal mode. You now have two lines of text in your Vim window:

+———————————————+ |A very intelligent turtle | |Found programming UNIX a hurdle | | | | | | | +———————————————+ WHAT IS THE MODE?

To be able to see what mode you are in, type this command:

:set showmode You will notice that when typing the colon Vim moves the cursor to the last line of the window. That's where you type colon commands (commands that start with a colon). Finish this command by pressing the <Enter> key (all commands that start with a colon are finished this way). Now, if you type the "i" command Vim will display –INSERT– at the bottom of the window. This indicates you are in Insert mode.

+———————————————+ |A very intelligent turtle | |Found programming UNIX a hurdle | | | | | | |– INSERT – | +———————————————+ If you press <Esc> to go back to Normal mode the last line will be made blank.

GETTING OUT OF TROUBLE

One of the problems for Vim novices is mode confusion, which is caused by forgetting which mode you are in or by accidentally typing a command that switches modes. To get back to Normal mode, no matter what mode you are in, press the key. Sometimes you have to press it twice. If Vim beeps back at you, you already are in Normal mode.

==

Moving around After you return to Normal mode, you can move around by using these keys:

h left *hjkl* j down k up l right At first, it may appear that these commands were chosen at random. After all, who ever heard of using l for right? But actually, there is a very good reason for these choices: Moving the cursor is the most common thing you do in an editor, and these keys are on the home row of your right hand. In other words, these commands are placed where you can type them the fastest (especially when you type with ten fingers).

Note: You can also move the cursor by using the arrow keys. If you do, however, you greatly slow down your editing because to press the arrow keys, you must move your hand from the text keys to the arrow keys. Considering that you might be doing it hundreds of times an hour, this can take a significant amount of time. Also, there are keyboards which do not have arrow keys, or which locate them in unusual places; therefore, knowing the use of the hjkl keys helps in those situations. One way to remember these commands is that h is on the left, l is on the right and j points down. In a picture:

k h l j The best way to learn these commands is by using them. Use the "i" command to insert some more lines of text. Then use the hjkl keys to move around and insert a word somewhere. Don't forget to press to go back to Normal mode. The |vimtutor| is also a nice way to learn by doing.

For Japanese users, Hiroshi Iwatani suggested using this:

Komsomolsk $^{|} HuanHo <------> LosAngeles(Yellowriver)|vJava(theisland, nottheprogramming$

Deleting characters To delete a character, move the cursor over it and type "x". (This is a throwback to the old days of the typewriter, when you deleted things by typing xxxx over them.) Move the cursor to the beginning of the first line, for example, and type xxxxxxx (seven x's) to delete "A very ". The result should look like this:

+——————————————+ |intelligent turtle | |Found programming UNIX a hurdle | | || | | | | +——————————————+ Now you can insert new text, for example by typing:

iA young <Esc> This begins an insert (the i), inserts the words "A young", and then exits insert mode (the final). The result:

+——————————————+ |A young intelligent turtle | |Found programming UNIX a hurdle | | || | | | | +——————————————+ DELETING A LINE

To delete a whole line use the "dd" command. The following line will then move up to fill the gap:

+——————————————+ |Found programming UNIX a hurdle | | | | || | | | | +——————————————+ DELETING A LINE BREAK

In Vim you can join two lines together, which means that the line break between them is deleted. The "J" command does this. Take these two lines:

A young intelligent turtle Move the cursor to the first line and press "J":

A young intelligent turtle ==

Undo and Redo Suppose you delete too much. Well, you can type it in again, but an easier way exists. The "u" command undoes the last edit. Take a look at this in action: After using "dd" to delete the first line, "u" brings it back. Another one: Move the cursor to the A in the first line:

A young intelligent turtle Now type xxxxxxx to delete "A young". The result is as follows:

intelligent turtle Type "u" to undo the last delete. That delete removed the g, so the undo restores the character.

g intelligent turtle The next u command restores the next-to-last character deleted:

ng intelligent turtle The next u command gives you the u, and so on:

ung intelligent turtle oung intelligent turtle young intelligent turtle young intelligent turtle A young intelligent turtle

Note: If you type "u" twice, and the result is that you get the same text back, you have Vim configured to work Vi compatible. Look here to fix this: |not-compatible|. This text assumes you work "The Vim Way". You might prefer to use the good old Vi way, but you will have to watch out for small differences in the text then. REDO

If you undo too many times, you can press CTRL-R (redo) to reverse the preceding command. In other words, it undoes the undo. To see this in action, press CTRL-R twice. The character A and the space after it disappear:

young intelligent turtle There's a special version of the undo command, the "U" (undo line) command. The undo line command undoes all the changes made on the last line that was edited. Typing this command twice cancels the preceding "U".

A very intelligent turtle xxxx Delete very

A intelligent turtle xxxxxx Delete turtle

A intelligent Restore line with "U" A very intelligent turtle Undo "U" with "u" A intelligent The "U" command is a change by itself, which the "u" command undoes and CTRL-R redoes. This might be a bit confusing. Don't worry, with "u" and CTRL-R you can go to any of the situations you had. More about that in section |32.2|.

Reference: http://vimdoc.sourceforge.net/htmldoc/usr$_0$2.*html*

12.2 Virtual Box

Virtual Box Export and Import VirtualBox VM images? Export Open VirtualBox and enter into the File option to choice Export Appliance...

You will then get an assistance window to help you generating the image.

Select the VM to export Enter the output file path and name

You can choice a format, which I always leave the default OVF 1.

Finally you can write metadata like Version and Description Now you have an OVA file that you can carry to whatever machine to use it.

Import Open VirtualBox and enter into the File option to choice Import

You will then get an assistance window to help you loading the image.

Enter the path to the file that you have previously exported

Then you can modify the settings of the VM like RAM size, CPU, etc.

My recommendation on this is to enable the Reinitialize the MAC address of all the network cards option

Press Import and done! Now you have cloned the VM from the host machine into another one

Reference: https://askubuntu.com/questions/588426/how-to-export-and-import-virtualbox-vm-images

Install Guest Additions Guest Additions installs on the guest system and includes device drivers and system applications that optimize performance of the machine. Launch the guest OS in VirtualBox and click on Devices and Install Guest Additions.

The AutoPlay window opens on the guest OS and click on the Run VBox Windows Additions executable.

Click yes when the UAC screen comes up.

Now simply follow through the installation wizard.

During the installation wizard you can choose the Direct3D acceleration if you would like it. Remember this is going to take up more of your Host OS's resources and is still experimental possibly making the guest unstable.

When the installation starts you will need to allow the Sun display adapters to be installed.

After everything has completed a reboot is required.

12.3 VMWare

VMWare VMware Workstation is a program that allows you to run a virtual computer within your physical computer. The virtual computer runs as if it was its own machine. A virtual machine is great for trying out new operating systems such as Linux, visiting websites you don't trust, creating a computing environment specifically for children, testing the effects of computer viruses, and much more. You can even print and plug in USB drives. Read this guide to get the most out of VMware Workstation.

Installing VMware Workstation

1. Make sure your computer meets the system requirements. Because you will be running an operating system from within your own operating system, VMware Workstation has fairly high system requirements. If you don't meet these, you may not be able to run VMware effectively. You must have a 64-bit processor. VMware supports Windows and Linux operating systems. You must have enough memory to run your operating system, the virtual operating system, and any programs inside that operating system. 1 GB is the minimum, but 3 or more is recommended. You must have a 16-bit or 32-bit display adapter. 3D effects will most likely not work well inside the virtual operating system, so gaming is not always efficient. You need at least 1.5 GB of free space to install VMware Workstation, along with at least 1 GB per operating system that you install.

2. Download the VMware software. You can download the VMware installer from the Download Center on the VMware website. Select the newest version and click the link for the installer. You will need to login with your VMware username. You will be asked to read and review the license agreement before you can download the file. You can only have one version of VMware Workstation installed at a time.

3. Install VMware Workstation. Once you have downloaded the file, right-click on the file and select "Run as administrator". You will be asked to review the license again. Most users can use the Typical installation option. At the end of the installation, you will be prompted for your license key. Once the installation is finished, restart the computer. Part

Installing an Operating System

1. Open VMware. Installing a virtual operating system is much like installing it on a regular PC. You will need to have the installation disc or ISO image as well as any necessary licenses for the operating system that you want to install.

You can install most distributions of Linux as well as any version of Windows.

2. Click File. Select New Virtual Machine and then choose Typical. VMware will prompt you for the installation media. If it recognizes the operating system, it will enable Easy Installation:

Physical disc – Insert the installation disc for the operating system you want to install and then select the drive in VMware. ISO image – Browse to the location of the ISO file on your computer. Install operating system later. This will create a blank virtual disk. You will need to manually install the operating system later.

3. Enter in the details for the operating system. For Windows and other licensed operating systems, you will need to enter your product key. You will also need to enter your preferred username and a password if you want one. * If you are not using Easy Install, you will need to browse the list for the operating system you are installing.

4. Name your virtual machine. The name will help you identify it on your physical computer. It will also help distinguish between multiple virtual computers running different operating systems.

5. Set the disk size. You can allocate any amount of free space on your computer to the virtual machine to act as the installed operating system's hard drive. Make sure to set enough to install any programs that you want to run in the virtual machine.

6. Customize your virtual machine's virtual hardware. You can set the virtual machine to emulate specific hardware by clicking the "Customize Hardware" button. This can be useful if you are trying to run an older program that only supports certain hardware. Setting this is optional.

7. Set the virtual machine to start. Check the box labeled "Power on this virtual machine after creation" if you want the virtual machine to start up as soon as you finish making it. If you don't check this box, you can select your virtual machine from the list in VMware and click the Power On button.

8. Wait for your installation to complete. Once you've powered on the virtual machine for the first time, the operating system will begin to install automatically. If you provided all of the correct information during the setup of the virtual machine, then you should not have to do anything. If you didn't enter your product key or create a username during the virtual machine setup, you will most likely be prompted during the installation of the operating system.

9. Check that VMware Tools is installed. Once the operating system is installed, the program VMware Tools should be automatically installed. Check that it appears on the desktop or in the program files for the newly installed operating system.

VMware tools are configuration options for your virtual machine, and keeps your virtual machine up to date with any software changes.

Navigating VMware

1. Start a virtual machine. To start a virtual machine, click the VM menu and select the virtual machine that you want to turn on. You can choose to start the virtual machine normally, or boot directly to the virtual BIOS.

2. Stop a virtual machine. To stop a virtual machine, select it and then click the VM menu. Select the Power option.

Power Off – The virtual machine turns off as if the power was cut out. Shut Down Guest – This sends a shutdown signal to the virtual machine which causes the virtual machine to shut down as if you had selected the shutdown option. You can also turn off the virtual machine by using the shutdown option in the virtual operating system.

3. Move files between the virtual machine and your physical computer. Moving files between your computer and the virtual machine is as simple as dragging and dropping. Files can be moved in both directions between the computer and the virtual machine, and can also be dragged from one virtual machine to another.

When you drag and drop, the original will stay in the original location and a copy will be created in the new location. You can also move files by copying and pasting. Virtual machines can connect to shared folders as well.

4. Add a printer to your virtual machine. You can add any printer to your virtual machine without having to install any extra drivers, as long as it is already installed on your host computer.

Select the virtual machine that you want to add the printer to. Click the VM menu and select Settings. Click the Hardware tab, and then click Add. This will start the Add Hardware wizard. Select Printer and then click Finish. Your virtual printer will be enabled the next time you turn the virtual machine on.

5. Connect a USB drive to the virtual machine. Virtual machines can interact with a USB drive the same way that your normal operating system does. The USB drive cannot be accessed on both the host computer and the virtual machine at the same time.

If the virtual machine is the active window, the USB drive will be automatically connected to the virtual machine when it is plugged in. If the virtual machine is not the active window or is not running, select the virtual machine and click the VM menu. Select Removable Devices and then click Connect. The USB drive will automatically connect to your virtual machine.

6. Take a snapshot of a virtual machine. A snapshot is a saved state and will allow you to load the virtual machine to that precise moment as many times as you need.

Select your virtual machine, click the VM menu, hover over Snapshot and select Take Snapshot. Give your Snapshot a name. You can also give it a description, though this is optional. Click OK to save the Snapshot. Load a saved Snapshot by clicking the VM menu and then selecting Snapshot. Choose the Snapshot you wish to load from the list and click Go To.

7. Become familiar with keyboard shortcuts. A combination of the "Ctrl" and other keys are used to navigate virtual machines. For example, "Ctrl," "Alt" and "Enter" puts the current virtual machine in full screen mode or moves through multiple machines. "Ctrl," "Alt" and "Tab" will move between virtual machines when the mouse is being used by 1 machine.

Phần II

Xác suất

Chương 13

Các hàm phân phối thông dụng

Phần này có thêm khảo Goodfellow u.a. (2016) và giáo trình xác suất thống kê của thạc sỹ Trần Thiện Khải, đại học Trà Vinh [1]

17/01/2018 Lòng vòng thế nào hôm nay lại tìm được của bạn Đỗ Minh Hải [2], rất hay

13.0.1 Biến rời rạc

Phân phối đều - Discrete Uniform distribution

Là phân phối mà xác suất xuất hiện của các sự kiện là như nhau.
Biến ngẫu nhiên X tuân theo phân phối đều rời rạc

$$X \sim \mathcal{U}(a, b)$$

với tham số $a, b \in \mathbb{Z}; a < b$ là khoảng giá trị của X, đặt $n = b - a + 1$
Ta sẽ có:

Định nghĩa	Giá trị
PMF	$p(x) \mid \frac{1}{n}, \forall x \in [a, b]$
CDF - $F(x; a, b)$	$\frac{x-a+1}{n}, \forall x \in [a, b]$
Kỳ vọng - $E[X]$	$\frac{a+b}{2}$
Phương sai - $Var(X)$	$\frac{n^2-1}{12}$

Ví dụ: Lịch chạy của xe buýt tại một trạm xe buýt như sau: chiếc xe buýt đầu tiên trong ngày sẽ khởi hành từ trạm này vào lúc 7 giờ, cứ sau mỗi 15 phút sẽ có một xe khác đến trạm. Giả sử một hành khách đến trạm trong khoảng thời gian từ 7 giờ đến 7 giờ 30. Tìm xác suất để hành khách này chờ:

a) Ít hơn 5 phút.

b) Ít nhất 12 phút.

Giải

[1] http://www.ctec.tvu.edu.vn/ttkhai/xacsuatthongke_dh.htm
[2] https://dominhhai.github.io/vi/2017/10/prob-com-var

Gọi X là số phút sau 7 giờ mà hành khách đến trạm.

Ta có: $X \sim R[0; 30]$.

a) Hành khách sẽ chờ ít hơn 5 phút nếu đến trạm giữa 7 giờ 10 và 7 giờ 15 hoặc giữa 7 giờ 25 và 7 giờ 30. Do đó xác suất cần tìm là:

$$P(0 < X < 15) + P(25 < X < 30) = \frac{5}{30} + \frac{5}{30} = \frac{1}{3}$$

b) Hành khách chờ ít nhất 12 phút nếu đến trạm giữa 7 giờ và 7 giờ 3 phút hoặc giữa 7 giờ 15 phút và 7 giờ 18 phút. Xác suất cần tìm là:

$$P(0 < X < 3) + P(15 < X < 18) = \frac{3}{30} + \frac{3}{30} = \frac{1}{5}$$

Phân phối Béc-nu-li - Bernoulli distribution

Như đã đề cập về phép thử Béc-nu-li rằng mọi phép thử của nó chỉ cho 2 kết quả duy nhất là A với xác suất p và $\bar{A}$ với xác suất $q = 1 - p$ Biến ngẫu nhiên X tuân theo phân phối Béc-nu-li

$$X \sim B(p)$$

với tham số $p \in \mathbb{R}, 0 \leq p \leq 1$ là xác suất xuất hiện của A tại mỗi phép thử

Định nghĩa		Giá trị
PMF	p(x)	$p(x) \mid p^x(1-p)^{1-x}, x \in \{0, 1\}$
CDF	$F(x; p)$	$\begin{cases} 0 & \text{for } x < 0 \\ 1 - p & \text{for } 0 \leq x < 1 \\ 1 & \text{for } x \geq 1 \end{cases}$
Kỳ vọng	$E[X]$	p
Phương sai	$Var(X)$	$p(1-p)$

Ví dụ

Tham khảo thêm các thuật toán khác tại Hai (2018)

Phần III

Khoa học máy tính

Chương 14

Data Structure and Algorithm

View online http://magizbox.com/training/danda/site/

14.1 Introduction

Algorithms + Data Structures = Programs

In computer science, a data structure is a particular way of organizing data in a computer so that it can be used efficiently. Data structures can implement one or more particular abstract data types (ADT), which specify the operations that can be performed on a data structure and the computional complexity of those operations. In comparison, a data structure is a concrete implementation of the specification provided by an ADT.

In mathematics and computer science, an algorithm is a self-contained step-by-step set of operations to be performed. Algorithms perform calculation, data processing, and/or automated reasoning tasks.

Software engineering is the study of ways in which to create large and complex computer applications and that generally involve many programmers and designers. At the heart of software engineering is with the overall design of the applications and on the creation of a design that is based on the needs and requirements of end users. While software engineering involves the full life cycle of a software project, is includes many different components - specification, requirements gathering, design, verification, coding, testing, quality assurance, user acceptance testing, production, and ongoing maintenance.

Having an in-depth understanding on every component of software engineering is not mandatory, however, it is important to understand that the subject of data structures and algorithms is concerned with the coding phase. The use of data structures and algorithms is the nuts-and-blots used by programmers to store and manipulate data.

This article, along with the other examples in this section focuses on the essentials of data structures and algorithms. Attempts will be made to understand how they work, which structure or algorithm is best in a particular situation in an easy to understand environment.

Data Structures and Algorithms - Defined A data structure is an arrangement of data in a computer's memory or even disk storage. An example of several common data structures are arrays, linked lists, queues, stacks, binary trees, and hash tables. Algorithms, on the other hand, are used to manipulate the data contained in these data structures as in searching and sorting.

Many algorithms apply directly to a specific data structures. When working with certain data structures you need to know how to insert new data, search for a specified item, and deleting a specific item.

Commonly used algorithms include are useful for:

Searching for a particular data item (or record). Sorting the data. There are many ways to sort data. Simple sorting, Advanced sorting Iterating through all the items in a data structure. (Visiting each item in turn so as to display it or perform some other action on these items)

14.1.1 Greedy Algorithm

Greedy Algorithms An algorithm is designed to achieve optimum solution for a given problem. In greedy algorithm approach, decisions are made from the given solution domain. As being greedy, the closest solution that seems to provide an optimum solution is chosen.

Greedy algorithms try to find a localized optimum solution, which may eventually lead to globally optimized solutions. However, generally greedy algorithms do not provide globally optimized solutions.

Counting Coins This problem is to count to a desired value by choosing the least possible coins and the greedy approach forces the algorithm to pick the largest possible coin. If we are provided coins of 1, 2, 5 and 10 and we are asked to count 18 then the greedy procedure will be

Select one 10 coin, the remaining count is 8

Then select one 5 coin, the remaining count is 3

Then select one 2 coin, the remaining count is 1

And finally, the selection of one 1 coins solves the problem

Though, it seems to be working fine, for this count we need to pick only 4 coins. But if we slightly change the problem then the same approach may not be able to produce the same optimum result.

For the currency system, where we have coins of 1, 7, 10 value, counting coins for value 18 will be absolutely optimum but for count like 15, it may use more coins than necessary. For example, the greedy approach will use 10 + 1 + 1 + 1 + 1 + 1, total 6 coins. Whereas the same problem could be solved by using only 3 coins (7 + 7 + 1)

Hence, we may conclude that the greedy approach picks an immediate optimized solution and may fail where global optimization is a major concern.

Examples Most networking algorithms use the greedy approach. Here is a list of few of them

Travelling Salesman Problem Prim's Minimal Spanning Tree Algorithm Kruskal's Minimal Spanning Tree Algorithm Dijkstra's Minimal Spanning Tree Algorithm Graph - Map Coloring Graph - Vertex Cover Knapsack Problem Job Scheduling Problem

14.1.2 Divide and Conquer

In divide and conquer approach, the problem in hand, is divided into smaller sub-problems and then each problem is solved independently. When we keep on dividing the subproblems into even smaller sub-problems, we may eventually reach a stage where no more division is possible. Those "atomic" smallest possible sub-problem (fractions) are solved. The solution of all sub-problems is finally merged in order to obtain the solution of an original problem.

Broadly, we can understand divide-and-conquer approach in a three-step process.

Divide/Break This step involves breaking the problem into smaller sub-problems. Sub-problems should represent a part of the original problem. This step generally takes a recursive approach to divide the problem until no sub-problem is further divisible. At this stage, sub-problems become atomic in nature but still represent some part of the actual problem.

Conquer/Solve This step receives a lot of smaller sub-problems to be solved. Generally, at this level, the problems are considered 'solved' on their own.

Merge/Combine When the smaller sub-problems are solved, this stage recursively combines them until they formulate a solution of the original problem. This algorithmic approach works recursively and conquer merge steps works so close that they appear as one.

Examples The following computer algorithms are based on divide-and-conquer programming approach

Merge Sort Quick Sort Binary Search Strassen's Matrix Multiplication Closest pair (points) There are various ways available to solve any computer problem, but the mentioned are a good example of divide and conquer approach.

14.1.3 Dynamic Programming

Dynamic programming approach is similar to divide and conquer in breaking down the problem into smaller and yet smaller possible sub-problems. But unlike, divide and conquer, these sub-problems are not solved independently. Rather, results of these smaller sub-problems are remembered and used for similar or overlapping sub-problems.

Dynamic programming is used where we have problems, which can be divided into similar sub-problems, so that their results can be re-used. Mostly, these algorithms are used for optimization. Before solving the in-hand sub-problem, dynamic algorithm will try to examine the results of the previously solved sub-problems. The solutions of sub-problems are combined in order to achieve the best solution.

So we can say that

The problem should be able to be divided into smaller overlapping sub-problem. An optimum solution can be achieved by using an optimum solution of smaller sub-problems. Dynamic algorithms use memorization. Comparison In contrast to greedy algorithms, where local optimization is addressed, dynamic algorithms are motivated for an overall optimization of the problem.

In contrast to divide and conquer algorithms, where solutions are combined to achieve an overall solution, dynamic algorithms use the output of a smaller sub-problem and then try to optimize a bigger sub-problem. Dynamic algorithms use memorization to remember the output of already solved sub-problems.

Example The following computer problems can be solved using dynamic programming approach

Fibonacci number series Knapsack problem Tower of Hanoi All pair shortest path by Floyd-Warshall Shortest path by Dijkstra Project scheduling Dynamic programming can be used in both top-down and bottom-up manner. And of course, most of the times, referring to the previous solution output is cheaper than recomputing in terms of CPU cycles.

14.1.4 7 Steps to Solve Algorithm Problems

Today, I viewed the video "7 Steps to Solve Algorithm Problems" by Gayle Laakmann McDowell - the author of Cracking the Coding Interview book. In this video, Gayle describe her method for solve algorithms problems which consists 7 steps: listen carefully, example, brute force, optimize, walk through your algorithms, code and test. In this article, I will summary these steps base on what I learned from this video.

Step 1: Listen carefully Every single detail in a question is necessary to solve it.

The first step is to listen carefully to the problem. So, generally speaking every single detail in a question is necessary to solve that problem - either to solve it all or to solve it optimally. So if there's some detail you haven't used in the question in your algorithm so far think about how you can put that to use because it might be necessary to solve the problem optimally.

Let me give you an example.

You have two arrays, sorted and distinct How did you find the number of elements in common between the two arrays? A lot of people solve this problem and they'll get kind of stuck for awhile and what they'll do is they'll be solving the problem and they'll know the arrays are sorted but they haven't actually used the fact that it's sorted.

This sorting detail - it's not necessary just to find an algorithm but it is necessary to solve the problem optimally.

So remember every single detail in the problem and make sure you use it.

Step 2: Example Make example big, no special cases

The second piece is to come up with a good example, so the last problem that I gave two arrays sorted and distinct compute the number of elements in common, most people's examples look like this.

too small and special case A: 1, 5, 15, 20 B: 2, 5, 13, 30 Yes technically if it's a problem but it's not very useful.

As soon as you glance at this example you notice that there's only one element common and you know exactly what it is and it's obvious because this example is so small and it's actually kind of a special case.

A better example is something like this

larger and avoid special cases A: 1, 5, 15, 20, 30, 37 B: 2, 5, 13, 30, 32, 35, 37, 42 It's much larger and you've avoided some special cases. One of the easiest ways of improving your performance on algorithm questions is just make your examples larger and really avoid special cases.

Step 3: Brute force Better to have a brute force than nothing at all

The third step is to come up with a brute force algorithm. Now I'm not saying you need to go out of your way to come up with something slow, I'm really just saying, hey if the first thing you have is only something really really slow and terrible that's okay. It is so much better to start off with something slow then to start off with nothing at all. So it's fine if your first algorithm is slow and terrible whatever. However, and this is very very very important, I'm not saying to code the brute force. I'm saying just state your brute force algorithm, state its runtime, and then immediately go to optimizing.

A good chunk of the time on algorithm interview question will often be spent on optimizations. So that's step 4 and spend some good time on it.

Step 4: Optimize The fourth step is optimize and spend some good time on it.

Step 5: Walk through your algorithms Know exactly what you're going to do before coding

what variables data structures? how, why, why do they change? what is the structure of your code Then once you have an optimal algorithm or you're ready to start coding take a step back and just make sure you know exactly what you're going to do in your code.

So many people code prematurely when they aren't really really comfortable with what they're about to do and it ends in disaster. An eighty percent understanding of what you're about to write is really not enough for a whiteboard especially. So take a moment and walk through your algorithm and make sure you know exactly what you're about to do.

Step 6: Code Use space wisely, coding style matters, modularize

Step 6 is to start coding and I'm gonna go into this in a bit of detail. So a couple things to keep in mind particularly when you're coding on a whiteboard. The first couple tips are kind of whiteboard specific but try to write your lines straight. I'm not gonna be judging you on your handwriting and things like that but when people start writing their lines and sharp angles they start to lose track over whether this if statement under this for loop or not. The second thing is use your board space wisely. If you don't need stuff up on the board anymore just erase it. Try to write in this top left corner etc.

Basically give yourself as much space as you possibly can to write your code. If you do run out of space though, it's ok to use arrows, that's fine, I'm really not gonna be judging you on this kind of stuff. So more important things.

Coding style matters (consistent braces, consistent variable naming, consistence spaces, descriptive variables)

Coding style matters even on a whiteboard but on a computer as well, so that means things like braces, naming conventions, or using camel case or underscores, things like that. Those kind of style things absolutely matter. I'm not that concerned over which style you pick, I don't care if you write braces on the same line or the next line but I do care a lot that you have a style and you stick to it. So be consistent in your style. When it comes to variable names, yeah I know it's an annoying to write long variable names on a whiteboard but descriptive variable names are important to good style. So one compromise here is write the good descriptive variable name first and then just ask your interviewer, hey is it okay if I abbreviate this the next time. So that'll be a nice little compromise - you'd show that you care about good variable names but you also don't waste a lot of time.

Modularize (before. not after)

Last thing I want to talk about is modularization. Modularize your code up front and just any little conceptual chunks of code, push that off to another function. So suppose you have three steps in your algorithm - process the first string, process the second string, and then compare the results. Don't start writing these for loops that walk through each string in the very beginning. Instead write this overall function that wraps these three steps. So step one, step two, step three, and then start drilling in and going into all the details there. Remember any conceptual chunks of code push those off to other functions, don't write them in line.

Step 7: Test Analyse: think about each line, double check things that look weired/risky (for-loop that decrement, math)

Use test cases (smaller test-cases first (faster to run, you will problably be more through, edge cases, big test cases)

Then once you're done with the coding you have to start testing your code. One of the mistakes a lot of people do here is they take their big example from step 2 and throw that in as a test case. The problem with that is it's very large so it will take you a long time to run through but also you just used that to develop your code, so if here's an oversight there, the problem will probably repeat itself here.

What's a better step to do, what's a better process to do, is just walk through your code line by line and just think about each line up front not with the test case but just consider, is it really doing the right thing?

Double check anything that looks weird, so for loops that decrement instead of increment and any math at all is a really common place for errors. Just think, look at your code analytically and think what are the most likely places for errors to be and double-check those.

Start with small rather than big

Then once you start with actual test cases start with small test cases rather than big ones. Small test cases work pretty much as effectively as big test cases but they are so much faster to run through, and in fact because they're faster people tend to be much more thorough so you're much more likely to actually find bugs with small test cases than big test cases. So start with small test cases then go in to edge cases after that and then if you have time maybe throw in some big test cases. A couple last techniques with testing. The first one is make sure that when you're testing you're really thinking about what you're doing. A lot of people when they're testing they're just walking through their code almost like they're a bot, and they only look to see if things made sense at the very end when they look at their output. It's much better to really think as you're testing, this way you find the bug as soon as it happens rather than six lines later at the very bottom.

Test your code not your algorithm

The second thing is when you're testing make sure that you're actually testing your code and not your algorithm. An amazing number of people will just take their example and like just walk through it again as though they're just walking through their algorithm but they're never even looking at their code, they're not looking at the exact calculations their code actually did. So make sure that you're really testing your code.

Find bugs

Then the last thing is when you find in a bug, don't panic. Just really think about what caused the bug. A lot of times people will panic and just try to make the first fix that fixes it for that output but they haven't really given it some thought and then they're in a much worse position because if you make the wrong fix to your code, the thing that just fixed the output but didn't fix a real bug you've not fixed the actual bug, you've made your code more complex, and you potentially introduced a brand new bug and you're in a much worse position. It's much better to just when you find the bug, it's ok, it's not that big of a deal to have a bug it's very normal just really think through what the actual bug, where the actual plug came from.

Remember

think as you test (don't be a bot) test your code, not your algorithm think before you fix bugs. Don't panic! (wrong fixes are worse than no fix) Suggested Reading 7 Steps to Solve Algorithm Problems. Gayle Laakmann McDowell

14.2 Data Structures

14.2.1 Array

Arrays An array is an aggregate data structure that is designed to store a group of objects of the same or different types. Arrays can hold primitives as well as references. The array is the most efficient data structure for storing and accessing a sequence of objects.

Here is the list of most important array features you must know (i.e. be able to program)

copying and cloning insertion and deletion searching and sorting You already know that the Java language has only two data types, primitives and references. Which one is an array? Is it primitive? An array is not a primitive data type - it has a field (and only one), called length. Formally speaking, an array is a reference type, though you cannot find such a class in the Java APIs. Therefore, you deal with arrays as you deal with references. One of the major diffeences between refeences and primituives is that you cannot copy arrays by assigning one to another:

int[] a = 9, 5, 4; int[] b = a; The assignment operator creates an alias to the object, like in the picture below

Since these two references a and b refer to the same object, comparing them with the double equal sign "==" will always return true. In the next code example,

int [] a = 1,2,3; int [] b = 1,2,3; a and b refer to two different objects (though with identical contents). Comparing them with the double equal sign will return false. How would you compare two objects with identical contents? In short, using the equals method. For array comparison, the Java APIs provides the Arrays class.

The Arrays class The java.util.Arrays class is a convenience class for various array manipulations, like comparison, searching, printing, sorting and others. Basically, this class is a set of static methods that are all useful for working with arrays. The code below demonstrates a proper invocation of equals:

int[] a = 1,2,3; int[] b = 1,2,3; if(Arrays.equals(a, b)) System.out.println("arrays with identical contents"); Another commonly used method is toString() which takes care of of printing

int[] a = 1,2,3; System.out.println(Arrays.toString(a)); Here is the example of sorting

int[] a = 3,2,1; Arrays.sort(a); System.out.println(Arrays.toString(a)); In addition to that, the class has other utility methods for supporting operations over multidimensional arrays.

Copying arrays There are four ways to copy arrays

using a loop structure using Arrays.copyOf() using System.arraycopy() using clone() The first way is very well known to you

int[] a = 1, 2, 3; int[] b = new int[a.length]; for(int i= 0; i < a.length; i++) b[i] = a[i]; The next choice is to use Arrays.copyOf()

int[] a = 1, 2, 3; int[] b = Arrays.copyOf(a, a.length); The second parameter specifies the length of the new array, which could either less or equal or bigger than the original length.

The most efficient copying data between arrays is provided by System.arraycopy() method. The method requires five arguments. Here is its signature

public static void arraycopy(Object source, int srcIndex, Object destination, int destIndex, int length) The method copies length elements from a source array starting with the index srcIndex to a new array destination at the index destIndex.The above code example can be rewritten as it follows

int[] a = 1, 2, 3; int[] b = new int[a.length]; System.arraycopy(a, 0, b, 0, 3) And the last copying choice is the use of cloning. Cloning involves creating a new array of the same size and type and copying all the old elements into the new array. The clone() method is defined in the Object class and its invocation is demonstrated by this code segment

int[] a = 1, 2, 3; int[] b = (int[]) a.clone(); Note, that casting (int[]) is the must.

Examine the code in ArrayCopyPrimitives.java for further details.

Insertion and Deletion Arrays in Java have no methods and only one immutable field length. Once an array is created, its length is fixed and cannot be changed. What do you do to resize the array? You allocate the array with a different size and copy the contents of the old array to the new array. This code example demonstrates deletion from an array of primitives

public char[] delete(char[] data, int pos) if(pos >= 0 pos < data.length) char[] tmp = new char[data.length-1]; System.arraycopy(data, 0, tmp, 0, pos); System.arraycopy(data, pos+1, tmp, pos, data.length-pos-1); return tmp; else return data; The first arraycopy copies the elements from index 0 to index pos-1, the second arraycopy copies the elements from index pos+1 to data.length.

Examine the code in ArrayDemo.java for further details.

The ArrayList class The java.util.ArrayList class supports an idea of a dynamic array - an array that grows and shrinks on demand to accomodate the number of elements in the array. Below is a list of commonly used methods

add(object) - adds to the end add(index, object) - inserts at the index set(index, object) - replaces at the index get(index) - returns the element at that index remove(index) - deletes the element at that index size() - returns the number of elements The following code example will give you a heads up into how some of them are used.

/* ADD */ ArrayList<Integer> num = new ArrayList<Integer>(); for(int i = 0; i < 10; i++) num.add(i); System.out.println(num);

/* REMOVE even integers */ for(int i = 0; i < num.size(); i++) if(num.get(i)System.out.println(num); Copying arrays of objects This topic is more complex for understanding.. Let us start with a simple loop structure

Object[] obj1 = new Integer(10), new StringBuffer("foobar"), new Double(12.95); Object[] obj2 = new Object[obj1.length]; for(int i = 0; i ‹ obj1.length; i++) obj2[i] = obj1[i]; At the first glance we might think that all data is copied.

In reality, the internal data is shared between two arrays. The figure below illustrates the inner structure

The assignment operator obj2[i] = obj1[i] is a crucial part of understanding the concept. You cannot copy references by assigning one to another. The assignment creates an alias rather than a copy. Let us trace down changes in the above picture after execution the following statements

obj1[0] = new Integer(5);

and ((StringBuffer) obj1[1]).append('s');

As you see, obj1[0] and obj2[0] now refer to different objects. However, obj1[1] and obj2[1] refer to the same object (which is "foobars"). Since both arrays shares the data, you must be quite careful when you modify your data, because it might lead to unexpected effects.

The same behavior will take place again, if we use Arrays.copuyOf(), System.arraycopy() and clone(). Examine the code example ArrayCopyReferences.java for further details.

Multi-dimensional arrays In many practical application there is a need to use two- or multi-dimensional arrays. A two-dimensional array can be thought of as a table of rows and columns. This creates a table of 2 rows and 4 columns:

int[][] ar1 = new int[2][4]; You can create and initialize an array by using nested curcly braces. For example, this creates a table of 3 rows and 2 columns:

int[][] ar2 = 1,2,3,4,5,6; Generally speaking, a two-dimensional array is not exactly a table - each row in such array can have a different length. Consider this code fragment

Object[][] obj = new Integer(1),new Integer(2), new Integer(10), "bozo", new Double(1.95); The accompanying picture sheds a bit of light on internal representation

From the picture you clearly see that a two-dimensional array in Java is an array of arrays. The array obj has two elements obj[0] and obj[1] that are arrays of length 2 and 3 respectively.

Cloning 2D arrays The procedure is even more confusing and less expected. Consider the following code segment

Object[][] obj = new Integer(1),new Integer(2), new Integer(10), "bozo", new Double(1.95);

Object[][] twin = (Object[][]) obj.clone(); The procedure of clonig 2d arrays is virtually the same as cloning an array of references. Unfortunately, built-in clone() method does not actualy clone each row, but rather creates references to them Here is a graphical interpretation of the above code

Let us change the value of obj[1][1]

obj[1][1] = "xyz"; This assignment effects the value of twin[1][1] as well

Such a copy is called a "shallow" copy. The default behavior of clone() is to return a shallow copy of the object. If we want a "deep" copy instead, we must provide our own implementation by overriding Object's clone() method.

The idea of a "deep" copy is simple - it makes a distinct copy of each of the object's fields, recursing through the entire object. A deep copy is thus a completely separate object from the original; changes to it don't affect the original, and vise versa. In relevance to the above code, here is a deep clone graphically

Further, making a complete deep copy is not always needed. Consider an array of immutable objects. As we know, immutable objects cannot be modified, allowing clients to share the same instance without interfering with each other. In this case there is no need to clone them, which leads to the following picture

Always in this course we will create data structures of immutable objets, therefore implementing the clone method will require copying a structure (a shape) and sharing its internal data. We will discuss these issues later on in the course.

Challenges "Arrays: Left Rotation". hackerrank. 2016 References "Array Data Structure". Victor S.Adamchik, CMU. 2009

14.2.2 Linked List

A linked list is a sequence of data structures, which are connected together via links.

Linked List is a sequence of links which contains items. Each link contains a connection to another link. Linked list is the second most-used data structure after array. Following are the important terms to understand the concept of Linked List.

Link Each link of a linked list can store a data called an element. Next Each link of a linked list contains a link to the next link called Next. LinkedList A Linked List contains the connection link to the first link called First. Representation Linked list can be visualized as a chain of nodes, where every node points to the next node.

As per the above illustration, following are the important points to be considered.

Linked List contains a link element called first. Each link carries a data field(s) and a link field called next. Each link is linked with its next link using its next link. Last link carries a link as null to mark the end of the list. Types of Linked List Following are the various types of linked list.

Simple Linked List Item navigation is forward only. Doubly Linked List Items can be navigated forward and backward. Circular Linked List Last item contains link of the first element as next and the first element has a link to the last element as previous. Basic Operations Following are the basic operations supported by a list.

Insertion Adds an element at the beginning of the list. Deletion Deletes an element at the beginning of the list. Display Displays the complete list. Search Searches an element using the given key. Delete Deletes an element using the given key. Insertion Operation Adding a new node in linked list is a more than one step activity. We shall learn this with diagrams here. First, create a node using the same structure and find the location where it has to be inserted.

Imagine that we are inserting a node B (NewNode), between A (LeftNode) and C (RightNode). Then point B.next to C

NewNode.next $>$ RightNode; It should look like this

Now, the next node at the left should point to the new node.

LeftNode.next $>$ NewNode;

This will put the new node in the middle of the two. The new list should look like this

Similar steps should be taken if the node is being inserted at the beginning of the list. While inserting it at the end, the second last node of the list should point to the new node and the new node will point to NULL.

Deletion Operation Deletion is also a more than one step process. We shall learn with pictorial representation. First, locate the target node to be removed, by using searching algorithms.

The left (previous) node of the target node now should point to the next node of the target node

LeftNode.next $>$ TargetNode.next;

This will remove the link that was pointing to the target node. Now, using the following code, we will remove what the target node is pointing at.

TargetNode.next $>$ NULL;

We need to use the deleted node. We can keep that in memory otherwise we can simply deallocate memory and wipe off the target node completely.

Reverse Operation This operation is a thorough one. We need to make the last node to be pointed by the head node and reverse the whole linked list.

First, we traverse to the end of the list. It should be pointing to NULL. Now, we shall make it point to its previous node

We have to make sure that the last node is not the lost node. So we'll have some temp node, which looks like the head node pointing to the last node. Now, we shall make all left side nodes point to their previous nodes one by one.

Except the node (first node) pointed by the head node, all nodes should point to their predecessor, making them their new successor. The first node will point to NULL.

We'll make the head node point to the new first node by using the temp node.

The linked list is now reversed.

14.2.3 Stack and Queue

An array is a random access data structure, where each element can be accessed directly and in constant time. A typical illustration of random access is a book - each page of the book can be open independently of others. Random access is critical to many algorithms, for example binary search.

A linked list is a sequential access data structure, where each element can be accesed only in particular order. A typical illustration of sequential access is a roll of paper or tape - all prior material must be unrolled in order to get to data you want.

In this note we consider a subcase of sequential data structures, so-called limited access data sturctures.

Stacks A stack is a container of objects that are inserted and removed according to the last-in first-out (LIFO) principle. In the pushdown stacks only two operations are allowed: push the item into the stack, and pop the item out of the stack. A stack is a limited access data structure - elements can be added and removed from the stack only at the top. push adds an item to the top of the stack, pop removes the item from the top. A helpful analogy is to think of a stack of books; you can remove only the top book, also you can add a new book on the top. A stack is a recursive data structure. Here is a structural definition of a Stack:

a stack is either empty or it consistes of a top and the rest which is a stack;

Applications The simplest application of a stack is to reverse a word. You push a given word to stack - letter by letter - and then pop letters from the stack. Another application is an "undo" mechanism in text editors; this operation is accomplished by keeping all text changes in a stack. Backtracking. This is a process when you need to access the most recent data element in a series of elements. Think of a labyrinth or maze - how do you find a way from an entrance to an exit? Once you reach a dead end, you must backtrack. But backtrack to where? to the previous choice point. Therefore, at each choice point you store on a stack all possible choices. Then backtracking simply means popping a next choice from the stack.

Language processing: space for parameters and local variables is created internally using a stack. compiler's syntax check for matching braces is implemented by using stack. support for recursion Implementation In the standard library of classes, the data type stack is an adapter class, meaning that a stack is built on top of other data structures. The underlying structure for a stack could be an array, a vector, an ArrayList, a linked list, or any other collection. Regardless of the type of the underlying data structure, a Stack must implement the same functionality. This is achieved by providing a unique interface:

public interface StackInterface<AnyType> public void push(AnyType e);

public AnyType pop();

public AnyType peek();

public boolean isEmpty(); The following picture demonstrates the idea of implementation by composition.

Another implementation requirement (in addition to the above interface) is that all stack operations must run in constant time O(1). Constant time means that there is some constant k such that an operation takes k nanoseconds of computational time regardless of the stack size.

Array-based implementation

In an array-based implementation we maintain the following fields: an array A of a default size (1), the variable top that refers to the top element in the stack and the capacity that refers to the array size. The variable top changes from -1 to capacity - 1. We say that a stack is empty when top = -1, and the stack is full when top = capacity-1. In a fixed-size stack abstraction, the capacity stays unchanged, therefore when top reaches capacity, the stack object throws an exception. See ArrayStack.java for a complete implementation of the stack class.

In a dynamic stack abstraction when top reaches capacity, we double up the stack siz

Linked List-based implementation

Linked List-based implementation provides the best (from the efficiency point of view) dynamic stack implementation. See ListStack.java for a complete implementation of the stack class.

Queues A queue is a container of objects (a linear collection) that are inserted and removed according to the first-in first-out (FIFO) principle. An excellent example of a queue is a line of students in the food court of the UC. New additions to a line made to the back of the queue, while removal (or serving) happens in the front. In the queue only two operations are allowed enqueue and dequeue. Enqueue means to insert an item into the back of the queue, dequeue means removing the front item. The picture demonstrates the FIFO access. The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

Implementation In the standard library of classes, the data type queue is an adapter class, meaning that a queue is built on top of other data structures. The underlying structure for a queue could be an array, a Vector, an ArrayList, a LinkedList, or any other collection. Regardless of the type of the underlying data structure, a queue must implement the same functionality. This is achieved by providing a unique interface.

interface QueueInterface‹AnyType› public boolean isEmpty();

public AnyType getFront();

public AnyType dequeue();

public void enqueue(AnyType e);

public void clear(); Each of the above basic operations must run at constant time O(1). The following picture demonstrates the idea of implementation by composition.

Circular Queue Given an array A of a default size (1) with two references back and front, originally set to -1 and 0 respectively. Each time we insert (enqueue) a new item, we increase the back index; when we remove (dequeue) an item - we increase the front index. Here is a picture that illustrates the model after a few steps:

As you see from the picture, the queue logically moves in the array from left to right. After several moves back reaches the end, leaving no space for adding new elements

However, there is a free space before the front index. We shall use that space for enqueueing new items, i.e. the next entry will be stored at index 0, then 1, until front. Such a model is called a wrap around queue or a circular queue

Finally, when back reaches front, the queue is full. There are two choices to handle a full queue:a) throw an exception; b) double the array size.

The circular queue implementation is done by using the modulo operator (denoted

See ArrayQueue.java for a complete implementation of a circular queue.

Applications The simplest two search techniques are known as Depth-First Search(DFS) and Breadth-First Search (BFS). These two searches are described by looking at how the search tree (representing all the possible paths from the start) will be traversed.

Deapth-First Search with a Stack In depth-first search we go down a path until we get to a dead end; then we backtrack or back up (by popping a stack) to get an alternative path.

Create a stack Create a new choice point Push the choice point onto the stack while (not found and stack is not empty) Pop the stack Find all possible choices after the last one tried Push these choices onto the stack Return Breadth-First Search with a Queue In breadth-first search we explore all the nearest possibilities by finding all possible successors and enqueue them to a queue.

Create a queue Create a new choice point Enqueue the choice point onto the queue while (not found and queue is not empty) Dequeue the queue Find all possible choices after the last one tried Enqueue these choices onto the queue Return We will see more on search techniques later in the course.

Arithmetic Expression Evaluation An important application of stacks is in parsing. For example, a compiler must parse arithmetic expressions written using infix notation:

1 + ((2 + 3) * 4 + 5)*6 We break the problem of parsing infix expressions into two stages. First, we convert from infix to a different representation

called postfix. Then we parse the postfix expression, which is a somewhat easier problem than directly parsing infix.

Converting from Infix to Postfix. Typically, we deal with expressions in infix notation

2 + 5 where the operators (e.g. +, *) are written between the operands (e.q, 2 and 5). Writing the operators after the operands gives a postfix expression 2 and 5 are called operands, and the '+' is operator. The above arithmetic expression is called infix, since the operator is in between operands. The expression

2 5 + Writing the operators before the operands gives a prefix expression

+2 5 Suppose you want to compute the cost of your shopping trip. To do so, you add a list of numbers and multiply them by the local sales tax (7.25

70 + 150 * 1.0725 Depending on the calculator, the answer would be either 235.95 or 230.875. To avoid this confusion we shall use a postfix notation

70 150 + 1.0725 * Postfix has the nice property that parentheses are unnecessary.

Now, we describe how to convert from infix to postfix.

Read in the tokens one at a time If a token is an integer, write it into the output If a token is an operator, push it to the stack, if the stack is empty. If the stack is not empty, you pop entries with higher or equal priority and only then you push that 1. token to the stack. If a token is a left parentheses '(', push it to the stack If a token is a right parentheses ')', you pop entries until you meet '('. When you finish reading the string, you pop up all tokens which are left there. Arithmetic precedence is in increasing order: '+', '-', '*', '/'; Example. Suppose we have an infix expression:2+(4+3*2+1)/3. We read the string by characters.

'2' - send to the output. '+' - push on the stack. '(' - push on the stack. '4' - send to the output. '+' - push on the stack. '3' - send to the output. '*' - push on the stack. '2' - send to the output. Evaluating a Postfix Expression. We describe how to parse and evaluate a postfix expression.

We read the tokens in one at a time. If it is an integer, push it on the stack If it is a binary operator, pop the top two elements from the stack, apply the operator, and push the result back on the stack. Consider the following postfix expression

5 9 3 + 4 2 * * 7 + * Here is a chain of operations

Stack Operations Output ———————————— push(5); 5 push(9); 5 9 push(3); 5 9 3 push(pop() + pop()) 5 12 push(4); 5 12 4 push(2); 5 12 4 2 push(pop() * pop()) 5 12 8 push(pop() * pop()) 5 96 push(7) 5 96 7 push(pop() + pop()) 5 103 push(pop() * pop()) 515 Note, that division is not a commutative operation, so 2/3 is not the same as 3/2.

Challenges Stacks: Balanced Brackets Queues: A Tale of Two Stacks References "Stacks and Queues". Victor S.Adamchik, CMU. 2009

14.2.4 Tree

Binary Tree Fundamentally, a binary tree is composed of nodes connected by edges (with further restrictions discussed below). Some binary tree, tt, is either empty or consists of a single root element with two distinct binary tree child elements known as the left subtree and the right subtree of tt. As the name binary suggests, a node in a binary tree has a maximum of 22 children.

The following diagrams depict two different binary trees:

Here are the basic facts and terms to know about binary trees:

The convention for binary tree diagrams is that the root is at the top, and the subtrees branch down from it. A node's left and right subtrees are referred to as children, and that node can be referred to as the parent of those subtrees. A non-root node with no children is called a leaf. Some node aa is an ancestor of some node bb if bb is located in a left or right subtree whose root node is aa. This means that the root node of binary tree tt is the ancestor of all other nodes in the tree. If some node aa is an ancestor of some node bb, then the path from aa to bb is the sequence of nodes starting with aa, moving down the ancestral chain of children, and ending with bb. The depth (or level) of some node aa is its distance (i.e., number of edges) from the tree's root node. Simply put, the height of a tree is the number of edges between the root node and its furthest leaf. More technically put, it's 1+max(height(leftSubtree),height(rightSubtree))1+max(height(leftSubtree),height(rightSubtree))) (i.e., one more than the maximum of the heights of its left and right subtrees). Any node has a height of 11, and the height of an empty subtree is 11. Because the height of each node is 1+1+ the maximum height of its subtrees and an empty subtree's height is 11, the height of a single-element tree or leaf node is 00. Let's apply some of the terms we learned above to the binary tree on the right:

The root node is AA.

The respective left and right children of AA are BB and EE. The left child of BB is CC. The respective left and right children of EE are FF and DD.

Nodes CC, FF, and DD are leaves (i.e., each node is a leaf).

The root is the ancestor of all other nodes, BB is an ancestor of CC, and EE is an ancestor of FF and DD.

The path between AA and CC is ABCABC. The path between AA and FF is AEFAEF. The path between AA and DD is A EDED.

The depth of root node AA is 00. The depth of nodes BB and EE is 11. The depth of nodes CC, FF, and DD, is 22.

The height of the tree, height(t)height(t), is 22. We calculate this recursively as height(t)=1+(max(height(root.leftChild),height(root.rightChild)))height(t)=1+(max(height(root.leftChil

Because this is long and complicated when expanded, we'll break it down using an image of a slightly simpler version of tt whose height is still 22:

Binary Search Tree A Binary Search Tree (BST), tt, is a binary tree that is either empty or satisfies the following three conditions:

Each element in the left subtree of tt is less than or equal to the root element of tt (i.e., max(leftTree(t).value)t.valuemax(leftTree(t).value)t.valuemax(leftTree(t).value)t.valuemax(leftTre

Each element in the right subtree of tt is greater than the root element of tt (i.e., max(rightTree(t).value)t.valuemax(rightTree(t).value)t.valuemax(rightTree(t).value)t.valuemax(rightTr

Both leftTree(t)leftTree(t) and rightTree(t)rightTree(t) are BSTs.

You can essentially think of it as a regular binary tree where for each node parent having a leftChildleftChild and rightChildrightChild, leftChild.valueparent.valuerightChild.valueleftCh In the first diagram at the top of this article, the binary tree of integers on the left side is a binary search tree.

Advantages and Drawbacks Searching for elements is very fast. We know that each node has a maximum of two children and we know that the items are always in the left subtree and the » items are always in the right subtree. To search for an element, we simply need to compare the value we want against the value stored in the root node of the current subtree and work our way down the appropriate child subtrees until we either find the value we're looking for or we hit null (i.e., an empty subtree) which indicates the item is not in the BST. Inserting or searching for a node in a balanced tree is O(logn)O(logn) because you're discarding half of the possible values each time you go left or right.

It can easily become unbalanced. Depending on the insertion order, the tree can very easily become unbalanced (which makes for longer search times). For example, if we create a new tree where the sequence of inserted nodes is 213456213456, we end up with the following unbalanced tree:

Observe that the root's left subtree only has one node, whereas the root's right subtree has four nodes. For this reason, inserting or searching for a node in an unbalanced tree is O(n)O(n).

Challenges "Trees: Is This a Binary Search Tree?". hackerrank. 2016 References "Binary Trees and Binary Search Trees". AllisonP, hackerrank. 2016

14.2.5 Binary Search Tree

A Binary Search Tree (BST) is a tree in which all the nodes follow the below-mentioned properties

The left sub-tree of a node has a key less than or equal to its parent node's key. The right sub-tree of a node has a key greater than to its parent node's key. Thus, BST divides all its sub-trees into two segments; the left sub-tree and the right sub-tree and can be defined as

left$_s ubtree(keys) node(key) right_s ubtree(keys) RepresentationBSTisacollectionofnodesarrangedinaway$

Following is a pictorial representation of BST

We observe that the root node key (27) has all less-valued keys on the left sub-tree and the higher valued keys on the right sub-tree.

Basic Operations Following are the basic operations of a tree

Search Searches an element in a tree. Insert Inserts an element in a tree. Pre-order Traversal Traverses a tree in a pre-order manner. In-order Traversal Traverses a tree in an in-order manner. Post-order Traversal Traverses a tree in a post-order manner. Node Define a node having some data, references to its left and right child nodes.

struct node int data; struct node *leftChild; struct node *rightChild; ; Search Operation Whenever an element is to be searched, start searching from the root node. Then if the data is less than the key value, search for the element in the left subtree. Otherwise, search for the element in the right subtree. Follow the same algorithm for each node.

Algorithm

struct node* search(int data) struct node *current = root; printf("Visiting elements: ");

while(current->data != data)

if(current != NULL) printf("

//go to left tree if(current->data > data) current = current->leftChild; //else go to right tree else current = current->rightChild;

//not found if(current == NULL) return NULL; return current; Insert Operation Whenever an element is to be inserted, first locate its proper location. Start searching from the root node, then if the data is less than the key value, search for the empty location in the left subtree and insert the data. Otherwise, search for the empty location in the right subtree and insert the data.

Algorithm

void insert(int data) struct node *tempNode = (struct node*) malloc(sizeof(struct node)); struct node *current; struct node *parent;

tempNode->data = data; tempNode->leftChild = NULL; tempNode->rightChild = NULL;

//if tree is empty if(root == NULL) root = tempNode; else current = root; parent = NULL;

while(1) parent = current;

//go to left of the tree if(data < parent->data) current = current->leftChild; //insert to the left

if(current == NULL) parent->leftChild = tempNode; return; //go to right of the tree else current = current->rightChild;

//insert to the right if(current == NULL) parent->rightChild = tempNode; return;

14.3 Heaps

A heap is just what it sounds like — a pile of values organized into a binary tree-like structure adhering to some ordering property. When we add elements to a heap, we fill this tree-like structure from left to right, level by level. This makes heaps really easy to implement in an array, where the value for some index ii's left child is located at index 2i+12i+1 and the value for its right child is at index 2i+22i+2 (using zero-indexing). Here are the two most fundamental heap operations:

add: Insert an element into the heap. You may also see this referred to as push. poll: Retrieve and remove the root element of the heap. You may also see this referred to as pop. Max Heap This type heap orders the maximum value at the root.

When we add the values 12341234 to a Max heap, it looks like this:

When we poll the same Max heap until it's empty, it looks like this:

Min Heap This type of heap orders the minimum value at the root.

When we add the values 12341234 to a Min heap, it looks like this:

When we poll the same Min heap until it's empty, it looks like this:

Applications The heap data structure has many applications.

Heapsort: One of the best sorting methods being in-place and with no quadratic worst-case scenarios. Selection algorithms: A heap allows access to the min or max element in constant time, and other selections (such as median or kth-element) can be done in sub-linear time on data that is in a heap. Graph algorithms: By using heaps as internal traversal data structures, run time will be reduced by polynomial order. Examples of such problems are Prim's minimal-spanning-tree algorithm and Dijkstra's shortest-path algorithm. Priority Queue: A priority queue is an abstract concept like "a list" or "a map"; just as a list can be implemented with a linked list or an array, a priority queue can be implemented with a heap or a variety of other methods. Order statistics: The Heap data structure can be used to efficiently find the kth smallest (or largest) element in an array. Challenges "Heaps: Find the Running Median". hackerrank. 2016 References "Heaps". AllisonP, hackerrank. 2016 "Heap (data structure)". wikipedia. 2016

14.4 Sort

14.4.1 Introduction

Sorting refers to arranging data in a particular format. Sorting algorithm specifies the way to arrange data in a particular order. Most common orders are in numerical or lexicographical order.

The importance of sorting lies in the fact that data searching can be optimized to a very high level, if data is stored in a sorted manner. Sorting is also used to represent data in more readable formats. Following are some of the examples of sorting in real-life scenarios

Telephone Directory The telephone directory stores the telephone numbers of people sorted by their names, so that the names can be searched easily. Dictionary The dictionary stores words in an alphabetical order so that searching of any word becomes easy. In-place Sorting and Not-in-place Sorting Sorting algorithms may require some extra space for comparison and temporary storage of few data elements. These algorithms do not require any extra space and sorting is said to happen in-place, or for example, within the array itself. This is called in-place sorting. Bubble sort is an example of in-place sorting.

However, in some sorting algorithms, the program requires space which is more than or equal to the elements being sorted. Sorting which uses equal or more space is called not-in-place sorting. Merge-sort is an example of not-in-place sorting.

Stable and Not Stable Sorting If a sorting algorithm, after sorting the contents, does not change the sequence of similar content in which they appear, it is called stable sorting.

If a sorting algorithm, after sorting the contents, changes the sequence of similar content in which they appear, it is called unstable sorting.

Stability of an algorithm matters when we wish to maintain the sequence of original elements, like in a tuple for example.

Adaptive and Non-Adaptive Sorting Algorithm A sorting algorithm is said to be adaptive, if it takes advantage of already 'sorted' elements in the list that is to be sorted. That is, while sorting if the source list has some element already sorted, adaptive algorithms will take this into account and will try not to re-order them.

A non-adaptive algorithm is one which does not take into account the elements which are already sorted. They try to force every single element to be re-ordered to confirm their sortedness.

Important Terms Some terms are generally coined while discussing sorting techniques, here is a brief introduction to them

Increasing Order

A sequence of values is said to be in increasing order, if the successive element is greater than the previous one. For example, 1, 3, 4, 6, 8, 9 are in increasing order, as every next element is greater than the previous element.

Decreasing Order

A sequence of values is said to be in decreasing order, if the successive element is less than the current one. For example, 9, 8, 6, 4, 3, 1 are in decreasing order, as every next element is less than the previous element.

Non-Increasing Order

A sequence of values is said to be in non-increasing order, if the successive element is less than or equal to its previous element in the sequence. This order occurs when the sequence contains duplicate values. For example, 9, 8, 6, 3, 3, 1 are in non-increasing order, as every next element is less than or equal to (in case of 3) but not greater than any previous element.

Non-Decreasing Order

A sequence of values is said to be in non-decreasing order, if the successive element is greater than or equal to its previous element in the sequence. This order occurs when the sequence contains duplicate values. For example, 1, 3, 3, 6, 8, 9 are in non-decreasing order, as every next element is greater than or equal to (in case of 3) but not less than the previous one.

14.4.2 Bubble Sort

Bubble sort is a simple sorting algorithm. This sorting algorithm is comparison-based algorithm in which each pair of adjacent elements is compared and the elements are swapped if they are not in order. This algorithm is not suitable for large data sets as its average and worst case complexity are of O(n2)(n2) where n is the number of items.

How Bubble Sort Works? We take an unsorted array for our example. Bubble sort takes O(n2)(n2) time so we're keeping it short and precise.

Bubble sort starts with very first two elements, comparing them to check which one is greater.

In this case, value 33 is greater than 14, so it is already in sorted locations. Next, we compare 33 with 27.

We find that 27 is smaller than 33 and these two values must be swapped.

The new array should look like this

Next we compare 33 and 35. We find that both are in already sorted positions.

Then we move to the next two values, 35 and 10.

We know then that 10 is smaller 35. Hence they are not sorted.

We swap these values. We find that we have reached the end of the array. After one iteration, the array should look like this

To be precise, we are now showing how an array should look like after each iteration. After the second iteration, it should look like this

Notice that after each iteration, at least one value moves at the end.

And when there's no swap required, bubble sorts learns that an array is completely sorted.

Now we should look into some practical aspects of bubble sort.

Algorithm We assume list is an array of n elements. We further assume that swap function swaps the values of the given array elements.

begin BubbleSort(list)

for all elements of list if list[i] > list[i+1] swap(list[i], list[i+1]) end if end for

return list

end BubbleSort Pseudocode We observe in algorithm that Bubble Sort compares each pair of array element unless the whole array is completely sorted in an ascending order. This may cause a few complexity issues like what if the array needs no more swapping as all the elements are already ascending.

To ease-out the issue, we use one flag variable swapped which will help us see if any swap has happened or not. If no swap has occurred, i.e. the array requires no more processing to be sorted, it will come out of the loop.

Pseudocode of BubbleSort algorithm can be written as follows

procedure bubbleSort(list : array of items)

loop = list.count;

for i = 0 to loop-1 do: swapped = false

for j = 0 to loop-1 do:

/* compare the adjacent elements */ if list[j] > list[j+1] then /* swap them */ swap(list[j], list[j+1]) swapped = true end if

end for

/*if no number was swapped that means array is sorted now, break the loop.*/

if(not swapped) then break end if

end for

end procedure return list Implementation One more issue we did not address in our original algorithm and its improvised pseudocode, is that, after every iteration the highest values settles down at the end of the array. Hence, the next iteration need not include already sorted elements. For this purpose, in our implementation, we restrict the inner loop to avoid already sorted values.

14.4.3 Insertion Sort

This is an in-place comparison-based sorting algorithm. Here, a sub-list is maintained which is always sorted. For example, the lower part of an array is maintained to be sorted. An element which is to be 'insert'ed in this sorted sub-list, has to find its appropriate place and then it has to be inserted there. Hence the name, insertion sort.

The array is searched sequentially and unsorted items are moved and inserted into the sorted sub-list (in the same array). This algorithm is not suitable for large data sets as its average and worst case complexity are of (n2), where n is the number of items.

How Insertion Sort Works? We take an unsorted array for our example.

Insertion sort compares the first two elements.

It finds that both 14 and 33 are already in ascending order. For now, 14 is in sorted sub-list.

Insertion sort moves ahead and compares 33 with 27.

And finds that 33 is not in the correct position.

It swaps 33 with 27. It also checks with all the elements of sorted sub-list. Here we see that the sorted sub-list has only one element 14, and 27 is greater than 14. Hence, the sorted sub-list remains sorted after swapping.

By now we have 14 and 27 in the sorted sub-list. Next, it compares 33 with 10.

These values are not in a sorted order.

So we swap them.

However, swapping makes 27 and 10 unsorted.

Hence, we swap them too.

Again we find 14 and 10 in an unsorted order.

We swap them again. By the end of third iteration, we have a sorted sub-list of 4 items.

This process goes on until all the unsorted values are covered in a sorted sub-list. Now we shall see some programming aspects of insertion sort.

Algorithm Now we have a bigger picture of how this sorting technique works, so we can derive simple steps by which we can achieve insertion sort.

Step 1 If it is the first element, it is already sorted. return 1; Step 2 Pick next element Step 3 Compare with all elements in the sorted sub-list Step 4 Shift all the elements in the sorted sub-list that is greater than the value to be sorted Step 5 Insert the value Step 6 Repeat until list is sorted Pseudocode procedure insertionSort(A : array of items) int holePosition int valueToInsert

for i = 1 to length(A) inclusive do:

/* select value to be inserted */ valueToInsert = A[i] holePosition = i

/*locate hole position for the element to be inserted */

while holePosition > 0 and A[holePosition-1] > valueToInsert do: A[holePosition] = A[holePosition-1] holePosition = holePosition -1 end while

/* insert the number at hole position */ A[holePosition] = valueToInsert

end for

end procedure

14.4.4 Selection Sort

Selection sort is a simple sorting algorithm. This sorting algorithm is an in-place comparison-based algorithm in which the list is divided into two parts, the sorted part at the left end and the unsorted part at the right end. Initially, the sorted part is empty and the unsorted part is the entire list.

The smallest element is selected from the unsorted array and swapped with the leftmost element, and that element becomes a part of the sorted array. This process continues moving unsorted array boundary by one element to the right.

This algorithm is not suitable for large data sets as its average and worst case complexities are of O(n2)(n2), where nn is the number of items.

How Selection Sort Works? Consider the following depicted array as an example.

For the first position in the sorted list, the whole list is scanned sequentially. The first position where 14 is stored presently, we search the whole list and find that 10 is the lowest value.

So we replace 14 with 10. After one iteration 10, which happens to be the minimum value in the list, appears in the first position of the sorted list.

For the second position, where 33 is residing, we start scanning the rest of the list in a linear manner.

We find that 14 is the second lowest value in the list and it should appear at the second place. We swap these values.

After two iterations, two least values are positioned at the beginning in a sorted manner.

The same process is applied to the rest of the items in the array.

Following is a pictorial depiction of the entire sorting process

Now, let us learn some programming aspects of selection sort.

Algorithm Step 1 Set MIN to location 0 Step 2 Search the minimum element in the list Step 3 Swap with value at location MIN Step 4 Increment MIN to point to next element Step 5 Repeat until list is sorted Pseudocode procedure selection sort list : array of items n : size of list

for i = 1 to n - 1 /* set current element as minimum*/ min = i

/* check the element to be minimum */

for j = i+1 to n if list[j] < list[min] then min = j; end if end for

/* swap the minimum element with the current element*/ if indexMin != i then swap list[min] and list[i] end if

end for

end procedure

14.4.5 Merge Sort

Merge sort is a sorting technique based on divide and conquer technique. With worst-case time complexity being O(nlogn)(nlogn), it is one of the most respected algorithms.

Merge sort first divides the array into equal halves and then combines them in a sorted manner.

How Merge Sort Works? To understand merge sort, we take an unsorted array as the following

We know that merge sort first divides the whole array iteratively into equal halves unless the atomic values are achieved. We see here that an array of 8 items is divided into two arrays of size 4.

This does not change the sequence of appearance of items in the original. Now we divide these two arrays into halves.

We further divide these arrays and we achieve atomic value which can no more be divided.

Now, we combine them in exactly the same manner as they were broken down. Please note the color codes given to these lists.

We first compare the element for each list and then combine them into another list in a sorted manner. We see that 14 and 33 are in sorted positions. We compare 27 and 10 and in the target list of 2 values we put 10 first, followed by 27. We change the order of 19 and 35 whereas 42 and 44 are placed sequentially.

In the next iteration of the combining phase, we compare lists of two data values, and merge them into a list of found data values placing all in a sorted order.

After the final merging, the list should look like this

Now we should learn some programming aspects of merge sorting.

Algorithm Merge sort keeps on dividing the list into equal halves until it can no more be divided. By definition, if it is only one element in the list, it is sorted. Then, merge sort combines the smaller sorted lists keeping the new list sorted too.

Step 1 if it is only one element in the list it is already sorted, return. Step 2 divide the list recursively into two halves until it can no more be divided. Step 3 merge the smaller lists into new list in sorted order. Pseudocode We shall now see the pseudocodes for merge sort functions. As our algorithms point out two main functions divide merge.

Merge sort works with recursion and we shall see our implementation in the same way.

procedure mergesort(var a as array) if (n == 1) return a

var l1 as array = a[0] ... a[n/2] var l2 as array = a[n/2+1] ... a[n]

l1 = mergesort(l1) l2 = mergesort(l2)

return merge(l1, l2) end procedure

procedure merge(var a as array, var b as array) var c as array

while (a and b have elements) if (a[0] > b[0]) add b[0] to the end of c remove b[0] from b else add a[0] to the end of c remove a[0] from a end if end while

while (a has elements) add a[0] to the end of c remove a[0] from a end while

while (b has elements) add b[0] to the end of c remove b[0] from b end while

return c

end procedure

14.4.6 Shell Sort

Shell sort is a highly efficient sorting algorithm and is based on insertion sort algorithm. This algorithm avoids large shifts as in case of insertion sort, if the smaller value is to the far right and has to be moved to the far left.

This algorithm uses insertion sort on a widely spread elements, first to sort them and then sorts the less widely spaced elements. This spacing is termed as interval. This interval is calculated based on Knuth's formula as

Knuth's Formula h=h3+1h=h3+1

where

hh is interval with initial value 1 This algorithm is quite efficient for medium-sized data sets as its average and worst case complexity are of O(n)(n), where nn is the number of items.

How Shell Sort Works? Let us consider the following example to have an idea of how shell sort works. We take the same array we have used in our previous examples. For our example and ease of understanding, we take the interval of 4. Make a virtual sub-list of all values located at the interval of 4 positions. Here these values are 35, 14, 33, 19, 42, 27 and 10, 44

We compare values in each sub-list and swap them (if necessary) in the original array. After this step, the new array should look like this

Then, we take interval of 2 and this gap generates two sub-lists - 14, 27, 35, 42, 19, 10, 33, 44

We compare and swap the values, if required, in the original array. After this step, the array should look like this

Finally, we sort the rest of the array using interval of value 1. Shell sort uses insertion sort to sort the array.

Following is the step-by-step depiction

We see that it required only four swaps to sort the rest of the array.

Algorithm Following is the algorithm for shell sort.

Step 1 Initialize the value of h Step 2 Divide the list into smaller sub-list of equal interval h Step 3 Sort these sub-lists using insertion sort Step 3 Repeat until complete list is sorted Pseudocode Following is the pseudocode for shell sort.

procedure shellSort() A : array of items

/* calculate interval*/ while interval < A.length /3 do: interval = interval * 3 + 1 end while

while interval > 0 do:

for outer = interval; outer < A.length; outer ++ do:

/* select value to be inserted */ valueToInsert = A[outer] inner = outer;

/*shift element towards right*/ while inner > interval -1 A[inner - interval] >= valueToInsert do: A[inner] = A[inner - interval] inner = inner - interval end while

/* insert the number at hole position */ A[inner] = valueToInsert

end for

/* calculate interval*/ interval = (interval -1) /3;

end while

end procedure

14.4.7 Quick Sort

Quick sort is a highly efficient sorting algorithm and is based on partitioning of array of data into smaller arrays. A large array is partitioned into two arrays one of which holds values smaller than the specified value, say pivot, based on which the partition is made and another array holds values greater than the pivot value.

Quick sort partitions an array and then calls itself recursively twice to sort the two resulting subarrays. This algorithm is quite efficient for large-sized data sets as its average and worst case complexity are of (n2), where n is the number of items.

Partition in Quick Sort Following animated representation explains how to find the pivot value in an array.

The pivot value divides the list into two parts. And recursively, we find the pivot for each sub-lists until all lists contains only one element.

Quick Sort Pivot Algorithm Based on our understanding of partitioning in quick sort, we will now try to write an algorithm for it, which is as follows.

Step 1 Choose the highest index value has pivot Step 2 Take two variables to point left and right of the list excluding pivot Step 3 left points to the low index Step 4 right points to the high Step 5 while value at left is less than pivot move right Step 6 while value at right is greater than pivot move left Step 7 if both step 5 and step 6 does not match swap left and right Step 8 if left right, the point where they met is new pivot Quick Sort Pivot Pseudocode The pseudocode for the above algorithm can be derived as

function partitionFunc(left, right, pivot) leftPointer = left rightPointer = right - 1

while True do while A[++leftPointer] < pivot do //do-nothing end while

while rightPointer > 0 A[–rightPointer] > pivot do //do-nothing end while

if leftPointer >= rightPointer break else swap leftPointer,rightPointer end if

end while

swap leftPointer,right return leftPointer

end function Quick Sort Algorithm Using pivot algorithm recursively, we end up with smaller possible partitions. Each partition is then processed for quick sort. We define recursive algorithm for quicksort as follows

Step 1 Make the right-most index value pivot Step 2 partition the array using pivot value Step 3 quicksort left partition recursively Step 4 quicksort right partition recursively Quick Sort Pseudocode To get more into it, let see the pseudocode for quick sort algorithm

procedure quickSort(left, right)

if right-left <= 0 return else pivot = A[right] partition = partitionFunc(left, right, pivot) quickSort(left,partition-1) quickSort(partition+1,right) end if end procedure

14.5 Search

14.5.1 Linear Search

Linear search is a very simple search algorithm. In this type of search, a sequential search is made over all items one by one. Every item is checked and if a match is found then that particular item is returned, otherwise the search continues till the end of the data collection.

Algorithm Linear Search (Array A, Value x)

Step 1: Set i to 1 Step 2: if i > n then go to step 7 Step 3: if A[i] = x then go to step 6 Step 4: Set i to i + 1 Step 5: Go to Step 2 Step 6: Print Element x Found at index i and go to step 8 Step 7: Print element not found Step 8: Exit

Pseudocode procedure linear$_search(list, value)$

for each item in the list

if match item == value

return the item's location

end if

end for

end procedure

14.5.2 Binary Search

Binary search is a fast search algorithm with run-time complexity of (log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.

Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues on the sub-array as well until the size of the subarray reduces to zero.

How Binary Search Works? For a binary search to work, it is mandatory for the target array to be sorted. We shall learn the process of binary search with a pictorial example. The following is our sorted array and let us assume that we need to search the location of value 31 using binary search.

First, we shall determine half of the array by using this formula

mid = low + (high - low) / 2 Here it is, 0 + (9 - 0) / 2 = 4 (integer value of 4.5). So, 4 is the mid of the array.

Now we compare the value stored at location 4, with the value being searched, i.e. 31. We find that the value at location 4 is 27, which is not a match. As the value is greater than 27 and we have a sorted array, so we also know that the target value must be in the upper portion of the array.

We change our low to mid + 1 and find the new mid value again.

low = mid + 1 mid = low + (high - low) / 2 Our new mid is 7 now. We compare the value stored at location 7 with our target value 31.

The value stored at location 7 is not a match, rather it is more than what we are looking for. So, the value must be in the lower part from this location.

Hence, we calculate the mid again. This time it is 5.

We compare the value stored at location 5 with our target value. We find that it is a match.

We conclude that the target value 31 is stored at location 5.

Binary search halves the searchable items and thus reduces the count of comparisons to be made to very less numbers.

Pseudocode The pseudocode of binary search algorithms should look like this

Procedure binary$_search A sorted array n size of array x value to be searched$

Set lowerBound = 1 Set upperBound = n

while x not found if upperBound < lowerBound EXIT: x does not exists.

set midPoint = lowerBound + (upperBound - lowerBound) / 2

if A[midPoint] < x set lowerBound = midPoint + 1

if A[midPoint] > x set upperBound = midPoint - 1

if A[midPoint] = x EXIT: x found at location midPoint

end while

end procedure

14.5.3 Interpolation Search

Interpolation search is an improved variant of binary search. This search algorithm works on the probing position of the required value. For this algorithm to work properly, the data collection should be in a sorted form and equally distributed.

Binary search has a huge advantage of time complexity over linear search. Linear search has worst-case complexity of (n) whereas binary search has (log n).

There are cases where the location of target data may be known in advance. For example, in case of a telephone directory, if we want to search the telephone number of Morphius. Here, linear search and even binary search will seem slow as we can directly jump to memory space where the names start from 'M' are stored.

Positioning in Binary Search In binary search, if the desired data is not found then the rest of the list is divided in two parts, lower and higher. The search is carried out in either of them.

Even when the data is sorted, binary search does not take advantage to probe the position of the desired data.

Position Probing in Interpolation Search Interpolation search finds a particular item by computing the probe position. Initially, the probe position is the position of the middle most item of the collection.

If a match occurs, then the index of the item is returned. To split the list into two parts, we use the following method

mid = Lo + ((Hi - Lo) / (A[Hi] - A[Lo])) * (X - A[Lo]) where

A = list Lo = Lowest index of the list Hi = Highest index of the list A[n] = Value stored at index n in the list If the middle item is greater than the item, then the probe position is again calculated in the sub-array to the right of the middle item. Otherwise, the item is searched in the subarray to the left of the middle item. This process continues on the sub-array as well until the size of subarray reduces to zero.

Runtime complexity of interpolation search algorithm is O(log(logn))(log(logn)) as compared to O(logn)(logn) of BST in favorable situations.

Algorithm As it is an improvisation of the existing BST algorithm, we are mentioning the steps to search the 'target' data value index, using position probing

Step 1 Start searching data from middle of the list. Step 2 If it is a match, return the index of the item, and exit. Step 3 If it is not a match, probe position. Step 4 Divide the list using probing formula and find the new midle. Step 5 If data is greater than middle, search in higher sub-list. Step 6 If data is smaller than middle, search in lower sub-list. Step 7 Repeat until match. Pseudocode A Array list N Size of A X Target Value

Procedure Interpolation$search()$

Set Lo 0 Set Mid -1 Set Hi N-1

While X does not match

if Lo equals to Hi OR A[Lo] equals to A[Hi] EXIT: Failure, Target not found end if

Set Mid = Lo + ((Hi - Lo) / (A[Hi] - A[Lo])) * (X - A[Lo])

if A[Mid] = X EXIT: Success, Target found at Mid else if A[Mid] $<$ X Set Lo to Mid+1 else if A[Mid] $>$ X Set Hi to Mid-1 end if end if

End While

End Procedure

14.5.4 Hash Table

Hash Table is a data structure which stores data in an associative manner. In a hash table, data is stored in an array format, where each data value has its own unique index value. Access of data becomes very fast if we know the index of the desired data.

Thus, it becomes a data structure in which insertion and search operations are very fast irrespective of the size of the data. Hash Table uses an array as a storage medium and uses hash technique to generate an index where an element is to be inserted or is to be located from.

Hashing Hashing is a technique to convert a range of key values into a range of indexes of an array. We're going to use modulo operator to get a range of key values. Consider an example of hash table of size 20, and the following items are to be stored. Item are in the (key,value) format.

(1,20) (2,70) (42,80) (4,25) (12,44) (14,32) (17,11) (13,78) (37,98) Sr. No. Key Hash Array Index 1 1 1 2 2 2 3 42 42 4 4 4 5 12 12 6 14 4 7 17 7 8 13 3 9 37 7 Linear Probing As we can see, it may happen that the hashing technique is used to create an already used index of the array. In such a case, we can search the next empty location in the array by looking into the next cell until we find an empty cell. This technique is called linear probing.

Sr. No. Key Hash Array Index After Linear Probing, Array Index 1 1 1 2 2 2 3 42 42 4 4 4 5 12 12 6 14 14 7 17 17 8 13 13 9 37 37 Basic Operations Following are the basic primary operations of a hash table.

Search Searches an element in a hash table. Insert inserts an element in a hash table. delete Deletes an element from a hash table. DataItem Define a data item having some data and key, based on which the search is to be conducted in a hash table.

struct DataItem int data; int key; ; Hash Method Define a hashing method to compute the hash code of the key of the data item.

int hashCode(int key) return key Search Operation Whenever an element is to be searched, compute the hash code of the key passed and locate the element using that hash code as index in the array. Use linear probing to get the element ahead if the element is not found at the computed hash code.

Example

struct DataItem *search(int key) //get the hash int hashIndex = hashCode(key);

//move in array until an empty while(hashArray[hashIndex] != NULL)

if(hashArray[hashIndex]->key == key) return hashArray[hashIndex];

//go to next cell ++hashIndex;

//wrap around the table hashIndex

return NULL; Insert Operation Whenever an element is to be inserted, compute the hash code of the key passed and locate the index using that hash code as an index in the array. Use linear probing for empty location, if an element is found at the computed hash code.

Example

void insert(int key,int data) struct DataItem *item = (struct DataItem*) malloc(sizeof(struct DataItem)); item->data = data; item->key = key;

//get the hash int hashIndex = hashCode(key);

//move in array until an empty or deleted cell while(hashArray[hashIndex] != NULL hashArray[hashIndex]->key != -1) //go to next cell ++hashIndex;

//wrap around the table hashIndex

hashArray[hashIndex] = item; Delete Operation Whenever an element is to be deleted, compute the hash code of the key passed and locate the index using that hash code as an index in the array. Use linear probing to get the element ahead if an element is not found at the computed hash code. When found, store a dummy item there to keep the performance of the hash table intact.

Example

struct DataItem* delete(struct DataItem* item) int key = item->key;

//get the hash int hashIndex = hashCode(key);

//move in array until an empty while(hashArray[hashIndex] !=NULL)

if(hashArray[hashIndex]->key == key) struct DataItem* temp = hashArray[hashIndex];

//assign a dummy item at deleted position hashArray[hashIndex] = dummyItem; return temp;

//go to next cell ++hashIndex;

//wrap around the table hashIndex

return NULL;

14.6 Graph

14.6.1 Graph Data Structure

A graph is a pictorial representation of a set of objects where some pairs of objects are connected by links. The interconnected objects are represented by points termed as vertices, and the links that connect the vertices are called edges.

Formally, a graph is a pair of sets (V,E)(V,E), where VV is the set of vertices and EE is the set of edges, connecting the pairs of vertices. Take a look at the following graph

In the above graph,

V=a,b,c,d,eV=a,b,c,d,e

E=ab,ac,bd,cd,deE=ab,ac,bd,cd,de

Definitions Mathematical graphs can be represented in data structure. We can represent a graph using an array of vertices and a two-dimensional array of edges. Before we proceed further, let's familiarize ourselves with some important terms

Vertex Each node of the graph is represented as a vertex. In the following example, the labeled circle represents vertices. Thus, A to G are vertices. We can represent them using an array as shown in the following image. Here A can be identified by index 0. B can be identified using index 1 and so on. Edge Edge represents a path between two vertices or a line between two vertices. In the following example, the lines from A to B, B to C, and so on represents edges. We can use a two-dimensional array to represent an array as shown in the following image. Here AB can be represented as 1 at row 0, column 1, BC as 1 at row 1, column 2 and so on, keeping other combinations as 0. Adjacency Two node or vertices are adjacent if they are connected to each other through an edge. In the following example, B is adjacent to A, C is adjacent to B, and so on. Path Path represents a sequence of edges between the two vertices. In the following example, ABCD represents a path from A to D.

Basic Operations Following are basic primary operations of a Graph

Add Vertex Adds a vertex to the graph. Add Edge Adds an edge between the two vertices of the graph. Display Vertex Displays a vertex of the graph.

14.6.2 Depth First Traversal

Depth First Search (DFS) algorithm traverses a graph in a depthward motion and uses a stack to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

As in the example given above, DFS algorithm traverses from A to B to C to D first then to E, then to F and lastly to G. It employs the following rules.

Rule 1 Visit the adjacent unvisited vertex. Mark it as visited. Display it. Push it in a stack. Rule 2 If no adjacent vertex is found, pop up a vertex from the stack. (It will pop up all the vertices from the stack, which do not have adjacent vertices.) Rule 3 Repeat Rule 1 and Rule 2 until the stack is empty. Algorithms Step Traversal Description 1. Initialize the stack. 2. Mark S as visited and put it onto the stack. Explore any unvisited adjacent node from S. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order. 3. Mark A as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both Sand D are adjacent to A but we are concerned for unvisited nodes only. 4. Visit D and mark it as visited and put onto the stack. Here, we have B and C nodes, which are adjacent to D and both are unvisited. However, we shall again choose in an alphabetical order. 5. We choose B, mark it as visited and put onto the stack. Here Bdoes not have any unvisited adjacent node. So, we pop Bfrom the stack. 6. We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find D to be on the top of the stack. 7. Only unvisited adjacent node is from D is C now. So we visit C, mark it as visited and put it onto the stack.

14.6.3 Breadth First Traversal

Breadth First Search (BFS) algorithm traverses a graph in a breadthward motion and uses a queue to remember to get the next vertex to start a search, when a dead end occurs in any iteration.

As in the example given above, BFS algorithm traverses from A to B to E to F first then to C and G lastly to D. It employs the following rules.

Rule 1 Visit the adjacent unvisited vertex. Mark it as visited. Display it. Insert it * in a queue. Rule 2 If no adjacent vertex is found, remove the first vertex from the queue. Rule 3 Repeat Rule 1 and Rule 2 until the queue is empty. Algorithms Step Traversal Description 1. Initialize the stack. 2. Mark S as visited and put it onto the stack. Explore any unvisited adjacent node from S. We have three nodes and we can pick any of them. For this example, we shall take the node in an alphabetical order. 3. Mark A as visited and put it onto the stack. Explore any unvisited adjacent node from A. Both Sand D are adjacent to A but we are concerned for unvisited nodes only. 4. Visit D and mark it as visited and put onto the stack. Here, we have B and C nodes, which are adjacent to D and both are unvisited. However, we shall again choose in an alphabetical order. 5. We choose B, mark it as visited and put onto the stack. Here Bdoes not have any unvisited adjacent node. So, we pop Bfrom the stack. 6. We check the stack top for return to the previous node and check if it has any unvisited nodes. Here, we find D to be on the top of the stack. 7. Only unvisited adjacent node is from D is C now. So we visit C, mark it as visited and put it onto the stack. At this stage, we are left with no unmarked (unvisited) nodes. But as per the algorithm we keep on dequeuing in order to get all unvisited nodes. When the queue gets emptied, the program is over

14.7 String

String manipulation is a basic operation of many algorithms and utilities such as data validation, text parsing, file conversions and others. The Java APIs contain three classes that are used to work with character data:

Character – A class whose instances can hold a single character value. String – An immutable class for working with multiple characters. StringBuffer and StringBuilder – Mutable classes for working with multiple characters. The String and StringBuffer classes are two you will use the most in your programming assignments. You use the String class in situations when you want to prohibit data modification; otherwise you use the StringBuffer class.

The String class In Java Strings can be created in two different ways. Either using a new operator

String demo1 = new String("This is a string");

char[] demo2 = 's','t','r','i','n','g'; String str = new String(demo2); or using a string literal

String demo3 = "This is a string"; The example below demonstrates differences between these initializations

String s1 = new String("Fester"); String s2 = new String("Fester"); String s3 = "Fester"; String s4 = "Fester"; Then

s1 == s2 returns false s1 == s3 returns false s3 == s4 returns true Because of the importance strings in real life, Java stores (at compile time) all strings in a special internal table as long as you create your strings using a string literal String s3 = "Fester". This process is called canonicalization - it replaces multiple string objects with a single object. This is why in the above example s3 and s4 refer to the same object. Also note that creating strings like s3 and s4 is more efficient. Review the code example StringOptimization.java that demonstrates time comparisons between these two ways of string creation.

Here are some important facts you must know about strings:

1. A string is not an array of characters. Therefore, to access a particular character in a string, you have to use the charAt() method. In this code snippet we get the fourth character which is 't':

String str = "on the edge of history"; char ch = str.charAt(3); 2. The toString() method is used when we need a string representation of an object.

The method is defined in the Object class. For most important classes that you create, you will want to override toString() and provide your own string representation.

3. Comparing strings content using == is the most common mistake beginners do. You compare the content using either equals() or compareTo() methods.

Basic String methods The String class contains an enormous amount of useful methods for string manipulation. The following table presents the most common String methods:

str.charAt(k) returns a char at position k in str. str.substring(k) returns a substring from index k to the end of str s.substring(k, n) returns a substring from index k to index n-1 of str str.indexOf(s) returns an index of the first occurrence of String s in str str.indexOf(s, k) returns an index of String s starting an index k in str str.startsWith(s) returns true if str starts with s str.startsWith(s, k) returns true if str starts with s at index k str.equals(s) returns true if the two strings have equal values str.equalsIgnoreCase(s) same as above ignoring case str.compareTo(s) compares two strings s.compareToIgnoreCase(t) same as above ignoring case Examine the code in BasicStringDemo.java for further details.

The StringBuffer class In many cases when you deal with strings you will use methods available in the companion StringBuffer class. This mutable class is used when you want to modify the contents of the string. It provides an efficient approach to dealing with strings, especially for large dynamic string data. StringBuffer is similar to ArrayList in a way that the memory allocated to an object is automatically expanded to take up additional data.

Here is an example of reversing a string using string concatenation

public static String reverse1(String s) String str = "";

for(int i = s.length() - 1; i>=0; i–) str += s.charAt(i);

return str; and using a StringBuffer's append

public static String revers2(String s) StringBuffer sb = new StringBuffer();

for(int i = s.length() - 1; i>=0; i–) sb.append(s.charAt(i));

return sb.toString(); Another way to reverse a string is to convert a String object into a StringBuffer object, use the reverse method, and then convert it back to a string:

public static String reverse3(String s) return new StringBuffer(s).reverse().toString(); The performance difference between these two classes is that StringBuffer is faster than String when performing concatenations. Each time a concatenation occurs, a new string is created, causing excessive system resource consumption.

Review the code example StringOverhead.java that demonstrates time comparisons of concatenation on Strings and StringBuffer.

StringTokenizer This class (from java.util package) allows you to break a string into tokens (substrings). Each token is a group of characters that are separated by delimiters, such as an empty space, a semicolon, and so on. So, a token is a maximal sequence of consecutive characters that are not delimiters. Here is an example of the use of the tokenizer (an empty space is a default delimiter):

String s = "Nothing is as easy as it looks"; StringTokenizer st = new StringTokenizer(s); while (st.hasMoreTokens()) String token = st.nextToken(); System.out.println("Token [" + token + "]"); Here, hasMoreTokens() method checks if there are more tokens available from the string, and nextToken() method returns the next token from the string tokenizer.

The set of delimiters (the characters that separate tokens) may be specified in the second argument of StringTokenizer. In the following example, StringTokenizer has a set of two delimiters: an empty space and an underscore:

String s = "Every$_s olution_b reedsnewproblems"; StringTokenizerst = newStringTokenizer(s, "_"); while($
$handcolumnspecifiestheregularexpressionconstructs, whiletheright-handcolumndescribestheconditionsu$

Character Classes

[abc] a, b, or c (simple class) $[^abc]Anycharacterexcepta, b, orc(negation)[a-zA-Z]athroughz, orAthroughZ, inclusive(range)[a-d[m-p]]athroughd, ormthroughp: [a-dm-p](union)[a-z[def]]d, e, orf(intersection)[a-z[^bc]]athroughz, exceptforbandc: [ad-z](subtraction)[a-z[^m-p]]athroughz, andnotmthroughp: [a-lq-z](subtraction)$
$danydigitfrom0to9$
$wanywordcharacter(a-z, A-Z, 0-9and_)$
$Wanynon-wordcharacter$
$sanywhitespacecharacter?appearingonceornotatall*appearingzeroormoretimes+$
$appearingoneormoretimesTheJavaStringclasshasseveralmethodsthatallowyoutoperformanoperationusin$

The matches() method The matches("regex") method returns true or false depending whether the string can be matched entirely by the regular expression "regex". For example,

"abc".matches("abc") returns True, but

"abc".matches("bc") returns False. In the following code examples we match all strings that start with any number of dots (denoted by *), followed by "abc" and end with one or more underscores (denoted by +).

String regex = ".*"+"abc"+"$_+$";

"..abc$_{".matches(regex);}$

"abc$_{".matches(regex);}$

"abc$_".matches(regex); ThereplaceAll()methodThemethodreplaceAll("regex", "replacement")replacese$
$lettersfromagivenstring$

String str = "Nothing 2is as <> easy AS it +$_=looks!"; str = str.replaceAll("[^a-zA-Z]", ""); Thepattern"[a-zA-Z]"describesallletters(inupperandlowercases).Nextwenegatethispattern, toge$
$letters"[^a-zA-Z]".$

In the next example, we replace a sequence of characters by "-"

String str = "aabfooaaaabfooabfoob"; str = str.replaceAll("a*b", "-"); The star "" in the pattern "ab" denotes that character "a" may be repeated zero or more times. The output: "-foo-foo-foo-";

The split() method The split("regex") splits the string at each "regex" match and returns an array of strings where each element is a part of the original string between two "regex" matches.

In the following example we break a sentence into words, using an empty space as a delimiter:

String s = "Nothing is as easy as it looks"; String[] st = s.split(" "); Tokens are stored in in an array of strings and could be be easily accessible using array indexes. In the next code example, we choose two delimiters: either an empty space or an underscore:

String s = "Every$_s olution_b reedsnewproblems"; String[] st = s.split("_|"); Whatifastringcontainsseveral$

", $thatdenotesarepetitivepattern$

String s = "Every$_s olution_{b^{reeds_n ew}{}_{problems";String[]st=s.split("_+");It'simportanttoobservethatsplit()mightreturnsemptytokens.}}$

String[] st = "Tomorrow".split("r"); we have three tokens, where the second token is empty string. That is so because split() returns tokens between two "regex" matches.

One of the widely use of split() is to break a given text file into words. This could be easily done by means of the metacharacter "" (any non-word character), which allows you to perform a "whole words only" search using a regular expression. A "word character" is either an alphabet character (a-z and A-Z) or a digit (0-9) or a underscore.

"Let's go, Steelers!!!".split("
W"); returns the following array of tokens

[Let, s, go, Steelers] Examine the code in Split.java for further details.

Pattern matching Pattern matching in Java is based on use of two classes

Pattern - compiled representation of a regular expression. Matcher - an engine that performs match operations. A typical invocation is the following, first we create a pattern

String seq = "CCCAA"; Pattern p = Pattern.compile("C*A*"); In this example we match all substrings that start with any number of Cs followed by any number of As. Then we create a Matcher object that can match any string against our pattern

Matcher m = p.matcher(seq); Finally, we do actual matching

boolean res = m.matches(); The Matcher class has another widely used method, called find(), that finds next substring that matches a given pattern. In the following example we cound the number of matches "ACC"

String seq = "CGTATCCCACAGCACCACACCCAACAACCCA"; Pattern p = Pattern.compile("A1C2"); Matcher m = p.matcher(seq); int count = 0; while(m.find()) count++; System.out.println("there are " + count + " ACC"); Examine the code example Matching.java for further details.

Pattern matching in Computational Biology

The DNA (the genetic blueprint) of any species is composed of about 4 billion ACGT nucleotides. DNA forms a double helix that has two strands of DNA binding and twisting together. In pattern matching problems we ignore the fact that DNA forms a double helix, and think of it only as a single strand. The other strand is complimentary. Knowing one strand allows uniquely determine the other one. Thus, DNA is essentially a linear molecule that looks like a string composed out of only four characters A, C, G, and T:

CGTATCCCACAGCACCACACCCAACAACCC Each nucleotides (also called a base) strongly binds to no more than two other bases. These links provides a linear model of DNA strand. The particular order of ACGT nucleotides is extremely important. Different orders generate humans, animals, corn, and other organisms. The size of the genome (a genome is all the DNA in an organism) does not necessarily correlate with the complexity of the organism it belongs to. Humans have less than a third as many genes as were expected.

Pattern matching in computational biology arises from the need to know characteristics of DNA sequences, such as

find the best way to align two sequences. find any common subsequences determine how well a sequence fits into a given model. Comparing various DNA sequencesn provide many uses. Current scientific theories suggest that very similar DNA sequences have a common ancestor. The more similar two sequences are, the more recently they evolved from a single ansestor. With such knowledge, for example, we can reconstruct a phylogenetic tree (known as a "tree of life".) that shows how long ago various organisms diverged and which species are closely related.

Challenges Strings: Making Anagrams References "Strings". Victor S.Adamchik, CMU. 2009

14.7.1 Tries

Introduction There are many algorithms and data structures to index and search strings inside a text, some of them are included in the standard libraries, but not all of them; the trie data structure is a good example of one that isn't.

Let word be a single string and let dictionary be a large set of words. If we have a dictionary, and we need to know if a single word is inside of the dictionary the tries are a data structure that can help us. But you may be asking yourself, "Why use tries if set and hash tables can do the same?"

There are two main reasons:

The tries can insert and find strings in O(L)O(L) time (where L represent the length of a single word). This is much faster than set , but is it a bit faster than a hash table. The set and the hash tables can only find in a dictionary words that match exactly with the single word that we are finding; the trie allow us to find words that have a single character different, a prefix in common, a character missing, etc.

The tries can be useful in TopCoder problems, but also have a great amount of applications in software engineering. For example, consider a web browser. Do you know how the web browser can auto complete your text or show you many possibilities of the text that you could be writing? Yes, with the trie you can do it very fast. Do you know how an orthographic corrector can check that every word that you type is in a dictionary? Again a trie. You can also use a trie for suggested corrections of the words that are present in the text but not in the dictionary.

What is a Tree? You may read about how wonderful the tries are, but maybe you don't know yet what the tries are and why the tries have this name. The word trie is an infix of the word "retrieval" because the trie can find a single word in a dictionary with only a prefix of the word. The main idea of the trie data structure consists of the following parts:

The trie is a tree where each vertex represents a single word or a prefix. The root represents an empty string (“”), the vertexes that are direct sons of the root represent prefixes of length 1, the vertexes that are 2 edges of distance from the root represent prefixes of length 2, the vertexes that are 3 edges of distance from the root represent prefixes of length 3 and so on. In other words, a vertex that are k edges of distance of the root have an associated prefix of length k. Let v and w be two vertexes of the trie, and assume that v is a direct father of w, then v must have an associated prefix of w. The next figure shows a trie with the words “tree”, “trie”, “algo”, “assoc”, “all”, and “also.”

Note that every vertex of the tree does not store entire prefixes or entire words. The idea is that the program should remember the word that represents each vertex while lower in the tree.

Coding a Trie The tries can be implemented in many ways, some of them can be used to find a set of words in the dictionary where every word can be a little different than the target word, and other implementations of the tries can provide us with only words that match exactly with the target word. The implementation of the trie that will be exposed here will consist only of finding words that match exactly and counting the words that have some prefix. This implementation will be pseudo code because different coders can use different programming languages.

We will code these 4 functions:

addWord. This function will add a single string word to the dictionary. countPreffixes. This function will count the number of words in the dictionary that have a string prefix as prefix. countWords. This function will count the number of words in the dictionary that match exactly with a given string word. Our trie will only support letters of the English alphabet. We need to also code a structure with some fields that indicate the values stored in each vertex. As we want to know the number of words that match with a given string, every vertex should have a field to indicate that this vertex represents a complete word or only a prefix (for simplicity, a complete word is considered also a prefix) and how many words in the dictionary are represented by that prefix (there can be repeated words in the dictionary). This task can be done with only one integer field words.

Because we want to know the number of words that have like prefix a given string, we need another integer field prefixes that indicates how many words have the prefix of the vertex. Also, each vertex must have references to all his possible sons (26 references). Knowing all these details, our structure should have the following members:

structure Trie integer words; integer prefixes; reference edges[26]; And we also need the following functions:

initialize(vertex) addWord(vertex, word); integer countPrefixes(vertex, prefix); integer countWords(vertex, word); First of all, we have to initialize the vertexes with the following function:

initialize(vertex) vertex.words=0 vertex.prefixes=0 for i=0 to 26 edges[i]=NoEdge The addWord function consists of two parameters, the vertex of the insertion and the word that will be added. The idea is that when a string word is added to a vertex vertex, we will add word to the corresponding branch of vertex cutting the leftmost character of word. If the needed branch does not exist, we will have to create it. All the TopCoder languages can simulate the process of cutting a character in constant time instead of creating a copy of the original string or moving the other characters.

addWord(vertex, word) if isEmpty(word) vertex.words=vertex.words+1 else vertex.prefixes=vertex.prefixes+1 k=firstCharacter(word) if(notExists(edges[k])) edges[k]=createEdge() initialize(edges[k]) cutLeftmostCharacter(word) addWord(edges[k], word) The functions countWords and countPrefixes are very similar. If we are finding an empty string we only have to return the number of words/prefixes associated with the vertex. If we are finding a non-empty string, we should to find in the corresponding branch of the tree, but if the branch does not exist, we have to return 0.

countWords(vertex, word) k=firstCharacter(word) if isEmpty(word) return vertex.words else if notExists(edges[k]) return 0 else cutLeftmostCharacter(word) return countWords(edges[k], word);

countPrefixes(vertex, prefix) k=firstCharacter(prefix) if isEmpty(word) return vertex.prefixes else if notExists(edges[k]) return 0 else cutLeftmostCharacter(prefix) return countWords(edges[k], prefix) Complexity Analysis In the introduction you may read that the complexity of finding and inserting a trie is linear, but we have not done the analysis yet. In the insertion and finding notice that lowering a single level in the tree is done in constant time, and every time that the program lowers a single level in the tree, a single character is cut from the string; we can conclude that every function lowers L levels on the tree and every time that the function lowers a level on the tree, it is done in constant time, then the insertion and finding of a word in a trie can be done in O(L) time. The memory used in the tries depends on the methods to store the edges and how many words have prefixes in common.

Other Kinds of Tries We used the tries to store words with lowercase letters, but the tries can be used to store many other things. We can use bits or bytes instead of lowercase letters and every data type can be stored in the tree, not only strings. Let flow your imagination using tries! For example, suppose that you want to find a word in a dictionary but a single letter was deleted from the word. You can modify the countWords function:

countWords(vertex, word, missingLetters) k=firstCharacter(word) if isEmpty(word) return vertex.word else if notExists(edges[k]) and missingLetters=0 return 0 else if notExists(edges[k]) cutLeftmostCharacter(word) return countWords(vertex, word, missingLetters-1) Here we cut a character but we don't go lower in the tree else We are adding the two possibilities: the first character has been deleted plus the first character is present r=countWords(vertex, word, missingLetters-1) cutLeftmostCharacter(word) r=r+countWords(edges[k], word, missingLetters) return r The complexity of this function may be larger than the original, but it is faster than checking all the subsets of characters of a word.

Challenges "Tries: Contacts". hackerrank. 2016 References "Using Tries – Topcoder". Topcoder.com. N.p., 2016. Web. 11 Oct. 2016.

14.7.2 Suffix Array and suffix tree

A suffix tree T is a natural improvement over trie used in pattern matching problem, the one defined over a set of substrings of a string s. The idea is very simple here. Such a trie can have a long paths without branches. If we only can reduce these long paths into one jump, we will reduce the size of the trie significantly, so this is a great first step in improving the complexity of operations on such a tree. This reduced trie defined over a subset of suffixes of a string s is called a suffix tree of s

For better understanding, let's consider the suffix tree T for a string s = abakan. A word abakan has 6 suffixes abakan , bakan, akan, kan, an, n and its suffix tree looks like this:

There is a famous algorithm by Ukkonen for building suffix tree for s in linear time in terms of the length of s. However, because it may look quite complicated at first sight, many people are discouraged to learn how it works. Fortunately, there is a great, I mean an excellent, description of Ukkonen's algorithm given on StackOverflow. Please refer to it for better understanding what a suffix tree is and how to build it in linear time.

Suffix trees can solve many complicated problems, because it contain so many information about the string itself. Fo example, in order to know how many times a pattern P occurs in s, it is sufficient to find P in T and return the size of a subtree corresponding to its node. Another well known application is finding the number of distinct substrings of s, and it can be solved easily with suffix tree, while the problem looks very complicated at first sight.

The post I linked from StackOverflow is so great, that you simple must read it. After that, you will be able to identify problems solvable with suffix trees easily.

If you want to know more about when to use a suffix tree, you should read this paper about the applications of suffix trees.

Suffix Array Suffix array is a very nice array based structure. Basically, it is a lexicographically sorted array of suffixes of a string s. For example, let's consider a string s = abakan. A word abakan has 6 suffixes abakan , bakan, akan, kan, an, n and its suffix tree looks like this:

Of course, in order to reduce space, we do not store the exact suffixes. It is sufficient to store their indices.

Suffix arrays, especially combined with LCP table (which stands for Longest Common Prefix of neighboring suffixes table), are very very useful for solving many problems. I recommend reading this nice programming oriented paper about suffix arrays, their applications and related problems by Stanford University.

Suffix arrays can be build easily in O(nlog2n)O(nlog2n) time, where n is the length of s, using the algorithm proposed in the paper from the previous paragraph. This time can be improved to O(nlogn)O(nlogn) using linear time sorting algorithm.

However, there is so extraordinary, cool and simple linear time algorithm for building suffix arrays by Kärkkäinen and Sanders, that reading it is a pure pleasure and you cannot miss it.

Correspondence between suffix tree and suffix array

It is also worth to mention, that a suffix array can be constructed directly from a suffix tree in linear time using DFS traversal. Suffix tree can be also constructed from the suffix array and LCP table as described here.

14.7.3 Knuth-Morris-Pratt Algorithm

The problem:

given a (short) pattern and a (long) text, both strings, determine whether the pattern appears somewhere in the text.

We'll go through the Knuth-Morris-Pratt (KMP) algorithm, which can be thought of as an efficient way to build these automata. I also have some working C++ source code which might help you understand the algorithm better.

First let's look at a naive solution.

suppose the text is in an array: char T[n]T[n] and the pattern is in another array: char P[m]P[m]. One simple method is just to try each possible position the pattern could appear in the text.

Naive string matching:

for (i=0; T[i] != "; i++) for (j=0; T[i+j] != " P[j] != " T[i+j]==P[j]; j++) ; if (P[j] == ") found a match There are two nested loops; the inner one takes O(m) iterations and the outer one takes O(n) iterations so the total time is the product, O(mn). This is slow; we'd like to speed it up.

In practice this works pretty well – not usually as bad as this O(mn) worst case analysis. This is because the inner loop usually finds a mismatch quickly and move on to the next position without going through all m steps. But this method still can take O(mn) for some inputs. In one bad example, all characters in T[] are "a"s, and P[] is all "a"'s except for one "b" at the end. Then it takes m comparisons each time to discover that you don't have a match, so mn overall.

Here's a more typical example. Each row represents an iteration of the outer loop, with each character in the row representing the result of a comparison (X if the comparison was unequal). Suppose we're looking for pattern "nano" in text "banananobano".

0 1 2 3 4 5 6 7 8 9 10 11 T: b a n a n a n o b a n o

i=0: X i=1: X i=2: n a n X i=3: X i=4: n a n o i=5: X i=6: n X i=7: X i=8: X i=9: n X i=10: X

Some of these comparisons are wasted work! For instance, after iteration i=2, we know from the comparisons we've done that T[3]="a", so there is no point comparing it to "n" in iteration i=3. And we also know that T[4]="n", so there is no point making the same comparison in iteration i=4.

Skipping outer iterations The Knuth-Morris-Pratt idea is, in this sort of situation, after you've invested a lot of work making comparisons in the inner loop of the code, you know a lot about what's in the text. Specifically, if you've found a partial match of j characters starting at position i, you know what's in positions T[i]...T[i+j-1]. You can use this knowledge to save work in two ways. First, you can skip some iterations for which no match is possible. Try overlapping the partial match you've found with the new match you want to find:

i=2: n a n i=3: n a n o Here the two placements of the pattern conflict with each other – we know from the i=2 iteration that T[3] and T[4] are "a" and "n", so they can't be the "n" and "a" that the i=3 iteration is looking for. We can keep skipping positions until we find one that doesn't conflict:

i=2: n a n i=4: n a n o Here the two "n"'s coincide. Define the overlap of two strings x and y to be the longest word that's a suffix of x and a prefix of y. Here the overlap of "nan" and "nano" is just "n". (We don't allow the overlap to be all of x or y, so it's not "nan"). In general the value of i we want to skip to is the one corresponding to the largest overlap with the current partial match:

String matching with skipped iterations:

i=0; while (i<n) for (j=0; T[i+j] != " P[j] != " T[i+j]==P[j]; j++) ; if (P[j] == ") found a match; i = i + max(1, j-overlap(P[0..j-1],P[0..m])); Skipping inner iterations The other optimization that can be done is to skip some iterations in

the inner loop. Let's look at the same example, in which we skipped from i=2 to i=4:

i=2: n a n i=4: n a n o In this example, the "n" that overlaps has already been tested by the i=2 iteration. There's no need to test it again in the i=4 iteration. In general, if we have a nontrivial overlap with the last partial match, we can avoid testing a number of characters equal to the length of the overlap. This change produces (a version of) the KMP algorithm:

KMP, version 1:

i=0; o=0; while (i<n) for (j=o; T[i+j] != '' P[j] != '' T[i+j]==P[j]; j++) ; if (P[j] == '') found a match; o = overlap(P[0..j-1],P[0..m]); i = i + max(1, j-o); The only remaining detail is how to compute the overlap function. This is a function only of j, and not of the characters in T[], so we can compute it once in a preprocessing stage before we get to this part of the algorithm. First let's see how fast this algorithm is.

KMP time analysis We still have an outer loop and an inner loop, so it looks like the time might still be O(mn). But we can count it a different way to see that it's actually always less than that. The idea is that every time through the inner loop, we do one comparison T[i+j]==P[j]. We can count the total time of the algorithm by counting how many comparisons we perform. We split the comparisons into two groups: those that return true, and those that return false. If a comparison returns true, we've determined the value of T[i+j]. Then in future iterations, as long as there is a nontrivial overlap involving T[i+j], we'll skip past that overlap and not make a comparison with that position again. So each position of T[] is only involved in one true comparison, and there can be n such comparisons total. On the other hand, there is at most one false comparison per iteration of the outer loop, so there can also only be n of those. As a result we see that this part of the KMP algorithm makes at most 2n comparisons and takes time O(n).

KMP and finite automata If we look just at what happens to j during the algorithm above, it's sort of like a finite automaton. At each step j is set either to j+1 (in the inner loop, after a match) or to the overlap o (after a mismatch). At each step the value of o is just a function of j and doesn't depend on other information like the characters in T[]. So we can draw something like an automaton, with arrows connecting values of j and labeled with matches and mismatches.

The difference between this and the automata we are used to is that it has only two arrows out of each circle, instead of one per character. But we can still simulate it just like any other automaton, by placing a marker on the start state (j=0) and moving it around the arrows. Whenever we get a matching character in T[] we move on to the next character of the text. But whenever we get a mismatch we look at the same character in the next step, except for the case of a mismatch in the state j=0.

So in this example (the same as the one above) the automaton goes through the sequence of states:

j=0 mismatch T[0] != "n" j=0 mismatch T[1] != "n" j=0 match T[2] == "n" j=1 match T[3] == "a" j=2 match T[4] == "n" j=3 mismatch T[5] != "o" j=1 match T[5] == "a" j=2 match T[6] == "n" j=3 match T[7] == "o" j=4 found match j=0 mismatch T[8] != "n" j=0 mismatch T[9] != "n" j=0 match T[10] == "n" j=1 mismatch T[11] != "a" j=0 mismatch T[11] != "n" This is essentially the same sequence of comparisons done by the KMP pseudocode above. So this automaton provides an equivalent definition of the KMP algorithm. As one student pointed out in lecture, the one transition in this automaton that may not be clear is the one from j=4 to j=0. In general, there should be a transition from j=m to some smaller value of j, which should happen on any character (there are no more matches to test before making this transition). If we want to find all occurrences of the pattern, we should be able to find an occurrence even if it overlaps another one. So for instance if the pattern were "nana", we should find both occurrences of it in the text "nanana". So the transition from j=m should go to the next longest position that can match, which is simply j=overlap(pattern,pattern). In this case overlap("nano","nano") is empty (all suffixes of "nano" use the letter "o", and no prefix does) so we go to j=0.

Alternate version of KMP The automaton above can be translated back into pseudo-code, looking a little different from the pseudo-code we saw before but performing the same comparisons.

KMP, version 2:

j = 0; for (i = 0; i < n; i++) for (;;) // loop until break if (T[i] == P[j]) // matches? j++; // yes, move on to next state if (j == m) // maybe that was the last state found a match; j = overlap[j]; break; else if (j == 0) break; // no match in state j=0, give up else j = overlap[j]; // try shorter partial match The code inside each iteration of the outer loop is essentially the same as the function match from the C++ implementation I've made available. One advantage of this version of the code is that it tests characters one by one, rather than performing random access in the T[] array, so (as in the implementation) it can be made to work for stream-based input rather than having to read the whole text into memory first. The overlap[j] array stores the values of overlap(pattern[0..j-1],pattern), which we still need to show how to compute.

Since this algorithm performs the same comparisons as the other version of KMP, it takes the same amount of time, O(n). One way of proving this bound directly is to note, first, that there is one true comparison (in which T[i]==P[j]) per iteration of the outer loop, since we break out of the inner loop when this happens. So there are n of these total. Each of these comparisons results in increasing j by one. Each iteration of the inner loop in which we don't break out of the loop results in executing the statement j=overlap[j], which decreases j. Since j can only decrease as many times as it's increased, the total number of times this happens is also O(n).

Computing the overlap function Recall that we defined the overlap of two strings x and y to be the longest word that's a suffix of x and a prefix of y. The missing component of the KMP algorithm is a computation of this overlap function: we need to know overlap(P[0..j-1],P) for each value of j>0. Once we've computed these values we can store them in an array and look them up when we need them. To compute these overlap functions, we need to know for strings x and y not just the longest word that's a suffix of x and a prefix of y, but all such words. The key fact to notice here is that if w is a suffix of x and a prefix of y, and it's not the longest such word, then it's also a suffix of overlap(x,y). (This follows simply from the fact that it's a suffix of x that is shorter than overlap(x,y) itself.) So we can list all words that are suffixes of x and prefixes of y by the following loop:

while (x != empty) x = overlap(x,y); output x; Now let's make another definition: say that shorten(x) is the prefix of x with one fewer character. The next simple observation to make is that shorten(overlap(x,y)) is still a prefix of y, but is also a suffix of shorten(x). So we can find overlap(x,y) by adding one more character to some word that's a suffix of shorten(x) and a prefix of y. We can just find all such words using the loop above, and return the first one for which adding one more character produces a valid overlap:

Overlap computation:

z = overlap(shorten(x),y) while (last char of x != y[length(z)]) if (z = empty) return overlap(x,y) = empty else z = overlap(z,y) return overlap(x,y) = z So this gives us a recursive algorithm for computing the overlap function in general. If we apply this algorithm for x=some prefix of the pattern, and y=the pattern itself, we see that all recursive calls have similar arguments. So if we store each value as we compute it, we can look it up instead of computing it again. (This simple idea of storing results instead of recomputing them is known as dynamic programming; we discussed it somewhat in the first lecture and will see it in more detail next time.) So replacing x by P[0..j-1] and y by P[0..m-1] in the pseudocode above and replacing recursive calls by lookups of previously computed values gives us a routine for the problem we're trying to solve, of computing these particular overlap values. The following pseudocode is taken (with some names changed) from the initialization code of the C++ implementation I've made available. The value in overlap[0] is just a flag to make the rest of the loop simpler. The code inside the for loop is the part that computes each overlap value.

KMP overlap computation:

overlap[0] = -1; for (int i = 0; pattern[i] != "; i++) overlap[i + 1] = overlap[i] + 1; while (overlap[i + 1] > 0 pattern[i] != pattern[overlap[i + 1] - 1]) overlap[i + 1] = overlap[overlap[i + 1] - 1] + 1; return overlap; Let's finish by analyzing the time taken by this part of the KMP algorithm. The outer loop executes m times. Each iteration of the inner loop decreases the value of the formula overlap[i+1], and this formula's value only increases by one when we move from one iteration of the outer loop to the next. Since the number of decreases is at most the number of increases, the inner loop also has at most m iterations, and the total time for the algorithm is O(m). The entire KMP algorithm consists of this overlap computation followed by the main part of the algorithm in which we scan the text (using the overlap values to speed up the scan). The first part takes O(m) and the second part takes O(n) time, so the total time is O(m+n).

Chương 15

Object Oriented Programming

View online http://magizbox.com/training/object$_o$$riented_p$$rogramming/site/$

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which may contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OOP, computer programs are designed by making them out of objects that interact with one another. There is significant diversity of OOP languages, but the most popular ones are class-based, meaning that objects are instances of classes, which typically also determine their type.

Many of the most widely used programming languages (such as C++, Java, Python etc.) are multi-paradigm programming languages that support object-oriented programming to a greater or lesser degree, typically in combination with imperative, procedural programming. Significant object-oriented languages include Java, C++, C, Python, PHP, Ruby, Perl, Delphi, Objective-C, Swift, Scala, Common Lisp, and Smalltalk.

15.1 OOP

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which are data structures that contain data, in the form of fields, often known as attributes; and code, in the form of procedures, often known as methods. A distinguishing feature of objects is that an object's procedures can access and often modify the data fields of the object with which they are associated (objects have a notion of "this" or "self"). In OO programming, computer programs are designed by making them out of objects that interact with one another.[1][2] There is significant diversity in object-oriented programming, but most popular languages are class-based, meaning that objects are instances of classes, which typically also determines their type. 1. A First Look Procedural vs Object Oriented 1

Procedural Approach

Focus is on procedures All data is shared: no protection More difficult to modify Hard to manage complexity Advantages of Object Orientation

People think in terms of object OO models map to reality OO models are: Easy to develop Easy to understand. 2. Principles encapsulation, inheritance, abstraction, polymorphism 2

Fundamental Principles of OOP In order for a programming language to be object-oriented, it has to enable working with classes and objects as well as the implementation and use of the fundamental object-oriented principles and concepts: inheritance, abstraction, encapsulation and polymorphism.

2.1 Encapsulation 3 4 5

Encapsulation is the packing of data and functions into a single component. The features of encapsulation are supported using classes in most object-oriented programming languages, although other alternatives also exist. It allows selective hiding of properties and methods in an object by building an impenetrable wall to protect the code from accidental corruption.

What it do? We will learn to hide unnecessary details in our classes and provide a clear and simple interface for working with them.

Example: A popular example you'll hear for encapsulation is driving a car. Do you need to know exactly how every aspect of a car works (engine, carburettor, alternator, and so on)? No - you need to know how to use the steering wheel, brakes, accelerator, and so on.

2.2 Inheritance 6 7

Inheritance is when an object or class is based on another object (prototypal inheritance) or class (class-based inheritance), using the same implementation (inheriting from an object or class) specifying implementation to maintain the same behavior (realizing an interface; inheriting behavior).

inherit everything, add data or functionality, override functions, super

What it do? We will explain how class hierarchies improve code readability and enable the reuse of functionality.

Example: A real-world example of inheritance is genetic inheritance. We all receive genes from both our parents that then define who we are. We share qualities of both our parents, and yet at the same time are different from them.

Example: we might classify different kinds of vehicles according to the inheritance hierarchy. Moving down the hierarchy, each kind of vehicle is both more specialized than its parent (and all of its ancestors) and more general than its children (and all of its descendants). A wheeled vehicle inherits properties common to all vehicles (it holds one or more people and carries them from place to place) but has an additional property that makes it more specialized (it has wheels). A car inherits properties common to all wheeled vehicles, but has additional, more specialized properties (four wheels, an engine, a body, and so forth). The inheritance relationship can be viewed as an is-a relationship. In this relationship, the objects become more specialized the lower in the hierarchy you go.

Look at the image above you will get a point.8 Yes, the derived class can access base class properties and still the derived class has its own properties.

2.3 Abstraction

In computer science, abstraction is a technique for managing complexity of computer systems. It works by establishing a level of complexity on which a person interacts with the system, suppressing the more complex details below the current level. The programmer works with an idealized interface (usually well defined) and can add additional levels of functionality that would otherwise be too complex to handle.

What it do? We will learn how to work through abstractions: to deal with objects considering their important characteristics and ignore all other details.

Example: You'll never buy a "device", but always buy something more specific : iPhone, Samsung Galaxy, Nokia 3310... Here, iPhone, Samsung Galaxy and Nokia 3310 are concrete things, device is abstract.

2.4 Polymorphism 9

Polymorphism is the provision of a single interface to entities of different types. A polymorphic type is one whose operations can also be applied to values of some other type, or types.

What it do? We will explain how to work in the same manner with different objects, which define a specific implementation of some abstract behavior.

Example: All animal can speak, but dogs woof, cats meow, and ducks quack

There are two types of polymorphism

Overloading (compile time polymorphism): methods have the same name but different parameters. Overriding (run time polymorphism): the implementation given in base class is replaced with that in sub class.

Example 10: Let us Consider Car example for discussing the polymorphism. Take any brand like Ford, Honda, Toyota, BMW, Benz etc., Everything is of type Car. But each have their own advanced features and more advanced technology involved in their move behavior.

3. Concepts Learn Object Oriented Programming though Mario Game

[embed]https://www.youtube.com/watch?v=HBbzYKMfx5Y[/embed]

How Mario get 1up

3.1. Object 11

Objects are key to understanding object-oriented technology. Look around right now and you'll find many examples of real-world objects: your dog, your desk, your television set, your bicycle. In mario world, Mario is an object.

Goomba is an object. Koopa is also an object. Even a coin and a pile are objects

Software objects are conceptually similar to real-world objects: they too consist of state and related behavior. An object stores its state in fields (variables in some programming languages) and exposes its behavior through methods (functions in some programming languages). Methods operate on an object's internal state and serve as the primary mechanism for object-to-object communication. Hiding internal state and requiring all interaction to be performed through an object's methods is known as data encapsulation — a fundamental principle of object-oriented programming. In Mario world, Mario has some fields like position (which indicate where Mario stands), state (which indicate whether Mario alive), and some methods like walk , fire or jump.

Goomba has some fields like position (which indicate where Goomba stands), state (which indicate whether Goomba die), and direction (which indicate the direction Goomba moves). Goomba has move method, and jumped$_{o}nmethod(whichoccurswhenitisjumpedonby$

Mario Objects, real scene

3.2 Class 12

In the real world, you'll often find many individual objects all of the same kind. There may be thousands of other bicycles in existence, all of the same make and model. Each bicycle was built from the same set of blueprints and therefore contains the same components. In object-oriented terms, we say that your bicycle is an instance of the class of objects known as bicycles. A class is the blueprint from which individual objects are created.

In Mario world, each coin object come from Coin class, and every Koomba come from Koomba class

3.3. Inheritance 13

Inheritance is a mechanism in OOP to design two or more entities that are different but share many common features.

Feature common to all classes are defined in the superclass The classes that inherit common features from the superclass are called subclasses In Mario World, Goomba and Koopa is in

AND MANY, MANY MORE

3.4. Association, Aggregation and Composition

13

Association:

Whenever two objects are related with each other the relationshiop is called association between object

Aggregation:

Aggregation is specialized from of association. In aggregation objects have their own life-cycle but there is ownership and child object can not belongs to another parent object. But this is only an ownership not the life-cycle control of child control through parent object.

Example: Student and Teacher, Person and address

Composition

Composition is again specialize form of aggregation and we can call this as 'life and death" relationship. It is a strong type of aggregation. Child object does not have their life-cycle and if parent object is deleted, all child object will also be deleted.

Example: House and room

3.5 Polymorphism 13

Polymorphism indicates the meaning of "many forms"

Polynorphism present a method that can have many definitions. Polymorphism is related to "over loading" and "over ridding".

Overloading indicates a method can have different definitions by defining different type of parameters.

[code] getPrice(): void getPrice(string name): void [/code]

3.6 Abstraction 13

Abstraction is the process of modelling only relevant features

Hide unnecessary details which are irrelevant for current purpose. Reduces complexity and aids understanding.

Abstraction provides the freedom to defer implementation decisions by avoiding commitments to details.

3.7 Interface 13

An interface is a contract consisting of group of related function prototypes whose usage is defined but whose implementation is not:

An interface definition specifies the interface's member functions, called methods, their return types, the number and types of parameters and what they must do.

These is no implementation associated with an interface.

4. Coupling and Cohesion 13

4.1 Coupling Coupling defines how dependent one object on another object (that is uses).

Coupling is a measure of strength of connection between any two system components. The more any one components knows about other components, the tighter (worse) the coupling is between those components.

4.2 Cohesion Cohesion defines how narrowly defined an object is. Functional cohesion refers measures how strongly objects are related.

Cohesion is a measure of how logically related the parts of an individual components are to each other, and to the overall components. The more logically related the parts of components are to each other higher (better) the cohesion of that components.

4.3 Object Oriented Design Low coupling and tight cohesion is good object oriented design.

Challenge Object Task 1: With boiler plate code, make an gif image (32x32) Mario fire ball and jump to get coins

5. NEXT Design Principles Design Patterns

15.2 UML

The Unified Modeling Language (UML) is a general-purpose, developmental, modeling language in the field of software engineering, that is intended to provide a standard way to visualize the design of a system.

http://www.yuml.me/ Use UML with IntellIJ: UML Designer Architecture 1

Design of a system consists of classes, interfaces and collaboration. UML provides class diagram, object diagram to support this. Implementation defines the components assembled together to make a complete physical system. UML component diagram is used to support implementation perspective. Process defines the flow of the system. So the same elements as used in Design are also used to support this perspective. Deployment represents the physical nodes of the system that forms the hardware. UML deployment diagram is used to support this perspective. Modelling Types 2

Diagrams Usecase Diagram 3 4 5

A use case diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

A use case diagram can identify the different types of users of a system and the different use cases and will often be accompanied by other types of diagrams as well.

Use case diagrams depict:

Use cases. A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse. (example) Actors. An actor is a person, organization, or external system that plays a role in one or more interactions with your system. Actors are drawn as stick figures. (example) Associations. Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line. The arrowhead is often used to indicating the direction of the initial invocation of the relationship or to indicate the primary actor within the use case. The arrowheads are typically confused with data flow and as a result I avoid their use. (example) Extend: Extend is a directed relationship that specifies how and when the behavior defined in usually supplementary (optional) extending use case can be inserted into the behavior defined in the extended use case. (example) Include is a directed relationship between two use cases which is used to show that behavior of the included use case (the addition) is inserted into the behavior of the including (the base) use case. (example) System boundary boxes (optional). You can draw a rectangle around the use cases, called the system boundary box, to indicates the scope of your system. Anything within the box represents functionality that is in scope and anything outside the box is not. System boundary boxes are rarely used, although on occasion I have used them to identify which use cases will be delivered in each major release of a system. (example) Packages (optional). Packages are UML constructs that enable you to organize model elements (such as use cases) into groups. Packages are depicted as file folders and can be used on any of the UML diagrams, including both use case diagrams and class diagrams. I use packages only when my diagrams become unwieldy, which generally implies they cannot be printed on a single page, to organize a large diagram into smaller ones. (example) Class Diagram 6

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects.

3.3.1 UML Association 9 10

Association

Association is reference based relationship between two classes. Here a class A holds a class level reference to class B. Association can be represented by a line between these classes with an arrow indicating the navigation direction. In case arrow is on the both sides, association has bidirectional navigation.

Aggregation

Aggregation (shared aggregation) is a "weak" form of aggregation when part instance is independent of the composite:

the same (shared) part could be included in several composites, and if composite is deleted, shared parts may still exist. Shared aggregation is shown as binary association decorated with a hollow diamond as a terminal adornment at the aggregate end of the association line. The diamond should be noticeably smaller than the diamond notation for N-ary associations. Shared aggregation is shown as binary association decorated with a hollow diamond.

Composition

Composition (composite aggregation) is a "strong" form of aggregation. Composition requirements/features listed in UML specification are:

it is a whole/part relationship, it is binary association part could be included in at most one composite (whole) at a time, and if a composite (whole) is deleted, all of its composite parts are "normally" deleted with it. Note, that UML does not define how, when and specific order in which parts of the composite are created. Also, in some cases a part can be removed from a composite before the composite is deleted, and so is not necessarily deleted as part of the composite.

Aggregation vs Composition

11

Sequence Diagram 7

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart.

A sequence diagram shows object interactions arranged in time sequence.

It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams or event scenarios.

Activity Diagram 8

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams are intended to model both computational and organizational processes (i.e. workflows). Activity diagrams show the overall flow of control.

UML - Architecture

UML - Modeling Types

UML - Use Case Diagrams

Use Case Diagram

UML Association Between Actor and Use Case

Class diagram

Sequence diagram

Activity diagram

Aggregation

UML Class Diagram: Association, Aggregation and Composition

Lecture Notes on Object-Oriented Programming: Object Oriented Aggregation

15.3 SOLID

SOLID Principles In computer programming, SOLID (single responsibility, open-closed, Liskov substitution, interface segregation and dependency inversion) is a mnemonic acronym introduced by Michael Feathers for the "first five principles" named by Robert C. Martin in the early 2000s that stands for five basic principles of object-oriented programming and design. The intention is that these principles, when applied together, will make it more likely that a programmer will create a system that is easy to maintain and extend over time. The principles of SOLID are guidelines that can be applied while working on software to remove code smells by providing a framework through which the programmer may refactor the software's source code until it is both legible and extensible. It is part of an overall strategy of agile and Adaptive Software Development.

"Dependency Management is an issue that most of us have faced. Whenever we bring up on our screens a nasty batch of tangled legacy code, we are experiencing the results of poor dependency management. Poor dependency management leads to code that is hard to change, fragile, and non-reusable."

Uncle Bob talk about several different design smells in the PPP book, all relating to dependency management. On the other hand, when dependencies are well managed, the code remains flexible, robust, and reusable. So dependency management, and therefore these principles, are at the foundation of the -ilities that software developers desire.

SRP - Single Responsibility A class should have one, and only one, reason to change.

A class should have only a single responsibility (i.e. only one potential change in the software's specification should be able to affect the specification of the class)

Original Paper

OCP - Open/Closed You should be able to extend a classes behavior, without modifying it.

Software entities ... should be open for extension, but closed for modification."

Original Paper

LSP - Liskov Substitution Derived classes must be substitutable for their base classes.

Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program. See also design by contract.

Original Paper

ISP - Interface Segregation Make fine grained interfaces that are client specific.

Many client-specific interfaces are better than one general-purpose interface.

Original Paper

DIP - Dependency Inversion Depend on abstractions, not on concretions.

One should "depend upon abstractions, not concretions."

Original Paper

References The Principles of OOD

15.4 Design Patterns

Design Patterns

Creational design patterns These design patterns are all about class instantiation. This pattern can be further divided into class-creation patterns and object-creational patterns. While class-creation patterns use inheritance effectively in the instantiation process, object-creation patterns use delegation effectively to get the job done.

Structural design patterns These design patterns are all about Class and Object composition. Structural class-creation patterns use inheritance to compose interfaces. Structural object-patterns define ways to compose objects to obtain new functionality.

Behavioral design patterns These design patterns are all about Class's objects communication. Behavioral patterns are those patterns that are most specifically concerned with communication between objects.

Design Pattern QA Examples of GoF Design Patterns in Java's core libraries Dependency Injection vs Factory Pattern What is Inversion of Control? What is so bad about singletons? What is the basic difference between Factory and Abstract Factory Patterns? When would you use the Builder Pattern? What is the difference between Builder Design pattern and Factory Design pattern? How do the Proxy, Decorator, Adapter, and Bridge Patterns differ? Abstract Factory Pattern Creates an instance of several families of classes Intuitive 1

Volkswagen Transparent Factory in Dresden

What is it? 2 The abstract factory pattern provides a way to encapsulate a group of individual factories that have a common theme without specifying their concrete classes.

In normal usage, the client software creates a concrete implementation of the abstract factory and then uses the generic interface of the factory to create the concrete objects that are part of the theme. The client doesn't know (or care) which concrete objects it gets from each of these internal factories, since it uses only the generic interfaces of their products.

This pattern separates the details of implementation of a set of objects from their general usage and relies on object composition, as object creation is implemented in methods exposed in the factory interface.

Design

Example Code

The most interesting factories in the world Abstract factory pattern Observer Pattern Intuitive

Definition 1 The observer pattern is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. It is mainly used to implement distributed event handling systems. The Observer pattern is also a key part in the familiar model–view–controller (MVC) architectural pattern. The observer pattern is implemented in numerous programming libraries and systems, including almost all GUI toolkits. Structure 2

Subject

knows its observers. Any number of Observer objects may observe a subject. provides an interface for attaching and detaching Observer objects Observer

defines an updating interface for objects that should be notified of changes in a subject. ConcreteSubject

stores state of interest to ConcreteObserver objects. sends a notification to its observers when its state changes. ConcreteObserver

maintains a reference to a ConcreteSubject object. stores state that should stay consistent with the subject's. implements the Observer updating interface to keep its state consistent with the subject's. Examples Example 1: Blog Manager Application

In this application, each user is an Observer, each blog is a Subject. When a blog post a new article (state change), user get an update. When users get update, they update their articles.

[code lang="java"] Blog sportBlog = new Blog("SPORT"); User user1 = new User("Fan1"); User user2 = new User("Fan2");

sportBlog.attach(user1); sportBlog.attach(user2);

sportBlog.post(new Article("football")); sportBlog.post(new Article("swimming")); user1.getArticles(); user2.getArticles();

sportBlog.detach(user1);‘ [/code]

Real Implementations Broadcast Receiver 3 4 on Android

More Articles http://javapapers.com/design-patterns/observer-design-pattern/ Comparison Observer/Observable pattern vs Publisher/Subscriber pattern 5 Observer/Observable pattern is mostly implemented in a synchronous way, i.e. the observable calls the appropriate method of all its observers when some event occurs. The Publisher/Subscriber pattern is mostly implemented in an asynchronous way (using message queue). In the Observer/Observable pattern, the observers are aware of the observable. Whereas, in Publisher/Subscriber, publishers and subscribers don't need to know each other. They simply communicate with the help of message queues. Observer pattern

Broadcast Receiver

Design Patterns: Elements of Reusable Object-Oriented Software

Which design patterns are used on Android?

stackoverflow, Difference between Observer, Pub/Sub, and Data Binding

Chương 16

Database

View online http://magizbox.com/training/computer$_science/site/database/$

16.1 Introduction

Relational DBMS: Oracle, MySQL, SQLite

Key-value Stores: Redis, Memcached

Document stores: MongoDB

Graph: Neo4j

Wide column stores: Cassandra, HBase

Design and Modeling (a.k.a Data Definition) 1.1 Schema A database schema of a database system is its structure described in a formal language supported by the database management system (DBMS) and refers to the organization of data as a blueprint of how a database is constructed (divided into database tables in the case of Relational Databases). The formal definition of database schema is a set of formulas (sentences) called integrity constraints imposed on a database. These integrity constraints ensure compatibility between parts of the schema. All constraints are expressible in the same language. A database can be considered a structure in realization of the database language. The states of a created conceptual schema are transformed into an explicit mapping, the database schema. This describes how real world entities are modeled in the database.

1.1.1 Type In computer science and computer programming, a data type or simply type is a classification identifying one of various types of data, such as real, integer or Boolean, that determines the possible values for that type; the operations that can be done on values of that type; the meaning of the data; and the way values of that type can be stored.

TEXT, INT, ENUM, TIMESTAMP

1.2 Cardinality (a.k.a Relationship) Foreign key, Primary key

1.2 Indexing A database index is a data structure that improves the speed of data retrieval operations on a database table at the cost of additional writes and storage space to maintain the index data structure. Indexes are used to quickly locate data without having to search every row in a database table every time a database table is accessed. Indexes can be created using one or more columns of a database table, providing the basis for both rapid random lookups and efficient access of ordered records. Why Indexing is important?

Indexing in MySQL

CREATE INDEX NameIndex ON Employee (name) SELECT * FROM Employee WHERE name = 'Ashish' 2. Data Manipulation Create - Read - Update - Delete Create or add new entries Read, retrieve, search, or view existing entries * Update or edit existing entries * Delete/deactivate existing entries /* create */ CREATE TABLE Guests (id INT(6) UNSIGNED AUTO$_I$NCREMENTPRIMARYKEY, firstnameVARCHAR(30)NOTNULL, lastnameVARCHAR(3 create(insert)*/INSERTINTOGuests(firstname, lastname, email)VALUES('John',' Doe',' john@exam read*/SELECT*FROMGuestsWHEREid = 1/*update*/UPDATEGuestsSETlastname =' Doe'WHEREid = 1/*delete*/DELETEFROMGuestsWHEREid = 1'3.DataRetrieveTransaction3.1Da

Get user id, user name and number of post of this user

SELECT user.id, user.name, COUNT(post.*) AS posts FROM user LEFT OUTER JOIN post ON post.owner$_i$d = user.idGROUPBYuser.id'Selectuserwhoonlyorderonetime.

SELECT name, COUNT(name) AS c FROM orders GROUP BY name HAVING c = 1; Calculate the longest period (in days) that the company has gone without a hiring or firing anyone.

SELECT x.date, MIN(y.date) y$_d$ate, DATEDIFF(MIN(y.date), x.date)daysFROM(SELECThiredate x.dateGROUPBYx.dateORDERBYdaysDESCLIMIT1; DataRetrieveAPI

API Description get get single item Get dog by id

Dog.get(1) find find items

@see collection.find()

Find dog name "Max"

Dog.find("name": "Max") sort sort items

@see cursor.sort

Get 10 older dogs

Dog.find().sort("age", limit: 10) aggregate sum, min, max items

@see collection.aggregate

Get sum of dogs' age

Dog.find().aggregate("sum$_a$ge" : sum: "age") 3.2 Transaction A transaction symbolizes a unit of work performed within a database management system (or similar system) against a database, and treated in a coherent and reliable way independent of other transactions. A transaction generally represents any change in database. Example: Transfer 900fromAccount

Bob to Alice

start transaction select balance from Account where Account$_N$umber =' Bob'; selectbalancefromAccountwhereAccount$_N$umber =' Alice'; updateAccountsetbalance = balance−900hereAccount$_N$umber =' Bob'; updateAccountsetbalance = balance+ 900hereAccount$_N$umber =' Alice'; commit; //ifallsqlqueriessuccedrollback; //ifanyofSqlqueriesfailedore

In computer science, ACID (Atomicity, Consistency, Isolation, Durability) is a set of properties that guarantee that database transactions are processed reliably. In the context of databases, a single logical operation on the data is called a transaction.

For example, a transfer of funds from one bank account to another, even involving multiple changes such as debiting one account and crediting another, is a single transaction. ![][16]

4. Backup and Restore Sometimes it is desired to bring a database back to a previous state (for many reasons, e.g., cases when the database is found corrupted due to a software error, or if it has been updated with erroneous data).

To achieve this a backup operation is done occasionally or continuously, where each desired database state (i.e., the values of its data and their embedding in database's data structures) is kept within dedicated backup files (many techniques exist to do this effectively). When this state is needed, i.e., when it is decided by a database administrator to bring the database back to this state (e.g., by specifying this state by a desired point in time when the database was in this state), these files are utilized to restore that state.

5. Migration In software engineering, schema migration (also database migration, database change management) refers to the management of incremental, reversible changes to relational database schemas. A schema migration is performed on a database whenever it is necessary to update or revert that database's schema to some newer or older version. Example: Android Migration by droid-migrate

droid-migrate init -d my$_database droid-migrate generate up droid-migrate generate down Example : Database Seeding with Laravel$

6. Active record pattern | Object-Relational Mapping (ORM) Object-relational mapping in computer science is a programming technique for converting data between incompatible type systems in object-oriented programming languages. This creates, in effect, a "virtual object database" that can be used from within the programming language. There are both free and commercial packages available that perform object-relational mapping, although some programmers opt to create their own ORM tools.

Example

php

$employee = new Employee()$;employee->setName("Joe"); $employee -> save(); Android$

public class User @DatabaseField(id = true) String username; @DatabaseField String password; @DatabaseField String email; @DatabaseField String alias; public User() Implementations

Android: [ormlite-android] PHP: [Eloquent]

16.2 SQL

SQL SELECT * FROM WORLD

INSERT INTO

SELECT * FROM girls

16.3 MySQL

MySQL

MySQL is an open-source relational database management system (RDBMS); in July 2013, it was the world's second most widely used RDBMS, and the most widely used open-source client–server model RDBMS. It is named after co-founder Michael Widenius's daughter, My. The SQL abbreviation stands for Structured Query Language. The MySQL development project has made its source code available under the terms of the GNU General Public License, as well as under a variety of proprietary agreements. MySQL was owned and sponsored by a single for-profit firm, the Swedish company MySQL AB, now owned by Oracle Corporation. For proprietary use, several paid editions are available, and offer additional functionality.

MySQL: Docker Docker Run docker pull mysql docker run -d -p 3306:3306 –env MYSQL$_ROOT_PASSWORD = docker\ --envMYSQL_DATABASE = docker\ --envMYSQL_USER = docker\ --envMYSQL_PASSWORD = docker$ mysql$Note : OnWindows, viewyour$0.0.0.0$IPbyrunningbelowcommandline(oryoucanturnonKitem$

Docker Compoose Step 1: Clone Docker Project

git clone https://github.com/magizbox/docker-mysql.git mv docker-mysql mysql Step 2: Docker Compose

version: "2"

services: mysql: build: ./mysql/. ports: - 3306:3306 environment: - MYSQL$_ROOT_PASSWORD = docker - MYSQL_DATABASE = docker - MYSQL_USER = docker - MYSQL_PASSWORD = dockervolumes : -./data/mysql : /var/lib/mysqlDockerFoldermysql/configmy.cnfDockerfileVerifydoc$ $machinelsNAMEACTIVEDRIVERSTATEURLSWARMdefault*virtualboxRunningtcp :$ //192.168.99.100 : 2376$YoucanaddphpmyadmintoseemysqIworks$

phpmyadmin: image: phpmyadmin/phpmyadmin links: - mysql environment: - PMA$_ARBITRARY = 1ports : -80 : 80Seeitworks$

Go to localhost Login with Server=mysql, Username=docker, Password=docker

16.4 Redis

Redis is an open source (BSD licensed), in-memory data structure store, used as database, cache and message broker. 1

It supports data structures such as strings, hashes, lists, sets, sorted sets with range queries, bitmaps, hyperloglogs and geospatial indexes with radius queries.

Redis has built-in replication, Lua scripting, LRU eviction, transactions and different levels of on-disk persistence, and provides high availability via Redis Sentinel and automatic partitioning with Redis Cluster.

Redis: Client Python Client pipy/redis

Installation

pip install redis Usage

import redis r = redis.StrictRedis(host='localhost', port=6379, db=0) r.set('foo', 'bar') -> True

r.get('foo') -> 'bar'

r.delete('foo')

after delete r.get('foo') -> None Java Client https://redislabs.com/redis-java

Redis: Docker Docker Run docker run -d -p 6379:6379 redis Docker Compose version: "2"

services: redis: image: redis ports: - 6379:6379 Redis.io

16.5 MongoDB

MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling.

MongoDB provides high performance data persistence. In particular, Support for embedded data models reduces I/O activity on database system. Indexes support faster queries and can include keys from embedded documents and arrays. MongoDB is 1 in the Document Store according to db-engines

Client Mongo Shell The mongo shell is an interactive JavaScript interface to MongoDB and is a component of the MongoDB package. You can use the mongo shell to query and update data as well as perform administrative operations.

Start Mongo

Once you have installed and have started MongoDB, connect the mongo shell to your running MongoDB instance. Ensure that MongoDB is running before attempting to launch the mongo shell.

mongo Interact with mongo via shell

Show list database > show dbs

Create or use a database > use <database$_{n}ame >> usetestexample$

List collection > show collections

Create or use a collection > db.<collection$_{n}ame >> db.new_{c}ollectionexample$

Read document > db.new$_{c}ollection.find()$

Insert new document > db.new$_{c}ollection.insertOne("a" : "b")$

Update document > db.new$_{c}ollection.update("a" : "b",$ set: "a": "bcd")

Remove document > db.new$_{c}ollection.remove("a" : "b")PyMongo-PythonClientPyMongoisaPythond$

Installation We recommend using pip to install pymongo on all platforms:

pip install pymongo Usage import pymongo create connection client = pymongo.MongoClient('127.0.0.1', 27017) -> MongoClient(host=['127.0.0.1:27017'], document$_{c}lass = dict, tz_{a}ware = False, connect = True)$

create database db = client.db$_{t}est- > Database(MongoClient(host = ['127.0.0.1 : 27017'], document_{c}lass = dict, tz_{a}ware = False, connect = True), u'db_{t}est')$

create collection (collection is the same with table in SQL) collection = db.new$_{c}ollection$

insert document to collection (document is the same with rows in SQL) db.collection.insert$_{o}ne("c" : "d")- >< pymongo.results.InsertOneResultat0x7f7eab3c9f00 >$

read document of collection db.new$_{c}ollection.find_{o}ne("c" : "d")- > u'_{i}d' : ObjectId('589a8195f23e627a9$

update documents of collection db.new$_{c}ollection.update("c" : "d",$ "set": "c": "def") -> u'n': 1, u'nModified': 1, u'ok': 1.0, 'updatedExisting': True

remove document of collection db.new$_{c}ollection.remove("c" : "def")- > u'n' : 1, u'ok' : 1.0DockerDockerRunRunimagesandshareport$

docker run -p 27017:27017 mongo:latest

Chương 17

Hệ điều hành

Những phần mềm không thể thiếu

* Trình duyệt Google Chrome (với các extensions Scihub, Mendeley Desktop, Adblock) * Adblock extension * Terminal (Oh-my-zsh) * IDE Pycharm để code python * Quản lý phiên bản code Git * Bộ gõ ibus-unikey trong Ubuntu hoặc unikey (Windows) (Ctrl-Space để chuyển đổi ngôn ngữ) * CUDA (lập trình trên GPU)

Xem thông tin hệ thống

Phiên bản ‘ubuntu 16.04‘

```
sudo apt-get install sysstat
```

Xem hoạt động (

“‘ mpstat -A “‘

CPU của mình có bao nhiều core, bao nhiêu siblibngs

“‘ cat /proc/cpuinfo

processor : 23 vendor$_i d : GenuineIntel cpufamily : 6 model : 62 modelname : Intel(R)Xeon(R)CPUE5-2430v2@2.50GHz stepping : 4 microcode : 0x428 cpuMHz : 1599.707 cachesize : 15360 KBphysicalid : 1 siblings : 12 coreid : 5 cpucores : 6 apicid : 43 initialapicid : 43 fpu : yes fpu_exception : yes cpuidlevel : 13 wp : yes flags : fpuvmedepsetscmsrpaemcecx8apicsepmtrrpgemcacmovpatpse36clflushdtsacpimmxfxsrssesse$

$5005.20 clflushsize : 64 cache_alignment : 64 addresssizes : 46 bitsphysical, 48 bitsvirtualpowermanagement :$

“‘

Kết quả cho thấy cpu của 6 core và 12 siblings

Chương 18

Ubuntu

Chuyện terminal

Terminal là một câu chuyện muôn thuở của bất kì ông coder nào thích customize, đẹp, tiện (và bug kinh hoàng). Hiện tại mình đang thấy combo này khá ổn Terminal (Ubuntu) (Color: Black on white, Build-in schemes: Tango) + zsh + oh-my-zsh (fishy-custom theme). Những features hay ho

* Làm việc tốt trên cả Terminal (white background) và embedded terminal của Pycharm (black background) * Hiển thị folder dạng ngắn (chỉ ký tự đầu tiên) * Hiển thị brach của git ở bên phải

`![Imgur](https://i.imgur.com/q53vQdH.png)`

Chuyện bộ gõ

Làm sao để khởi động lại ibus, thỉnh thoảng lại chết bất đắc kì tử [[1]]"'$ibus-daemonibusrestart$"'

Chuyện lỗi login loop

Phiên bản: 'ubuntu 16.04'

27/12/2017: Lại dính lỗi không thể login. Lần này thì lại phải xóa bạn KDE đi. Kể cũng hơn buồn. Nhưng nhất quyết phải enable được tính năng Windows Spreading (hay đại loại thế). Hóa ra khi ubuntu bị lỗi không có lancher hay toolbar là do bạn unity plugin chưa được enable. Oài. Sao người hiền lành như mình suốt ngày bị mấy lỗi vớ vẩn thế không biết.

20/11/2017: Hôm nay đen thật, dính lỗi login loop. Fix mãi mới được. Thôi cũng kệ. Cảm giác bạn KDE này đỡ bị lỗi ibus-unikey hơn bạn GNOME. Hôm nay cũng đổi bạn zsh theme. Chọn mãi chẳng được bạn nào ổn ổn, nhưng không thể chịu được kiểu suggest lỗi nữa rồi. Đôi khi thấy default vẫn là tốt nhất.

21/11/2017: Sau một ngày trải nghiệm KDE, cảm giác giao diện mượt hơn GNOME. Khi overview windows với nhiều màn hình tốt và trực quan hơn. Đặc biệt là không bị lỗi ibus nữa. Đổi terminal cũng cảm giác ổn ổn. Không bị lỗi suggest nữa.

[[1]] : $https://askubuntu.com/questions/389903/ibus-doesnt-seem-to-restart$

Chương 19

Networking

View online http://magizbox.com/training/computer$_{s}cience/site/networking/$

TCP/IP TCP/IP is the protocol that has run the Internet for 30 years.

How TCP/IP works

Read More

Happy 30th Anniversary, Internet and TCP/IP!!! P2P Peer-to-peer (P2P) computing or networking is a distributed application architecture that partitions tasks or workloads between peers. Peers are equally privileged, equipotent participants in the application. They are said to form a peer-to-peer network of nodes.

Peers make a portion of their resources, such as processing power, disk storage or network bandwidth, directly available to other network participants, without the need for central coordination by servers or stable hosts.[1] Peers are both suppliers and consumers of resources, in contrast to the traditional client-server model in which the consumption and supply of resources is divided. Emerging collaborative P2P systems are going beyond the era of peers doing similar things while sharing resources, and are looking for diverse peers that can bring in unique resources and capabilities to a virtual community thereby empowering it to engage in greater tasks beyond those that can be accomplished by individual peers, yet that are beneficial to all the peers.

bridge vs NAT When you create a new virtual machine, you have one of many options when it comes to choosing your network connectivity. Two common options are to use either bridged networking or network address translation (NAT). So, what exactly does that look like? Take a look at the figure below.

NAT: In this diagram, the vertical line next to the firewall represents the production network and you can see that 192.168.1.1 is the IP address of the company's firewall that connects them to the Internet. There is also a virtual host with three virtual machines running inside it. The big red circle represents the virtual adapter to which NAT-based virtual machines connect (172.16.1.1). You can see that there are two such virtual machines with IP addresses of 172.16.1.2 and 172.16.1.3. When you configure a virtual machine as using NAT, it doesn't see the production network directly. In fact, all traffic coming from the virtual machine will share the VM host's IP address. Behind the scenes, traffic from the virtual machines is routed on the virtual host and sent out via the host's physical adapter and, eventually, to the Internet.

bridge: The third virtual machine (192.168.1.3) is configured in "bridged" mode which basically means that the virtual network adapter in that virtual machine is bridged to the production network and that virtual machine operates as if it exists directly on the production network. In fact, this virtual machine won't even be able to see the two NAT-based virtual machines since they're on different networks.

Read more: NAT vs. bridged network: A simple diagram

Chương 20

UX - UI

View online http://magizbox.com/training/computer$_{s}cience/site/ux/$

1. Design Principles UI Design Do's and Don'ts Android Design Principles
2. Design Trends 2.1 Material Design 1 components

We challenged ourselves to create a visual language for our users that synthesizes the classic principles of good design with the innovation and possibility of technology and science. This is material design. This spec is a living document that will be updated as we continue to develop the tenets and specifics of material design.

Tools

materialpalette.com Icon: fa2png UI Components

Data Binding Transclusion Directive - Fragments

Messaging Intent Android 1

Intents are asynchronous messages which allow application components to request functionality from other Android components. Intents allow you to interact with components from the same applications as well as with components contributed by other applications. For example, an activity can start an external activity for taking a picture.

Intents are objects of the android.content.Intent type. Your code can send them to the Android system defining the components you are targeting. For example, via the startActivity() method you can define that the intent should be used to start an activity.

An intent can contain data via a Bundle. This data can be used by the receiving component.

Style Theme Android Development: Explaining Styles and Themes, https://m.youtube.com//watch?v=M

Responsive Design Support Multi Screen 2

Intent Android

Support Multi Screen

Chương 21

Service-Oriented Architecture

View online http://magizbox.com/training/computer$_{s}cience/site/software_{a}rchitecture/$

A service-oriented architecture (SOA) is an architectural pattern in computer software design in which application components provide services to other components via a communications protocol, typically over a network. The principles of service-orientation are independent of any vendor, product or technology. 2

Generally accepted view 1 Boundaries are explicit Services are autonomous Services share schema and contract, not class Service compatibility is based on policy Microservices In computing, microservices is a software architecture style in which complex applications are composed of small, independent processes communicating with each other using language-agnostic APIs. These services are small building blocks, highly decoupled and focussed on doing a small task, facilitating a modular approach to system-building. One of concepts which integrates microservices as a software architecture style is dew computing. 1

Properties 2 Each running in its own process Communicating with lightweight mechanisms, often an HTTP resource API Build around business capabilities Independently deployable fully automated deployment Maybe in a different programming language and use different data storage technologies Monolith vs Microservice Monolith Microservice Simplicity Partial Deployment Consistency Availability Inter-module refactoring Preserve Modularity Multiple Platforms Benefits 4 Their small size enables developers to be most productive. It's easy to comprehend and test each service. You can correctly handle failure of any dependent service. They reduce impact of correlated failures. Web Service RESTful API

REST Client Sense (Beta)

A JSON aware developer console to ElasticSearch.

API Document and Client Generator http://swagger.io/swagger-editor/

API Client CRUD Pet

API Client Method URL Body Return Body Method GET /pets [Pet] PetApi.list()

POST /pets/ Pet Pet PetApi.create(pet) GET /pets/pet$_{i}dPetPetApi.get(pet_{i}d)PUT/pets/pet_{i}dPetPetPetA$

CRUD Store

GET /stores StoreApi.list() ... ... Relationships

Many to many

Example [https://api.facebook.com/method/links.getStats?urls=Microservices

Slide 11/42, Micro-servies
Martin Fowler, Microservices, youtube
Rick E. Osowski, Microservices in action, Part 1: Introduction to microservices, IBM developerworks

Chương 22

License

View online http://magizbox.com/training/computer$_science/site/licenses/$

Licenses More Licenses

More Open Source Licenses Choose A License Top 20 Open Source Licenses

Chương 23

Semantic Web

View online http://magizbox.com/training/semantic$_w$*eb*/*site*/

The Semantic Web is an extension of the Web through standards by the World Wide Web Consortium (W3C). The standards promote common data formats and exchange protocols on the Web, most fundamentally the Resource Description Framework (RDF).

According to the W3C, "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries". The term was coined by Tim Berners-Lee for a web of data that can be processed by machines. While its critics have questioned its feasibility, proponents argue that applications in industry, biology and human sciences research have already proven the validity of the original concept.

23.1 Web 3.0

Tim Berners-Lee has described the semantic web as a component of "Web 3.0".

People keep asking what Web 3.0 is. I think maybe when you've got an overlay of scalable vector graphics – everything rippling and folding and looking misty – on Web 2.0 and access to a semantic Web integrated across a huge space of data, you'll have access to an unbelievable data resource . . .

—Tim Berners-Lee, 2006

"Semantic Web" is sometimes used as a synonym for "Web 3.0", though the definition of each term varies.

23.2 RDF

23.3 SPARQL

SPARQL (pronounced "sparkle", a recursive acronym for SPARQL Protocol and RDF Query Language) is an RDF query language, that is, a semantic query language for databases, able to retrieve and manipulate data stored in Resource Description Framework (RDF) format. It was made a standard by the RDF Data Access Working Group (DAWG) of the World Wide Web Consortium, and is recognized as one of the key technologies of the semantic web. On 15 January

2008, SPARQL 1.0 became an official W3C Recommendation, and SPARQL 1.1 in March, 2013.

SPARQL allows for a query to consist of triple patterns, conjunctions, disjunctions, and optional patterns.

A SPARQL query

Anatomy of a query

SPARQL has four query forms. These query forms use the solutions from pattern matching to form result sets or RDF graphs. The query forms are:

SELECT Returns all, or a subset of, the variables bound in a query pattern match. CONSTRUCT Returns an RDF graph constructed by substituting variables in a set of triple templates. ASK Returns a boolean indicating whether a query pattern matches or not. DESCRIBE Returns an RDF graph that describes the resources found. Example

Query Result Data filename: ex008.rq

PREFIX ab: <http://learningsparql.com/ns/addressbook>

SELECT ?person WHERE ?person ab:homeTel "(229) 276-5135" Offline query example GET CRAIG EMAILS PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns> PREFIX owl: <http://www.w3.org/2002/07/owl> PREFIX xsd: <http://www.w3.org/2001/XMLSchema> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema> PREFIX : <http://www.semanticweb.org/lananh/ontologies/2016/10/untitled-ontology-3>

SELECT ?craigEmail WHERE :craig :email ?craigEmail . Online query example PREFIX ab: <http://learningsparql.com/ns/addressbook>

SELECT ?craigEmail WHERE ab:craig ab:email ?craigEmail . Query in dbpedia.org Example

SELECT * WHERE ?a ?b ?c . LIMIT 20

Phần IV

Khoa học dữ liệu

Chương 24

Data Science with Python

View online http://magizbox.com/training/ml$_{d}ata_{p}ython/site/$

The ability to analyze data with Python is critical in data science. Learn the basics, and move on to create stunning visualizations.

24.1 Get Started

Get Started with Ubuntu Requirements

numpy, scipy matplotlib pandas scikit-learn ipython Install pip

sudo apt-get install python-pip Install numpy scipy

sudo apt-get install python-numpy python-scipy python-matplotlib python-pandas python-sympy python-nose Install scikit-learn

pip install jupyter ipython pip install -U scikit-learn

24.2 Data Transformation

DataFrame is a 2-dimensional labeled data structure with columns of potentially different types. You can think of it like a spreadsheet or SQL table, or a dict of Series objects. It is generally the most commonly used pandas object

Create data frame Create new data frame from lists

import pandas as pd students = pd.DataFrame('name' : ["Kate", "John", "Tom", "Mark"], 'age' : [20, 21, 19, 18]) age name 0 20 Kate 1 21 John 2 19 Tom 3 18 Mark Load dataframe Load dataframe from datasets

import pandas as pd from sklearn import datasets iris$_{d}ata = datasets.load_{i}ris()iris = pd.DataFrame(data = iris_{d}ata.data, columns = iris_{d}ata.feature_{n}ames)irisSelectionSelectbycolumnindex$

students.iloc[1:3, :] age name 1 21 John 2 19 Tom Filter students = pd.DataFrame('math' : [90, 80, 95, 50], 'physic' : [20, 50, 95, 60]) math physic 0 90 20 1 80 50 2 95 95 3 50 60 students[students['math'] > 85] math physic 0 90 20 2 95 95

students[students['math'] == students['physic']] math physic 2 95 95 Create new column students = pd.DataFrame('name' : ["Kate", "John", "Tom", "Mark"], 'age' : [20, 21, 19, 18]) students["birthyear"] = students.apply(lambda row: 2016 - row['age'], axis=1) students["birthyear"] = 2016 - students["age"]

age name birthyear 0 20 Kate 1996 1 21 John 1995 2 19 Tom 1997 3 18 Mark 1998 Delete column students = pd.DataFrame('name' : ["Kate", "John", "Tom", "Mark"], 'age' : [20, 21, 19, 18]) students = students.drop('age', 1) References Wes McKinney, 10-minute tour of pandas: video, notebook DataFrame, Intro to Data Structures

24.3 Data Preperation

Normalization Example

import numpy from sklearn.preprocessing import normalize matrix = numpy.arange(0,27,3).reshape(3,3).a

array([[0., 3., 6.], [9., 12., 15.], [18., 21., 24.]])

normed$_matrix = normalize(matrix, axis = 1, norm = `l1')$

[[0. 0.33333333 0.66666667] [0.25 0.33333333 0.41666667] [0.28571429 0.33333333 0.38095238]] Label Encoder Encode labels (categorical variables) with value between 0 and n$_classes - 1$.

import sklearn le = sklearn.preprocessing.LabelEncoder() le.fit([“paris”, “paris”, “tokyo”, “amsterdam”]) LabelEncoder() list(le.classes$_)[`amsterdam', `paris', `tokyo']le.transform([ıtokyoȷ, ıto$
$dimensionalnumpyarrayinpythonlessverbose?sklearn.preprocessing.LabelEncoder$

24.4 Data IO

This post shows how to import data to Python from numerous resources

CSV Read a csv file from local or from a server

import numpy as np import pandas as pd read data df = pd.read$_csv(ıdata.csvȷ, header =$ $0)writedatadf.to_csv(ıdata.csvȷ, header = 1, index = False)Excelimportpandasaspdreaddatadf =$ $pd.read_excel(ıdata.xlsȷ)writedatadf = pd.to_excel(ıdata.xlsȷ, index = False)Sqliteimportsqlite3$

DB$_NAME = ıdb.sqlite3ȷSELECT_QUERY = ıSELECTpage_id, typeFROMservice_pageȷconnecttosqlite$
$sqlite3.connect(DB_NAME)excutequerycursor = db_connector.execute(SELECT_QUERY)returndatasetdat$
$cursor.fetchall()Referencespandas.read_excelpandas.read_sqlitesqlite3.read_sqlite$

24.5 Numpy

NumPy Use the following import convention:

import numpy as np Creating Arrays a = np.array([1, 2, 3]) b = np.array([(1.5, 2, 3), (4, 5, 6)], dtype=float) c = np.array([[(1.5, 2, 3), (4, 5, 6)], [(3, 2, 1), (4, 5, 6)]], dtype=float) Initial Placeholders Create an array of zeros np.zeros((3, 4)) array([[0., 0., 0., 0.], [0., 0., 0., 0.], [0., 0., 0., 0.]]) Create an array of ones np.ones((2, 3, 4), dtype=np.int16)

array([[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]],

[[1, 1, 1, 1], [1, 1, 1, 1], [1, 1, 1, 1]]], dtype=int16) Create an array of evenly spaced values (step value) np.arange(10, 25, 5) array([10, 15, 20]) Create an array of evenly spaced values (number of samples) np.linspace(0, 2, 9) array([0. , 0.25, 0.5 , 0.75, 1. , 1.25, 1.5 , 1.75, 2.]) Create a constant array np.full((2, 2), 7) C:2-packages.py:301: FutureWarning: in the future, full((2, 2), 7) will return an array of dtype('int32') format(shape, fill$_value, array(fill_value).dtype), FutureWarning)array([[7., 7.], [7., 7.]$
$np.array([(1, 2), (3, 4)])b = np.array([(5, 6), (7, 8)])np.save(`my_array', a)np.savez(`arrays', a, b)np.load(`arr$
$ȷ, ı)array([[1., 2., 3.], [4., 5., 6.]])a = np.array([(1.5, 2, 3), (4, 5, 6)], dtype = float)np.savetxt(ımyarray.txtȷ, a,$
$ȷı)DataTypesSigned64-bitintegertypesnp.int64Stardarddouble-precisionfloatingpointnp.float32Comple$
$lengthstringtypenp.string_{Fixed-lengthunicodetypenp.unicode_numpy.unicode_InspectingYourArraya=np.array([(1.5,2,3),(4}$

a = np.random.random((3, 3)) a array([[0.07989823, 0.4180309 , 0.83932547], [0.06318651, 0.20509151, 0.08262809], [0.64938826, 0.531026 , 0.38633983]]) select the element at the 2nd index a[2] array([0.64938826, 0.531026 , 0.38633983]) select the element at row 0 column 2 a[1][2] a[1, 2] 0.0826280893797228 Slicing

select items at index 0 and 1 a[0:2] array([[0.07989823, 0.4180309 , 0.83932547], [0.06318651, 0.20509151, 0.08262809]]) select items at rớ 0 and 1 in column 1 a[0:2, 1] array([0.4180309 , 0.20509151]) select all items at row 0 a[1, ...] a[1,] array([0.06318651, 0.20509151, 0.08262809]) reversed array a a[::-1] array([[0.64938826, 0.531026 , 0.38633983], [0.06318651, 0.20509151, 0.08262809], [0.07989823, 0.4180309 , 0.83932547]]) Boolean indexing

select elements from a less than 0.5 a[a $<$ 0.5] array([0.07989823, 0.4180309 , 0.06318651, 0.20509151, 0.08262809, 0.38633983]) Fancy indexing

select elements (1,0), (0,1), (1, 2) and (0,0) a[[1, 0, 1, 0], [0, 1, 2, 0]] array([0.06318651, 0.4180309 , 0.08262809, 0.07989823]) select a subset of the matrix's rows and columns a[[1, 0, 1, 0]][:, [0, 1, 2, 0]] array([[0.06318651, 0.20509151, 0.08262809, 0.06318651], [0.07989823, 0.4180309 , 0.83932547, 0.07989823], [0.06318651, 0.20509151, 0.08262809, 0.06318651], [0.07989823, 0.4180309 , 0.83932547, 0.07989823]]) Array Manipulation Transposing Array a = np.random.random((2, 3)) a array([[0.57430709, 0.64401188, 0.12761183], [0.0726823 , 0.7951682 , 0.54114093]]) permulate array dimensions i = np.transpose(a) i array([[0.57430709, 0.0726823], [0.64401188, 0.7951682], [0.12761183, 0.54114093]]) permulate array dimensions i.T array([[0.57430709, 0.64401188, 0.12761183], [0.0726823 , 0.7951682 , 0.54114093]]) Changing Array Shape flatten the array a.ravel() array([0.57430709, 0.64401188, 0.12761183, 0.0726823 , 0.7951682 , 0.54114093]) reshape, but don't change data a.reshape(3, -2) array([[0.57430709, 0.64401188], [0.12761183, 0.0726823], [0.7951682 , 0.54114093]]) Adding/Removing Elements return a new array with shape (2, 6) a.resize(2, 3) a array([[0.57430709, 0.64401188, 0.12761183], [0.0726823 , 0.7951682 , 0.54114093]]) append items to an array h = np.random.random((2, 3)) print "h:", h g = np.random.random((2, 3)) print "g:", g np.append(h, g) h: [[0.67964404 0.09256795 0.90630423] [0.52906489 0.51567697 0.95132012]] g: [[0.03126344 0.84908154 0.74228134] [0.40333143 0.28595213 0.68416838]] array([0.67964404, 0.09256795, 0.90630423, 0.52906489, 0.51567697, 0.95132012, 0.03126344, 0.84908154, 0.74228134, 0.40333143, 0.28595213, 0.68416838]) insert items in an array a = np.random.random((1, 3)) print "a:", a np.insert(a, 1, 0.5) a: [[0.76135438 0.30331334 0.91866363]] array([0.76135438, 0.5 , 0.30331334, 0.91866363]) delete items from an array a = np.random.random((1, 3)) print "a:", a np.delete(a, [1]) a: [[0.1034073 0.93066432 0.49608264]] array([0.1034073 , 0.49608264]) Combining Arrays concatenate arrays a = np.random.random((1, 3)) print a b = np.random.random((1, 3)) print b np.concatenate((a, b), axis=0) [[0.34496986 0.59502574 0.43416152]] [[0.98921435 0.68832237 0.44286195]] array([[0.34496986, 0.59502574, 0.43416152], [0.98921435, 0.68832237, 0.44286195]]) stack arrays vertically (row-wise) a = np.random.random((1, 3)) print a b = np.random.random((2, 3)) print b np.vstack((a, b)) equivalent to np.r$_{[}a, b][[0.787938410.99234010.96372077]][[0.755370830.097813910.25327948]$
$wise)a = np.random.random((3, 1))printab = np.random.random((3, 2))printbnp.hstack((a, b))[[0.33728008$
$np.random.random((3, 4))printa[[0.642778160.759355990.649272470.80253242][0.876306640.197489310.51895$
$NumpyBasics$

24.6 Data Wrangling

Learn about data wrangling with pandas

Tiny Data A foundation for wrangling in pandas

Create DataFrames Specify values for each column

import pandas as pd

df = pd.DataFrame(“a”: [4, 5, 6], “b”: [7, 8, 9], “c”: [10, 11, 12] , index=[1, 2, 3]) df a b c

1 4 7 10

2 5 8 11

3 6 9 12

Specify values for each row

df = pd.DataFrame([[4, 5, 6], [7, 8, 9], [10, 11, 12]], index=[1, 2, 3], columns=[“a”, “b”, “c”]) df a b c

1 4 5 6

2 7 8 9

3 10 11 12

Create DataFrame with a MultiIndex

df = pd.DataFrame(“a”: [4, 5, 6], “b”: [7, 8, 9], “c”: [10, 11, 12]) index = pd.MultiIndex.from$_tuples([(‘d', 1), (‘d', 2), (‘e', 2)], names = [‘n', ‘v'])dfabc$

0 4 7 10

1 5 8 11

2 6 9 12

Reshaping Data melt “Unpivots” a DataFrame from wide format to long format, optionally leaving identifier variables set.

import pandas as pd

df = pd.DataFrame(“a”: [4, 5], “b”: [7, 8], “c”: [10, 11]) df a b c

0 4 7 10

1 5 8 11

pd.melt(df) variable value

0 a 4

1 a 5

2 b 7

3 b 8

4 c 10

5 c 11

pivot Reshape data (produce a “pivot” table) based on column values. Uses unique values from index / columns to form axes of the resulting DataFrame.

df = pd.DataFrame(‘foo’: [‘one’,‘one’,‘one’,‘two’,‘two’,‘two’], ‘bar’: [‘A’, ‘B’, ‘C’, ‘A’, ‘B’, ‘C’], ‘baz’: [1, 2, 3, 4, 5, 6]) df bar baz foo

0 A 1 one

1 B 2 one

2 C 3 one

3 A 4 two

4 B 5 two

5 C 6 two

df.pivot(index=‘foo’, columns=‘bar’, values=‘baz’) bar A B C

foo

one 1 2 3

two 4 5 6

df.pivot(index='foo', columns='bar')['baz'] bar A B C
foo
one 1 2 3
two 4 5 6
concat Append rows of DataFrames
df1 = pd.DataFrame([['a', 1], ['b', 2]], columns=['letter', 'number']) df1 letter number
0 a 1
1 b 2
df2 = pd.DataFrame([['c', 3], ['d', 4]], columns=['letter', 'number']) pd.concat([df1, df2]) letter number
0 a 1
1 b 2
0 c 3
1 d 4
Append columns of DataFrames
df1 = pd.DataFrame([['a', 1], ['b', 2]], columns=['letter', 'number']) df1 letter number
0 a 1
1 b 2
df2 = pd.DataFrame([['bird', 'polly'], ['monkey', 'george']], columns=['animal', 'name']) df2 animal name
0 bird polly
1 monkey george
pd.concat([df1, df2], axis=1) letter number animal name
0 a 1 bird polly
1 b 2 monkey george
sort df = pd.DataFrame([['a', 10, 1], ['b', 10, 5], ['c', 30, 3]], columns=['name', 'age', 'score']) df name age score
0 a 10 1
1 b 10 5
2 c 30 3
order rows by values of a column (low to high)
df.sort$_values('age')nameagescore$
0 a 10 1
1 b 10 5
2 c 30 3
order rows by values of a column (high to low)
df.sort$_values('age', ascending = False)nameagescore$
2 c 30 3
0 a 10 1
1 b 10 5
order rows by values of two column
df.sort$_values(['age', 'score'], ascending = [False, False])nameagescore$
2 c 30 3
1 b 10 5
0 a 10 1
sort the index of a DataFrame
df.sort$_index()nameagescore$
0 a 10 1

1 b 10 5
2 c 30 3
Reset index of DataFrame to row numbers, moving index to columns
df.reset$_{i}ndex()indexnameagescore$
0 0 a 10 1
1 1 b 10 5
2 2 c 30 3
drop drop columns from DataFrame
df.drop([‘age’, ‘score’], axis=1) name 0 a 1 b 2 c

24.7 Visualization

An introduction about data visualization techniques using Matplotlib and Seaborn.
Gallery line graph Line Graph
bar graph Bar Graph
pie graph Pie Graph
sratter plot Scatter Plot
References Patterns: The Data Visualisation Catalogue

Chương 25

Trí tuệ nhân tạo

View online http://magizbox.com/training/ai/site/

Artificial intelligence (AI) is the intelligence exhibited by machines or software. It is also the name of the academic field of study which studies how to create computers and computer software that are capable of intelligent behavior. Major AI researchers and textbooks define this field as "the study and design of intelligent agents", in which an intelligent agent is a system that perceives its environment and takes actions that maximize its chances of success. John McCarthy, who coined the term in 1955, defines it as "the science and engineering of making intelligent machines".

25.1 Autonomous Agents

limited ability to perceive its environment process the environment and calculate an action no global plan / leader Vehicles

Action / Selection Steering Locomotion Steering Behavior 1 2

Steering = Desired - Velocity

Seek Flow Filed Following Path Following Group Steering https://github.com/shiffman/The-Nature-of-Code-Examples/tree/master/chp06$_{a}gents$

Massive Battle: Coordinated Movement of Autonomous Agents

Craig Reynolds, Steering Behaviors For Autonomous Characters

25.2 Cellular Automator

https://www.youtube.com/watch?v=DKGodqDs9sAindex=1list=PLRqwX-V7Uu6YrWXvEQFOGbCt6cX8

Cellular Automata

Grid of cell Each cell has state, neighborhood cell state at time t defined by a function of neighborhood states at time t-1 Elementary Cellular Automata

25.3 Fractal

L-System

25.4 The Pac-Man project

Today I found an interesting AI project - The Pac-Man

http://ai.berkeley.edu/images/pacman$_game.gif$

Here is the project overview

The Pac-Man projects were developed for UC Berkeley's introductory artificial intelligence course, CS 188. They apply an array of AI techniques to playing Pac-Man. However, these projects don't focus on building AI for video games. Instead, they teach foundational AI concepts, such as informed state-space search, probabilistic inference, and reinforcement learning. These concepts underly real-world application areas such as natural language processing, computer vision, and robotics. We designed these projects with three goals in mind. The projects allow students to visualize the results of the techniques they implement. They also contain code examples and clear directions, but do not force students to wade through undue amounts of scaffolding. Finally, Pac-Man provides a challenging problem environment that demands creative solutions; real-world AI problems are challenging, and Pac-Man is too. In our course, these projects have boosted enrollment, teaching reviews, and student engagement. The projects have been field-tested, refined, and debugged over multiple semesters at Berkeley. We are now happy to release them to other universities for educational use. In the next part of this post, I will show my works on this project

Project 1: Search in Pacman

[caption id="" align="alignleft" width="231"]DFS[/caption]

[caption id="" align="alignleft" width="233"]BFS[/caption]

Chương 26

Học máy

- Vấn đề với HMM và CRF?
- Học MLE và MAP?

View online http://magizbox.com/training/machinelearning/site/

Machine learning is a branch of science that deals with programming the systems in such a way that they automatically learn and improve with experience. Here, learning means recognizing and understanding the input data and making wise decisions based on the supplied data.

We can think of machine learning as approach to automate tasks like predictions or modelling. For example, consider an email spam filter system, instead of having programmers manually looking at the emails and coming up with spam rules. We can use a machine learning algorithm and feed it input data (emails) and it will automatically discover rules that are powerful enough to distinguish spam emails.

Machine learning is used in many application nowadays like spam detection in emails or movie recommendation systems that tells you movies that you might like based on your viewing history. The nice and powerful thing about machine learning is: It learns when it gets more data and hence it gets more and more powerful the more data we give them.

Có bao nhiêu thuật toán Machine Learning?

Có rất nhiều thuật toán Machine Learning, bài viết [Điểm qua các thuật toán Machine Learning hiện đại](https://ongxuanhong.wordpress.com/2015/10/22/diem-qua-cac-thuat-toan-machine-learning-hien-dai/) của Ông Xuân Hồng tổng hợp khá nhiều thuật toán. Theo đó, các thuật toán Machine Learning được chia thành các nhánh lớn như ‘regression‘, ‘bayesian‘, ‘regularization‘, ‘decision tree‘, ‘instance based‘, ‘dimesionality reduction‘, ‘clustering‘, ‘deep learning‘, ‘neural networks‘, ‘associated rule‘, ‘ensemble‘... Ngoài ra thì còn có các cheatsheet của [sklearn](http://scikit-learn.org/stable/tutorial/machine$_{l}earning_{m}ap/index.html$).

Việc biết nhiều thuật toán cũng giống như ra đường mà có nhiều lựa chọn về xe cộ. Tuy nhiên, quan trọng là có task để làm, sau đó thì cập nhật SOTA của task đó để biết các công cụ mới.

Xây dựng model cần chú ý điều gì?

Khi xây dựng một model cần chú ý đến vấn đề tối ưu hóa tham số (có thể sử dụng [GridSearchCV](sklearn.model$_{s}election.GridSearchCV$))

Bài phát biểu này có vẻ cũng rất hữu ích [PYCON UK 2017: Machine learning libraries you'd wish you'd known about](https://www.youtube.com/watch?v=nDF7$_8FOhpI).CØcpØn$

* [DistrictDataLabs/yellowbrick](https://github.com/DistrictDataLabs/yellowbrick) (giúp visualize model được train bởi sklearn) * [marcotcr/lime](https://github.com/marcotcr/lime) (giúp inspect classifier) * [TeamHG-Memex/eli5](https://github.com/TeamHG-Memex/eli5) (cũng giúp inspect classifier, hỗ trợ nhiều model như xgboost, crfsuite, đặc biệt có TextExplainer sử dụng thuật toán từ eli5) * [rhiever/tpot](https://github.com/rhiever/tpot) (giúp tối ưu hóa pipeline) * [dask/dask](https://github.com/dask/dask (tính toán song song và lập lịch)

Ghi chú về các thuật toán trong xử lý ngôn ngữ tự nhiên tại [underthesea.flow/wiki](https://github.com/magizbox/underthesea.flow/wiki/Develop)

Framework để train, test hiện tại vẫn rất thoải mái sklearn. tensorboard cung cấp phần log cũng khá hay.

[Câu trả lời hay](https://www.quora.com/What-are-the-most-important-machine-learning-techniques-to-master-at-this-time/answer/Sean-McClure-3?srid=5O2u) cho câu hỏi [Những kỹ thuật machine learning nào quan trọng nhất để master?](https://www.quora.com/What-are-the-most-important-machine-learning-techniques-to-master-at-this-time), đặc biệt là dẫn đến bài [The State of ML and Data Science 2017](https://www.kaggle.com/surveys/2017) của Kaggle.

Tài liệu học PGM

[Playlist youtube](https://www.youtube.com/watch?v=WPSQfOkb1M8amp;list=PL50E6E80E8525B59C khóa học Probabilistic Graphical Models của cô Daphne Koller. Ngoài ra còn có một [tutorial](http://mensxmachina.org/files/software/demos/bayesnetdemo.html) dở hơi ở đâu về tạo Bayesian network

[Chưa biết] Tại sao Logistic Regression lại là Linear Model?

Trong quyển Deep Learning, chương 6, trang 165, tác giả có viết

"' Linear models, such as logistic regression and linear regression, are appealing because they can be t eciently and reliably, either in closed form or with convex optimization "'

Mình tự hỏi tại sao logistic regression lại là linear, trong khi nó có sử dụng hàm logit (nonlinear)? Tìm hiểu hóa ra cũng có bạn hỏi giống mình trên [stats.stackexchange.com](https://stats.stackexchange.com/questions/93569/why-is-logistic-regression-a-linear-classifier). Ngoài câu trả lời trên stats.stackexchange, đọc một số cái khác [Generalized Linear Models, SPSS Statistics 22.0.0](https://www.ibm.com/support/knowl $IntroductiontoGeneralizedLinearModels$, $AnalysisofDiscreteData$, $PennsylvaniaStateUniversity$]($https$ $//onlinecourses.science.psu.edu/stat504/node/216)cngvnchahiulm$.

Hiện tại chỉ hiểu là các lớp model này chỉ có thể hoạt động trên các tập linear separable, có lẽ do việc map input x, luôn có một liên kết linear $latexwx$, trước khi đưa vào hàm non-linear.

Các tập dữ liệu thú vị

Iris dataset: dữ liệu về hoa iris

Là một ví dụ cho bài toán phân loại

Weather problem: dữ liệu thời tiết. Có thể tìm được ở trong quyển Data Mining: Practical Machine Learning Tools and Techniques

Là một ví dụ cho bài toán cây quyết định

Deep Learning

Tài liệu Deep Learning

Lang thang thế nào lại thấy trang này [My Reading List for Deep Learning!](https://www.microsoft.com/en-us/research/wp-content/uploads/2017/02/DL$_{R}eading_{L}ist.pdf)camtanhMicrosoft.TrongØ,(Øngnhin)cDee$

Các layer trong deep learning [2]

Sparse Layers

[**nn.Embedding**](http://pytorch.org/docs/master/nn.htmlembedding) ([hướng dẫn](http://pytorch.org/tutorials/beginner/nlp/word$_embeddings_tutorial.html))grepcode : [Shawn1993/cnn-text-classification-pytorch](https://github.com/Shawn1993/cnn-text-classification-pytorch/blob/master/model.pyL18)Œngvaitrnhmtlookuptable, mapmtwordvidenseve$

Convolution Layers

[**nn.Conv1d**](http://pytorch.org/docs/master/nn.htmlconv1d), [**nn.Conv2d**](http://pytorch.org/ [**nn.Conv3d**](http://pytorch.org/docs/master/nn.htmlconv3d) [1]$grepcode : [Shawn1993/cnn-text-classification-pytorch](https://github.com/Shawn1993/cnn-text-classification-pytorch/blob/master/model.pyL20-L24), [galsang/CNN-sentence-classification-pytorch](https://github.com/galsang/CNN-sentence-classification-pytorch/blob/master/model.pyL36-L38)$

Các tham số trong Convolution Layer

* ‘kernel$_size‘(haylfiltersize)$

Đối với NLP, kernel$_sizethngbngregion_size*word_dim(Øiviconv1d)hay(region_size, word_dim)Øiviconv2d$

<small>Quá trình tạo feature map đối với region size bằng 2</small> `![](https://media.giphy.com/media/l2QE2y1UQP7vIgiti/giphy.gif)`

* ‘in$_channels‘, ‘out_channels‘(lslng‘featuremaps‘)$

Kênh (channels) là các cách nhìn (view) khác nhau đối với dữ liệu. Ví dụ, trong ảnh thường có 3 kênh RGB (red, green, blue), có thể áp dụng convolution giữa các kênh. Với văn bản cũng có thể có các kênh khác nhau, như khi có các kênh sử dụng các word embedding khác nhau (word2vec, GloVe), hoặc cùng một câu nhưng biểu diễn ở các ngôn ngữ khác nhau.

* ‘stride‘

Định nghĩa bước nhảy của filter.

`![](http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/11/Screen-Shot-2015-11-05-at-10.18.08-AM-1024x251.png)`

Hình minh họa sự khác biệt giữa các feature map đối với stride=1 và stride=2. Feature map đối với stride = 1 có kích thước là 5, feature map đối với stride = 3 có kích thước là 3. Stride càng lớn thì kích thước của feature map càng nhỏ.

Trong bài báo của Kim 2014, ‘stride = 1‘ đối với ‘nn.conv2d‘ và ‘stride = word$_dim‘Øivi‘nn.conv1d‘$

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014,

`![](http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/11/Screen-Shot-2015-11-06-at-8.03.47-AM.png)`

| Description | Values | |——————–|—————–| | input word vectors | Google word2vec | | filter region size | (3, 4, 5) | | feature maps | 100 | | activation function | ReLU | | pooling | 1-max pooling | | dropout rate | 0.5 | | $latexlamp;s = 22$ norm constraint | 3 |

Đọc thêm:

* [Lecture 13: Convolutional Neural Networks (for NLP). CS224n-2017](http://web.stanford.edu/class/cs2 2017-lecture13-CNNs.pdf) * [DeepNLP-models-Pytorch - 8. Convolutional Neural Networks](https://nbviewer.jupyter.org/github/DSKSD/DeepNLP-models-Pytorch/blob/master/notebooks/08.CNN-for-Text-Classification.ipynb) * [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Zhang 2015](https://arxiv.org/pdf/1510.03820.pdf)

BTS

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy. 24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về

Deep Learning [tại đây](https://docs.google.com/document/d/1KxDrw5s6uYHNLda7t0rhp0RM$_TlUGxydQ$-$Qi1JOPFr8/edit?usp = sharing$)

[1] : $[UnderstandingConvolutionalNeuralNetworksforNLP](http : //www.wildml.com/$2015/11/$unde$-$convolutional-neural-networks-for-nlp)$[2] : $http : //pytorch.org/docs/master/nn.html$

26.1 Machine Learning Process

The good life is a process, not a state of being. It is a direction not a destination.

Carl Rogers

I searched a framework fit for every data mining task, I found a good one from an article of Oracle.

And here is my summary. The data mining process has 4 steps:

Step 1. Problem Definition

This initial phase of a data mining project focuses on understanding the project objectives and requirements. Once you have specified the project from a business perspective, you can formulate it as a data mining problem and develop a preliminary implementation plan.

Step 2. Data Gathering Preparation

The data understanding phase involves data collection and exploration. As you take a closer look at the data, you can determine how well it addresses the business problem. You might decide to remove some of the data or add additional data. This is also the time to identify data quality problems and to scan for patterns in the data.

Data Access Data Sampling

Data Transformation

Data in the real world is dirty [3]. They are often incomplete (lacking attribute values, lacking certain attributes of interest, or containing only aggregate data), noisy (containing errors or outliers), inconsistent (containing discrepancies in codes or names). Step 3. Model Building In this phase, you select and apply various modeling techniques and calibrate the parameters to optimal values. If the algorithm requires data transformations, you will need to step back to the previous phase to implement them

Create Model Test Model

Evaluate Interpret Model

Some important questions [2]:

Is at least one of predictors useful in predicting the response? (F-statistics) Do all the predictors help to explain Y, or is only a subset of the predictors useful? (all subsets or best subsets) How well does the model fit the data? Given a set of predictor values, what response value should we predict, and how accurate is our prediction? Step 4. Knowledge Deployment Knowledge deployment is the use of data mining within a target environment. In the deployment phase, insight and actionable information can be derived from data. Model Apply Custom Reports External Applications References The Data Mining Process, Oracle Trevor Hastie and Rob Tibshirani, Model Selection and Qualitative Predictors, URL:https://www.youtube.com/watch?v=3T6RXmIHbJ4 Nguyen Hung Son, Data cleaning and Data preprocessing, URL:http://www.mimuw.edu.pl/ son/datamining/DM/4-preprocess.pdf

26.1.1 Problem Definition

This initial phase of a data mining project focuses on understanding the project objectives and requirements. Once you have specified the project from a business perspective, you can formulate it as a data mining problem and develop a preliminary implementation plan.

For example, your business problem might be: "How can I sell more of my product to customers?" You might translate this into a data mining problem such as: "Which customers are most likely to purchase the product?" A model that predicts who is most likely to purchase the product must be built on data that describes the customers who have purchased the product in the past. Before building the model, you must assemble the data that is likely to contain relationships between customers who have purchased the product and customers who have not purchased the product. Customer attributes might include age, number of children, years of residence, owners/renters, and so on.

26.1.2 Data Gathering

The data understanding phase involves data collection and exploration. As you take a closer look at the data, you can determine how well it addresses the business problem. You might decide to remove some of the data or add additional data. This is also the time to identify data quality problems and to scan for patterns in the data.

The data preparation phase covers all the tasks involved in creating the case table you will use to build the model. Data preparation tasks are likely to be performed multiple times, and not in any prescribed order. Tasks include table, case, and attribute selection as well as data cleansing and transformation. For example, you might transform a DATE$_O F_B IRTHcolumntoAGE; youmightinserttheaverageincomeincaseswhere$

Additionally you might add new computed attributes in an effort to tease information closer to the surface of the data. For example, rather than using the purchase amount, you might create a new attribute: "Number of Times Amount Purchase Exceeds $500ina12monthtimeperiod.''Customerswhofrequentlymakelargepurchasesmayalsoberel$

Thoughtful data preparation can significantly improve the information that can be discovered through data mining.

Data Sources Open Data

wikipedia dumps: https://dumps.wikimedia.org/other/pagecounts-raw/

26.1.3 Data Preprocessing

The quality of the data and the amount of useful information it contains affect greatly how well an algorithm can learn. Hence, it is important to preprocess the dataset before using it. The most common preprocessing steps are: removing missing values, converting categorical data into shape suitable for machine learning algorithm and feature scaling.

Missing Data Sometimes the samples in the dataset are missing some values and we want to deal with these missing values before passing it to the machine learning algorithm. There are a number of strategies we can follow

Remove samples with missing values: This approach is by far the most convenient but we may end up removing too many samples and by that we would be losing valuable information that can help the machine learning algorithm.

Imputing missing values: Instead of removing the entire sample we use interpolation to estimate the missing values. For example, we could substitute a missing value by the mean of the entire column. Categorical Data In general, features can be numerical (e.g. price, length, width, etc…) or categorical (e.g. color, size, etc..). Categorical features are further split into nominal and ordinal features.

Ordinal features can be sorted and ordered. For example, size (small, medium, large), we can order these sizes large > medium > small. While nominal features do not have an order for example, color, it doesn't make any sense to say that red is larger than blue.

Most machine learning algorithm require that you convert categorical features into numerical values. One solution would to assign each value a different number starting from zero. (e.g. small à 0 ,medium à 1 ,large à 2)

This works well for ordinal features but might cause problems with nominal features (e.g. blue à 0, white à 1, yellow à 2) because even though colors are not ordered the learning algorithm will assume that white is larger than blue and yellow is larger than white and this is not correct.

To get around this problem is to use one-hot encoding, the idea is to create a new feature for each unique value of the nominal feature.

In the above example, we converted the color feature into three new features Red, Green, Blue and we used binary values to indicate the color. For example, a sample with "Red" color is now encoded as (Red=1, Green=0, Blue=0)

Feature Scaling Why have we do Feature Scaling?

We have to predict the house prices base on 2 features:

House sizes (feet2) Number of bedrooms in the house And we relized that house sizes are about 1000 times the number of bedrooms. When features differ by orders of magnitude, first performing feature scaling can make gradient descent converge much more quickly.

Perform Feature Scaling

Subtract the mean value (the average value) of each feature from the dataset. After subtracting the mean, additionally scale (divide) the feature values by their respective "standard deviations." Function: x=xxx=xx where xx is the original feature vector, xx is the mean of that feature vector, and is its standard deviation. Feature Scaling Function implementation in Octave

function [X$_norm, mu, sigma] = featureNormalize(X) X_norm = X; mu = zeros(1, size(X, 2)); sigma = zeros(1, size(X, 2));$

for i = 1:length(mu), mu(i) = mean(X(:,i)); end;

for i = 1:length(sigma), sigma(i) = std(X(:,i)); end;

X$_norm = (X.-mu)./sigma; endRelatedReadingIntroductiontoMachineLearning$

26.1.4 Model Building

In this phase, you select and apply various modeling techniques and calibrate the parameters to optimal values. If the algorithm requires data transformations, you will need to step back to the previous phase to implement them

Create Model Test Model Evaluate Interpret Model Some important questions

Is at least one of predictors useful in predicting the response? (F-statistics) Do all the predictors help to explain Y, or is only a subset of the predictors useful? (all subsets or best subsets) How well does the model fit the data?

Given a set of predictor values, what response value should we predict, and how accurate is our prediction? Create Model First thing first, start with simple and fast model, then you known how difficult the problem is.

One import thing is create a well pipeline for your experiments, it is very helpful in turning features, model selection, save your experiment and write reports.

Feature Selections After train model, some model will give active features (such as CRF), it is clue for you to feature selection. If amount active features is too small compared to amount features, it is the problem. In this case the better way to enhance is try reduce amount of features and see how well this set fit data. Keep in mind the more number of features is, the complex model is, and it will make your model over fitting. Storing the model Number of active features: 5566 (35383) Number of active attributes: 4343 (20722) example after training crf model with python-crfsuite Test Model This phase determines how well the model fit data. See Evaluation for details.

What to do next In an interview Andrew Ng said about building machine learning model

"I often make an analogy to building a rocket ship. A rocket ship is a giant engine together with a ton of fuel. Both need to be really big. If you have a lot of fuel and a tiny engine, you won't get off the ground. If you have a huge engine and a tiny amount of fuel, you can lift up, but you probably won't make it to orbit. So you need a big engine and a lot of fuel.

The reason that machine learning is really taking off now is that we finally have the tools to build the big rocket engine — that is giant computers, that's our rocket engine. And the fuel is the data. We finally are getting the data that we need."

We need both big rocket engine and data to make our model works.

Related Reading Inside The Mind That Built Google Brain: On Life, Creativity, And Failure, huffingtonpost.com

26.1.5 Evaluation

Training vs Test Data We typically split the input data into learning and testing datasets. The then run the machine learning algorithm on the learning dataset to generate the prediction model. Later, we use the test dataset to evaluate our model.

It is important that the test data is separate from the one used in training otherwise we will be kind of cheating because may for example the generated model memorizes the data and hence if the test data is also part of the training data then our evaluation scores of the model will be higher than they actually are.

The data is usually split 75

In addition, when splitting the dataset, you need to maintaining class proportions and population statistics otherwise we will have some classes that are under represented in the training dataset and over represented in the test dataset.

For example, you may have 100 sample and a total of 80 samples are labeled with Class-A and the remaining 20 instances are labeled with Class-B. you want to make sure when splitting the data that you maintain this representation.

One way to avoid this problem and to make sure that all classes are represented in both training and testing datasets is stratification. It is the process of rearranging the data as to ensure each set is a good representative of the whole. In our previous example, (80/20 samples), it is best to arrange the data such that in every set, each class comprises around 80:20 ratios of the two classes.

Cross Validation A crucial step when building our machine learning model is to estimate its performance on that that the model hadn't seen before. We want to make sure that the model generalizes well to new unseen data.

One case, the machine learning algorithm has different parameters and we want to tune these parameters to achieve the best performance. (Note: the parameters of the machine learning algorithm are called hyperparameters). Another case, sometimes we want to try out different algorithms and choose the best performing one. Below are some of the techniques used.

Holdout Method We simply split the data into training and testing datasets. We train the algorithm on the training dataset to generate a model. In order, to evaluate different algorithms we use the testing data to evaluate each algorithm.

However, if we reuse the same test dataset over and over again during algorithm selection, the test data has now come part of the training data. Hence, when we use the test data for the final evaluation the generated model is biased towards the test data and the performance score is optimistic.

Holdout Validation As before, we split the data into training and testing dataset. Then, the training data is further split into training and validation sets.

The training data is used to train different models. Then the validation data is used to compute performance of each of them and we select the best one. Finally, the model is then used for the test set to evaluate performance. The next figure illustrates this idea.

However, because we use the validation set multiple times, Holdout validation is sensitive to how we partition the data and that is what K-fold cross validation tries to solve.

K-fold cross validation Initially, we split the data into training and testing dataset. Furthermore, the training dataset is split into K chunks.

Suppose we will use 5-fold cross validation, the training data set is split into 5 chunks and the training phase will take place over 5 iterations. In each iteration we use one chunk as the validation dataset while the rest of the chunk are grouped together to form the training dataset.

This is very similar to Holdout validation except in each iteration the validation data is different and this removed the bias. Each iteration generates a score and the final score is the average score of all iteration. As before we select the best model and use the test data for the final performance evaluation.

Related Readings

Introduction to Machine Learning

26.2 Types of Machine Learning

There are three different types of machine learning: supervised, unsupervised and reinforcement learning. 4

Supervised Learning The goal of supervised learning is to learn a model from labelled training data that allows us to make predictions about future data. For supervised machine learning to work we need to feed the algorithm two things: the input data and our knowledge about it labels).

The spam filter example mentioned earlier is a good example of supervised learning; we have a bunch of emails (data) and we know whether each email is spam or not (labels).

Supervised learning can be divided into two subcategories:

Classification: It is used to predict categories or class labels based on past observations i.e. we have discrete variable you want to distinguish into discrete categorical outcome. For example, in the email spam filter system the output is discrete "spam" or "not spam". Regression: It is used to predict a continuous outcome. For example, to determine the price of houses and how it is affected by the number of rooms in that house. The input data is the house features (no. of rooms, location, size in square feet,) and the output is the price (the continuous outcome). Unsupervised Learning The goal of unsupervised learning is to discover hidden structure or patterns in unlabeled data and it can be divided into two subcategories

Clustering: It is used to organize information into meaningful clusters (subgroups) without having prior knowledge of their meaning. For example, the figure below shows how we can use clustering to organize unlabeled data into groups based on their features.

Dimensionality Reduction (Compression): It is used to reduce a higher dimension data into a lower dimension ones. To put it more clearly consider this example. A telescope has terabytes of data and not all of these data can be stored and so we can use dimensionality reduction to extract the most informative features of these data to be stored. Dimensionality reduction is also a good candidate to visualize data because if you have data in higher dimensions you can compress it to 2D or 3D to easily plot and visualize it.

Reinforcement Learning The goal of reinforcement learning is to develop a system that improves its performance based on the interaction with a dynamic environment and there is a delayed feedback that act as a reward. i.e. reinforcement learning is learning by doing with a delayed reward. A classic example of reinforcement learning is a chess game, the computer decided a series of moves and the reward is the "win" or "lose" at the end the game.

You might think that this is similar to supervised learning where the reward is basically a label for the data but the core difference is this feedback/reward is not the truth but it is a measure of how well the action to achieving a certain goal.

Microsoft Azure Machine Learning 1

Machine Learning Cheat Sheet for scikit-learn 2

DLib C++ Library - Machine Learning Guide 3

Challenges Very much features ($>$ 100) Very much data ($>$ 1e9 items) Text Data, Images, Videos Training Times Accuracy, Over Fitting Machine learning algorithm cheat sheet for Microsoft Azure Machine Learning Studio

Machine Learning Cheat Sheet (for scikit-learn)

DLib C++ Library - Machine Learning Guide

Introduction to Machine Learning

26.3 How to learn a ML Algorithm?

1. Motivation

Each algorithm have its own motivation. It may a simple example to see how it work

2. Problem Definition

Where can we apply this algorithm? How did it work in real world applications

3. Mathematics Representation

Problem Equations, notations

We will discuss about mathematics representation of algorithm, notations we use for problem

4. Algorithm

We will discuss how to solve this mathematics problems

5. Examples

We will apply algorithm with a few examples (1-2 dimension is highly recommended, because we will plot these data and model easily)

In this section, we can see how well (bad) algorithm works with these data

6. Implementation Notice

We will give some notes about implement this algorithm to real world problems. What case we want to apply this algorithm? What case we don't?

7. Quiz

One way to rethink about problem is doing quiz.

8. Exercise

26.4 Machine Learning Algorithms

26.4.1 Linear Regression

Linear Regression In-Out

Input: Continuous Output: Continuous

When to use 1 Econometric Modeling Marketing Mix Model Customer Lifetime Value Examples Ex1. Linear Regression with Boston Dataset

$_{a}uthor'_{=}rain'$

from sklearn.datasets import load$_{b}oston from sklearn.cross_{v}alidation import train_{t}est_{s}plit from sklearn.l$
$load_{b}oston() data = boston['data'] X, y = data[:, : -1], data[:, -1] X_{t}rain, X_{t}est, y_{t}rain, y_{t}est =$
$train_{t}est_{s}plit(X, y, test_{s}ize = 0.3) print boston['DESCR'] clf_{l}inear = LinearRegression() clf_{l}inear.fit(X_{t}ra$
$clf_{l}inear.score(X_{t}est, y_{t}est) - > 0.671 print(clf_{l}inear.coef_{)} print(clf_{l}inear.intercept_{)}$

clf$_{r}idge = Ridge(alpha = 1.0) clf_{r}idge.fit(X_{t}rain, y_{t}rain) 0.674 ridge_{s}core =$
$clf_{r}idge.score(X_{t}est, y_{t}est)$

print y$_{t}est print clf_{l}inear.predict(X_{t}est) print clf_{r}idge.predict(X_{t}est) Ex2. Linear Regression with marketc$
$"notebook"]/python)$

Logistic Regression

In-Out 1 In: continuos Out: True/False 1. Hypothesis Representation h(x)=g(Tx) where g(z)=11+ez h(x)=g(Tx) where g(z)=11+ez g(z)g(z) is sigmoid function or logistic function

h(x)h(x) estimated probability of y=1y=1 given xx

In spam detection problem, h(x)=0.7h(x)=0.7 means it's 70

2. Decision Boundary Logistic Regression

3. Cost Function cost(h(x),y)=ylog(h(x))(1y)log(1h(x)) cost(h(x),y)=ylog(h(x))(1y)log(1h(x)) Loss Function

J()=1mi=1mcost(h(x(i)),y(i))=1mi=1my(i)logh(x(i))+(1y(i))log(1h(x(i))) J()=1mi=1mcost(h(x(i)),y(i))
4. Gradient Descent Gradient

J()j=1mi=1m(h(x(i))y(i))x(i)j J()j=1mi=1m(h(x(i))y(i))xj(i) 5. Predict p(,X)=h(X)0.5 p(,X)=h(X)0.5 6. Regularization 6.1 Feature Mapping Cost Function

mapFeature(x)=1x1x2x21x1x2x22x31x1x52x62 mapFeature(x)=[1x1x2x12x1x2x22x13x1x25x26]
6.2 Cost Function and Gradient Cost Function J()=1mi=1m[y(i)log(h(x(i)))(1y(i))log(1h(x(i)))]+2mj=1n2j J()=1mi=1m[y(i)log(h(x(i)))(1y(i))log(1h(x(i)))]+2mj=1nj2 Gradient

J()j=1mmi=1(h(x(i))y(i))x(i)jJ()j=1mi=1m(h(x(i))y(i))xj(i) for j=0j=0

J()j=(1mmi=1(h(x(i))y(i))x(i)j)+mjJ()j=(1mi=1m(h(x(i))y(i))xj(i))+mj for j1j1

Code Bank Marketing Data Set

import statsmodels.api as sm import pandas as pd from statsmodels.tools.tools import categorical from sklearn.preprocessing import LabelEncoder from sklearn.linear$_m odelimportLogisticR$

def get$_d ata() : returnpd.read_c sv(quot; ./bank/bank-full.csvquot; , header = 0, sep = quot; ; quot;)$

data = get$_d ata()$

data.job = LabelEncoder().fit$_t ransform(data.job)data.marital = LabelEncoder().fit_t ransform(data.m$
$LabelEncoder().fit_t ransform(data.education)data.default = LabelEncoder().fit_t ransform(data.default)$
$LabelEncoder().fit_t ransform(data.housing)data.loan = LabelEncoder().fit_t ransform(data.loan)data.mo$
$LabelEncoder().fit_t ransform(data.month)data.contact = LabelEncoder().fit_t ransform(data.contact)data$
$LabelEncoder().fit_t ransform(data.poutcome)$

X = data.iloc[:, :-1] y = data.iloc[:, -1]

X$_t rain, X_t est, y_t rain, y_t est = train_t est_s plit(X, y, test_s ize = 0.3)$

clf = LogisticRegression() clf.fit(X$_t rain, y_t rain)score = clf.score(X_t est, y_t est)$

print confusion$_m atrix(y_t est, clf.predict(X_t est))$[[11807203][1243311]]$it'stoobad$

Examples Affair Dataset, Logistic Regression with scikit-learn Linear Regression vs Logistic Regression vs Poisson Regression

26.4.2 Classification

Classification

Classification 1 A very familiar example is the email spam-catching system: given a set of emails marked as spam and not-spam, it learns the characteristics of spam emails and is then able to process future email messages to mark them as spam or not-spam.

The technique used in the above example of email spam-catching system is one of the most common machine learning techniques: classification (actually, statistical classification). More precisely it is a supervised statistical classification. Supervised because the system needs to be first trained using already classified training data as opposed to an unsupervised system where such training is not done.

A supervised learning system that performs classification is known as a learner or, more commonly, a classifier.

The classifier is first fed training data in which each item is already labeled with the correct label or class. This data is used to train the learning algorithm, which creates models that can then be used to label/classify similar data.

Formally, given a set of input items, and a set of labels/classes, and training data is the label/class for $latexx_i$, a classifier is a mapping from X to Y $latexf(T, x) = y$.

Binary Classification Algorithms 1 Two-class SVM 100 features, linear model

Two-class Logistic Regression Fast training, linear model Two-class Bayes point machine Fast training, linear model Two-class random forest Accuracy, fast training Two-class boosted decision tree Accuracy, fast training Two-class neural network Accuracy, long training times Multiclass Classification

Introduction 2 In machine learning, multiclass or multinomial classification is the problem of classifying instances into one of the more than two classes (classifying instances into one of the two classes is called binary classification).

While some classification algorithms naturally permit the use of more than two classes, others are by nature binary algorithms; these can, however, be turned into multinomial classifiers by a variety of strategies.

Multiclass classification should not be confused with multi-label classification, where multiple labels are to be predicted for each instance.

Algorithms 1 Multiclass Logistic Regression Multiclass SVM Multiclass Neural Network Multiclass Decision Forest Multiclass Decision Jungle Confusion Matrix sklearn plot confusion matrix with labels 3

import matplotlib.pyplot as plt def plot$_{c}onfusion_{m}atrix(cm, title =' Confusionmatrix', cmap = plt.cm.Blues, labels = None) : fig = plt.figure()ax = fig.add_{s}ubplot(111)cax = ax.matshow(cm)plt.title(title)fig.colorbar(cax)iflabels : ax.set_{x}ticklabels([''] + labels)ax.set_{y}ticklabels([''] + labels)plt.xlabel('Predicted')plt.ylabel('True')plt.show()$

Multilabel Classification

Introduction 1 In machine learning, multi-label classification and the strongly related problem of multi-output classification are variants of the classification problem where multiple target labels must be assigned to each instance. Multi-label classification should not be confused with multiclass classification, which is the problem of categorizing instances into one of more than two classes. Formally, multi-label learning can be phrased as the problem of finding a model that maps inputs x to binary vectors y, rather than scalar outputs as in the ordinary classification problem.

There are two main methods for tackling the multi-label classification problem:[1] problem transformation methods and algorithm adaptation methods. Problem transformation methods transform the multi-label problem into a set of binary classification problems, which can then be handled using single-class classifiers. Algorithm adaptation methods adapt the algorithms to directly perform multi-label classification. In other words, rather than trying to convert the problem to a simpler problem, they try to address the problem in its full form.

Implements Multiclass and multilabel algorithms SVM Multi-label classification

Multiclass classification

sklearn plot confusion matrix with labels

26.4.3 Clustering

Using K-Means to cluster wine dataset Recently, I joined Cluster Analysis course in coursera. The content of first week is about Partitioning-Based Clustering Methods where I learned about some cluster algorithms based on distance such as K-Means, K-Medians and K-Modes. I would like to turn what I learn into practice so I write this post as an excercise of this course.

In this post, I will use K-Means for clustering wine data set which I found in one of excellent posts about K-Mean in r-statistics website.

Meet the data

The wine data set contains the results of a chemical analysis of wines grown in a specific area of Italy. Three types of wine are represented in the 178 samples, with the results of 13 chemical analyses recorded for each sample. The Type variable has been transformed into a categoric variable.

data(wine, package=quot;rattlequot;) head(wine)

gt; Type Alcohol Malic Ash Alcalinity Magnesium Phenols gt; 1 1 14.23 1.71 2.43 15.6 127 2.80 gt; 2 1 13.20 1.78 2.14 11.2 100 2.65 gt; 3 1 13.16 2.36 2.67 18.6 101 2.80 gt; 4 1 14.37 1.95 2.50 16.8 113 3.85 gt; 5 1 13.24 2.59 2.87 21.0 118 2.80 gt; 6 1 14.20 1.76 2.45 15.2 112 3.27 gt; Flavanoids Nonflavanoids Proanthocyanins Color Hue gt; 1 3.06 0.28 2.29 5.64 1.04 gt; 2 2.76 0.26 1.28 4.38 1.05 gt; 3 3.24 0.30 2.81 5.68 1.03 gt; 4 3.49 0.24 2.18 7.80 0.86 gt; 5 2.69 0.39 1.82 4.32 1.04 gt; 6 3.39 0.34 1.97 6.75 1.05 gt; Dilution Proline gt; 1 3.92 1065 gt; 2 3.40 1050 gt; 3 3.17 1185 gt; 4 3.45 1480 gt; 5 2.93 735 gt; 6 2.85 1450 Explore and Preprocessing Data Let's see structure of wine data set

str(wine)

gt; apos;data.frameapos;: 178 obs. of 14 variables: gt; $Type : Factorw/3levelsquot;1quot;,quot;2quot;,quo$ $1111111111...gt$; Alcohol : num 14.2 13.2 13.2 14.4 13.2 ... gt; $Malic : num1.711.782.361.952.591.761.872.151.64$ Ash : num 2.43 2.14 2.67 2.5 2.87 2.45 2.45 2.61 2.17 2.27 ... gt; $Alcalinity : num15.611.218.616.82115.214.617.61416...gt$; Magnesium : int 127 100 101 113 118 112 96 121 97 98 ... gt; $Phenols : num2.82.652.83.852.83.272.52.62.82.98...gt$; Flavanoids : num 3.06 2.76 3.24 3.49 2.69 3.39 2.52 2.51 2.98 3.15 ... gt; $Nonflavanoids : num0.280.260.30.240.390.340.30.310.290.22...gt$; Proanthocyanins: num 2.29 1.28 2.81 2.18 1.82 1.97 1.98 1.25 1.98 1.85 ... gt; $Color : num5.644.385.687.84.326.755.255.055.27.22...gt$; Hue : num 1.04 1.05 1.03 0.86 1.04 1.05 1.02 1.06 1.08 1.01 ... gt; $Dilution : num3.923.43.173.452.932.853.583.582.853.55...gt$; Proline : int 1065 1050 1185 1480 735 1450 1290 1295 1045 1045 ... Wine data set contains 1 categorical variables (label) and 13 numerical variables. But these numerical variables is not scaled, I use scale function for scaling and centering data and then assign it as training data.

data.train lt;- scale(wine[-1]) Data is already centered and scaled.

summary(data.train) gt; Alcohol Malic gt; Min. :-2.42739 Min. :-1.4290 gt; 1st Qu.:-0.78603 1st Qu.:-0.6569 gt; Median : 0.06083 Median :-0.4219 gt; Mean : 0.00000 Mean : 0.0000 gt; 3rd Qu.: 0.83378 3rd Qu.: 0.6679 gt; Max. : 2.25341 Max. : 3.1004 gt; Ash Alcalinity gt; Min. :-3.66881 Min. :-2.663505 gt; 1st Qu.:-0.57051 1st Qu.:-0.687199 gt; Median :-0.02375 Median : 0.001514 gt; Mean : 0.00000 Mean : 0.000000 gt; 3rd Qu.: 0.69615 3rd Qu.: 0.600395 gt; Max. : 3.14745 Max. : 3.145637 gt; Magnesium Phenols gt; Min. :-2.0824 Min. :-2.10132 gt; 1st Qu.:-0.8221 1st Qu.:-0.88298 gt; Median :-0.1219 Median : 0.09569 gt; Mean : 0.0000 Mean : 0.00000 gt; 3rd Qu.: 0.5082 3rd Qu.: 0.80672 gt; Max. : 4.3591 Max. : 2.53237 gt; Flavanoids Nonflavanoids gt; Min. :-1.6912 Min. :-1.8630 gt; 1st Qu.:-0.8252 1st Qu.:-0.7381 gt; Median : 0.1059 Median :-0.1756 gt; Mean : 0.0000 Mean : 0.0000 gt; 3rd Qu.: 0.8467 3rd Qu.: 0.6078 gt; Max. : 3.0542 Max. : 2.3956 gt; Proanthocyanins Color gt; Min. :-2.06321 Min. :-1.6297 gt; 1st Qu.:-0.59560 1st Qu.:-0.7929 gt; Median :-0.06272 Median :-0.1588 gt; Mean : 0.00000 Mean : 0.0000 gt; 3rd Qu.: 0.62741 3rd Qu.: 0.4926 gt; Max. : 3.47527 Max. : 3.4258 gt; Hue Dilution gt; Min. :-2.08884 Min. :-1.8897 gt; 1st Qu.:-0.76540 1st Qu.:-0.9496 gt; Median : 0.03303 Median : 0.2371 gt; Mean : 0.00000 Mean : 0.0000 gt; 3rd Qu.: 0.71116 3rd Qu.: 0.7864 gt; Max. : 3.29241 Max. : 1.9554 gt; Proline gt; Min. :-1.4890 gt; 1st Qu.:-0.7824 gt; Median :-0.2331 gt; Mean : 0.0000 gt; 3rd Qu.: 0.7561 gt; Max. : 2.963 Model Fitting

Now the fun part begins. I use NbClust function to determine what is the best number of clusteres k for K-Means

nc lt;- NbClust(data.train, min.nc=2, max.nc=15, method=quot;kmeansquot;) barplot(table(nc$Best.n[1,]), xlab = quot; Numer of Clusters quot;, ylab = quot; Number of Criteria quot;, ma$ $quot; Number of Clusters Chosen by 26 Criteria quot;)$

According to the graph, we can find the best number of clusters is 3. Beside NbClust function which provides 30 indices for determing the number of clusters and proposes the best clustering scheme, we can draw the sum of square error (SSE) scree plot and look for a bend or elbow in this graph to determine appropriate k

wss lt;- 0 for (i in 1:15) wss[i] lt;- sum(kmeans(data.train, centers=i)$withinss)plot(1 :$ $15, wss, type = quot; b quot;, xlab = quot; Number of Clusters quot;, ylab = quot; Within groups sum of square$

Both two methods suggest k=3 is best choice for us. It's reasonsable if we take notice that the original data set also contains 3 classes.

Fit the model We now fit wine data to K-Means with k = 3

fit.km lt;- kmeans(data.train, 3) Then interpret the result

fit.km

gt; K-means clustering with 3 clusters of sizes 51, 65, 62 gt; gt; Cluster means: gt; Alcohol Malic Ash Alcalinity gt; 1 0.1644436 0.8690954 0.1863726 0.5228924 gt; 2 -0.9234669 -0.3929331 -0.4931257 0.1701220 gt; 3 0.8328826 -0.3029551 0.3636801 -0.6084749 gt; Magnesium Phenols Flavanoids Nonflavanoids gt; 1 -0.07526047 -0.97657548 -1.21182921 0.72402116 gt; 2 -0.49032869 -0.07576891 0.02075402 -0.03343924 gt; 3 0.57596208 0.88274724 0.97506900 -0.56050853 gt; Proanthocyanins Color Hue Dilution gt; 1 -0.77751312 0.9388902 -1.1615122 -1.2887761 gt; 2 0.05810161 -0.8993770 0.4605046 0.2700025 gt; 3 0.57865427 0.1705823 0.4726504 0.7770551 gt; Proline gt; 1 -0.4059428 gt; 2 -0.7517257 gt; 3 1.1220202 gt; gt; Clustering vector: gt; [1] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 gt; [26] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 gt; [51] 3 3 3 3 3 3 3 3 3 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 gt; [76] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 2 gt; [101] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 3 2 2 2 gt; [126] 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 gt; [151] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 gt; [176] 1 1 1 gt; gt; Within cluster sum of squares by cluster: gt; [1] 326.3537 558.6971 385.6983 gt; (between$_SS/total_SS = 44.8 gt; gt; Available components :$ $gt; gt; [1] quot; cluster quot; quot; centers quot; quot; totss quot; gt; [4] quot; withinss quot; quot; tot.withinss quot;$

First, I use plotcluster function from fpc package to draw discriminant projection plot

library(fpc) plotcluster(data.train, fit.km$cluster)$

We can see the data is clustered very well, there are no collapse between clusters. Next, we draw parallel coordinates plot to see how variables contributed in each cluster

library(MASS) parcoord(data.train, fit.km$cluster)$

We can extract some insights from above graph suc as black cluster contains wine with low flavanoids value, low proanthocyanins value, low hue value. Or green cluster contains wine which has dilution value higher than wine in red cluster.

Evaluation Because the original data set wine also has 3 classes, it is reasonable if we compare these classes with 3 clusters fited by K-Means

confuseTable.km lt;- table(wine$Type, fit.km$cluster) confuseTable.km gt; 1 2 3 gt; 1 0 0 59 gt; 2 3 65 3 gt; 3 48 0 We can see only 6 sample is missed. Let's use randIndex from flexclust to compare these two parititions - one from data set and one from result of clustering method.

library(flexclust) randIndex(ct.km) gt; ARI gt; 0.897495 It's quite close to 1 so K-Means is good model for clustering wine data set.

References Choosing number of cluster in K-Means, http://stackoverflow.com/a/15376462/1036500 K-means Clustering (from "R in Action"), http://www.r-statistics.com/2013/08/k-means-clustering-from-r-in-action/ Color the cluster output in r, http://stackoverflow.com/questions/1538696 the-cluster-output-in-r

26.4.4 Ensemble

Ensemble Algorithms 1 Ensemble methods are models composed of multiple weaker models that are independently trained and whose predictions are combined in some way to make the overall prediction.

Much effort is put into what types of weak learners to combine and the ways in which to combine them. This is a very powerful class of techniques and as such is very popular.

Boosting Bootstrapped Aggregation (Bagging) AdaBoost Stacked Generalization (blending) Gradient Boosting Machines (GBM) Gradient Boosted Regression Trees (GBRT) Random Forest XGBoost XGBoost is short for eXtreme gradient boosting.

Features 1 Easy to use Easy to install Highly developed R/python for users Efficiency Automatic parallel computation on a single machine Can be run on a cluster. Accuracy Good results for most data sets Feasibility Customized object and evaluation Turnable parameters Xgboost Optimization 2 You can use xgb.plot$_{i}mportanttodecidehowmanyfeaturesinyourmodel.Usexgb.cv(example)insteadofxgb.trainwithwat$ $//www.kaggle.com/c/otto-group-product-classification-challenge/forums/t/12947/achieve-$ $0-50776-on-the-leaderboard-in-a-minute-with-xgboost?page=5$

Installation Installation in Windows 64bit, Python 2.7, Anaconda

git clone https://github.com/dmlc/xgboost git checkout 9bc3d16 Open project in xgboost/windows with Visual Studio 2013 In Visual Studio 2013, open Configuration Manager..., choose Release in Active solution configuration choose x64 in Active solution platform Rebuild xgboost, xgboost$_{w}rapperCopyallfileinxgboost/windows/x64/Releasefold$ $package, runcommandpythonsetup.pyinstallCheckxgboostbyrunningcommandpython-$ $c"importxgboost"ExamplesMulticlassclassification:$

Understanding XGBoost Model on Otto Dataset

Resources http://www.slideshare.net/ShangxuanZhang/xgboost youtube, Kaggle Winning Solution Xgboost algorithm – Let us learn from its author

Notes on Parameter Tuning

26.4.5 Dimensionality Reduction

Dimensionality Reduction Algorithms Like clustering methods, dimensionality reduction seek and exploit the inherent structure in the data, but in this case in an unsupervised manner or order to summarise or describe data using less information.

This can be useful to visualize dimensional data or to simplify data which can then be used in a supervized learning method. Many of these methods can be adapted for use in classification and regression.

Principal Component Analysis (PCA) Principal Component Regression (PCR) Partial Least Squares Regression (PLSR) Sammon Mapping Multidimensional Scaling (MDS) Projection Pursuit Linear Discriminant Analysis (LDA) Mixture Discriminant Analysis (MDA) Quadratic Discriminant Analysis (QDA) Flexible Discriminant Analysis (FDA) t-SNE

t-Distributed Stochastic Neighbor Embedding (t-SNE) 1 is a (prize-winning) technique for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets. The technique can be implemented via Barnes-Hut approximations, allowing it to be applied on large real-world datasets. We applied it on data sets with up to 30 million examples. The technique and its variants are introduced in the following papers:

L.J.P. van der Maaten. Accelerating t-SNE using Tree-Based Algorithms. Journal of Machine Learning Research 15(Oct):3221-3245, 2014. PDF [Supplemental material] L.J.P. van der Maaten and G.E. Hinton. Visualizing Non-Metric Similarities in Multiple Maps. Machine Learning 87(1):33-55, 2012. PDF L.J.P. van der Maaten. Learning a Parametric Embedding by Preserving Local Structure. In Proceedings of the Twelfth International Conference on Artificial Intelligence Statistics (AI-STATS), JMLR WCP 5:384-391, 2009. PDF L.J.P. van der Maaten and G.E. Hinton. Visualizing High-Dimensional Data Using t-SNE. Journal of Machine Learning Research 9(Nov):2579-2605, 2008. PDF [Supplemental material] [Talk]

26.4.6 Anomaly Detection

Motivation and Examples Algorithms Evaluation AD: Examples Problem motivation 1 Anomaly detection is a reasonably commonly used type of machine learning application Can be thought of as a solution to an unsupervised learning problem But, has aspects of supervised learning What is anomaly detection? Imagine you're an aircraft engine manufacturer As engines roll off your assembly line you're doing QA Measure some features from engines (e.g. heat generated and vibration) You now have a dataset of x1 to xm (i.e. m engines were tested) Say we plot that dataset Next day you have a new engine An anomaly detection method is used to see if the new engine is anomalous (when compared to the previous engines) If the new engine looks like this; Probably OK - looks like the ones we've seen before But if the engine looks like this Uh oh! - this looks like an anomalous data-point More formally We have a dataset which contains normal (data) How we ensure they're normal is up to us In reality it's OK if there are a few which aren't actually normal Using that dataset as a reference point we can see if other examples are anomalous How do we do this? First, using our training dataset we build a model We can access this model using p(x) This asks, "What is the probability that example x is normal" Having built a model if $latexp(x_{test}) < \epsilon$ –> flag this as an anomaly if $latexp(x_{test}) \geq \epsilon$ –> this is OK is some threshold probability value which we define, depending on how sure we need/want to be We expect our model to (graphically) look something like this; i.e. this would be our model if we had 2D data Examples 1 Fraud detection Users have activity associated with them, such as Length on time on-line Location of login Spending frequency Using this data we can build a model of what normal users' activity is like What is the probability of "normal" behavior? Identify unusual users by sending their data through the model Flag up anything that looks a bit weird Automatically block cards/transactions Manufacturing Already spoke about aircraft engine example Monitoring computers in data center If you have many machines in a cluster Computer features of machine $latexx_1$ = memory use $latexx_2$ = number of disk accesses/sec $latexx_3$ = CPU load In addition to the measurable features you can also define your own complex features $latexx_4$ = CPU load/network traffic If you see an anomalous machine Maybe about to fail Look at replacing bits from it

26.5 Recomendation System

ntroduction 2 Two motivations for talking about recommender systems

Important application of ML systems Many technology companies find recommender systems to be absolutely key Think about websites (amazon, Ebay, iTunes genius) Try and recommend new content for you based on passed purchase Substantial part of Amazon's revenue generation Improvement in recommender system performance can bring in more income Kind of a funny problem In academic learning, recommender systems receives a small amount of attention But in industry it's an absolutely crucial tool Talk about the big ideas in machine learning Not so much a technique, but an idea As soon, features are really important There's a big idea in machine learning that for some problems you can learn what a good set of features are So not select those features but learn them Recommender systems do this - try and identify the crucial and relevant features Example - predict movie ratings You're a company who sells movies You let users rate movies using a 1-5 star rating To make the example nicer, allow 0-5 (makes math easier) You have five movies And you have four users Admittedly, business isn't going well, but you're optimistic about the future as a result of your truly outstanding (if limited) inventory

To introduce some notation

$latexn_u$ - Number of users (called $?^{nu}$ occasionally as we can't subscript in superscript) $latexn_m$ - Number of movies $latexr(i,j)$ - 1 if user j has rated movie i (i.e. bitmap) $latexy(i,j)$ - rating given by user j to move i (defined only if $latexr(i,j) = 1$) So for this example $latexn_u = 4$ $latexn_m = 5$ Summary of scoring Alice and Bob gave good ratings to rom coms, but low scores to action films Carol and Dave game good ratings for action films but low ratings for rom coms We have the data given above The problem is as follows Given $latexr(i,j)$ and $latexy^{(i,j)}$ - go through and try and predict missing values (?s) Come up with a learning algorithm that can fill in these missing values KDD 2015 Tutorial: Shlomo Berkovsky and Jill Freyne, Web Personalisation and Recommender Systems

1. Approaches 1

Attribute-based Recommendations

You like action movies, starring Clint Eastwood, you might like "Good, Bad and the Ugly" (Netflix)

Item Hierachy

You bought Printer you will also need ink (Bestbuy)

Association Rules

Content-Based Recommender Collaborative Filtering - Item-Item Similarity

You like Godfather so you will like Scarface (Netflix)

Collaborative Filtering - User-User Similarity

People like you who bought beer also bought diapers (Target)

Social+Interest Graph Based

Your friends like Lady Gaga so you will like Lady Gaga (Facebook, Linkedin)

Model Based

Training SVM, LDA, SVD for implicit features.

2. Challenges Kaggle Challenge: Million Song Dataset Challenge

3. Articles How Big Data is used in Recommendation Systems to change our lives 4. Recommendation Interface 4.1 Type of Input predictions recommendations filtering organic vs explicit presentation 4.2 Type of Output explicit implicit Apriori https://en.wikipedia.org/wiki/Apriori$_algorithm$

https://github.com/asaini/Apriori

Item item collaborative filtering Works when |U| » |I|

items dont change much RS: Examples Google News

http://1.bp.blogspot.com/$_7ZYqYi4xigk/TCuWLmXhdjI/AAAAAAAAGVI/umfi5tHpBr0/s1600/Go$ $News + Redesign + June + 30 + 2010 + AM + PT.jpg$

RS: Association Rules

Content Based Recommendation User–User Collaborative Filtering User - User 1 User user look simular in row space

$p_{u,i} = \overline{r_u} + \frac{\sum_{u' \in N} s(u,u')(r_{u',i} - \overline{r'_u})}{\sum_{u' \in N} |s(u,u')|} s = 2$

http://files.grouplens.org/papers/FnT

mlclass lecture notes, Recommender Systems

Chương 27

Probabilistic Graphical Model

View online http://magizbox.com/training/probabilistic$_{g}raphical_{m}odels/site/$

Probabilistic graphical models (PGMs) are a rich framework for encoding probability distributions over complex domains: joint (multivariate) distributions over large numbers of random variables that interact with each other. These representations sit at the intersection of statistics and computer science, relying on concepts from probability theory, graph algorithms, machine learning, and more. They are the basis for the state-of-the-art methods in a wide variety of applications, such as medical diagnosis, image understanding, speech recognition, natural language processing, and many, many more. They are also a foundational tool in formulating many machine learning problems.

27.1 Representation

Probabilistic graphical models (PGMs) are a rich framework for encoding probability distributions over complex domains: joint (multivariate) distributions over large numbers of random variables that interact with each other.

These representations sit at the intersection of statistics and computer science, relying on concepts from probability theory, graph algorithms, machine learning, and more. They are the basis for the state-of-the-art methods in a wide variety of applications, such as medical diagnosis, image understanding, speech recognition, natural language processing, and many, many more. They are also a foundational tool in formulating many machine learning problems.

27.2 Foundation: Probability Theory

he main focus of this book is on complex probability distributions. In this section we briefly review basic concepts from probability theory.

1 Probability Distributions When we use the word "probability" in day-to-day life, we refer to a degree of confidence that an event of an uncertain nature will occur. For example, the weather report might say "there is a low probability of light rain in the afternoon." Probability theory deals with the

formal foundations for discussing such estimates and the rules they should obey. Before we discuss the representation of probability, we need to define what the events are to which we want to assign a probability. These events might be dierent outcomes of throwing a die, the outcome of a horse race, the weather configurations in California, or the possible failures of a piece of machinery.

1.1 Event Spaces event Formally, we define events by assuming that there is an agreed upon space of possible outcomes, outcome space which we denote by . For example, if we consider dice, we might set = 1, 2, 3, 4, 5, 6. In the case of a horse race, the space might be all possible orders of arrivals at the finish line, a much larger space.

measurable event In addition, we assume that there is a set of measurable events S to which we are willing to assign probabilities. Formally, each event S is a subset of . In our die example, the event 6 represents the case where the die shows 6, and the event 1, 3, 5 represents the case of an odd outcome. In the horse-race example, we might consider the event "Lucky Strike wins," which contains all the outcomes in which the horse Lucky Strike is first. Probability theory requires that the event space satisfy three basic properties: • It contains the empty event , and the trivial event . • It is closed under union. That is, if , S, then so is . • It is closed under complementation. That is, if S, then so is . The requirement that the event space is closed under union and complementation implies that it is also closed under other Boolean operations, such as intersection and set dierence.

1.2 Probability Distributions Definition 2.1 A probability distribution P over (, S) is a mapping from events in S to real values that satisfies probability distribution the following conditions: • P() 0 for all S. • P() = 1. • If , S and = , then P( ) = P() + P(). The first condition states that probabilities are not negative. The second states that the "trivial event," which allows all possible outcomes, has the maximal possible probability of 1. The third condition states that the probability that one of two mutually disjoint events will occur is the sum of the probabilities of each event. These two conditions imply many other conditions. Of particular interest are P() = 0, and P( ) = P() + P() P( ).

1.3 Interpretations of Probability Before we continue to discuss probability distributions, we need to consider the interpretations that we might assign to them. Intuitively, the probability P() of an event quantifies the degree of confidence that will occur. If P() = 1, we are certain that one of the outcomes in occurs, and if P() = 0, we consider all of them impossible. Other probability values represent options that lie between these two extremes. This description, however, does not provide an answer to what the numbers mean. There are two common interpretations for probabilities. frequentist The frequentist interpretation views probabilities as frequencies of events. More precisely, the interpretation probability of an event is the fraction of times the event occurs if we repeat the experiment indefinitely. For example, suppose we consider the outcome of a particular die roll. In this case, the statement P() = 0.3, for = 1, 3, 5, states that if we repeatedly roll this die and record the outcome, then the fraction of times the outcomes in will occur is 0.3. More precisely, the limit of the sequence of fractions of outcomes in in the first roll, the first two rolls, the first three rolls, . . ., the first n rolls, . . . is 0.3.

The frequentist interpretation gives probabilities a tangible semantics. When we discuss concrete physical systems (for example, dice, coin flips, and card games) we can envision how these frequencies are defined. It is also relatively straightforward to check that frequencies must satisfy the requirements of proper distributions. The frequentist interpretation fails, however, when we consider events such as "It will rain tomorrow afternoon." Although the time span of "Tomorrow afternoon" is somewhat ill defined, we expect it to occur exactly once. It is not clear how we define the frequencies of such events. Several attempts have been made to define the probability for such an event by finding a reference class reference class of similar events for which frequencies are well defined; however, none of them has proved entirely satisfactory. Thus, the frequentist approach does not provide a satisfactory interpretation for a statement such as "the probability of rain tomorrow afternoon is 0.3." An alternative interpretation views probabilities as subjective degrees of belief. Under subjective interpretation this interpretation, the statement P() = 0.3 represents a subjective statement about one's own degree of belief that the event will come about. Thus, the statement "the probability of rain tomorrow afternoon is 50 percent" tells us that in the opinion of the speaker, the chances of rain and no rain tomorrow afternoon are the same. Although tomorrow afternoon will occur only once, we can still have uncertainty about its outcome, and represent it using numbers (that is, probabilities). This description still does not resolve what exactly it means to hold a particular degree of belief. What stops a person from stating that the probability that Bush will win the election is 0.6 and the probability that he will lose is 0.8? The source of the problem is that we need to explain how subjective degrees of beliefs (something that is internal to each one of us) are reflected in our actions. This issue is a major concern in subjective probabilities. One possible way of attributing degrees of beliefs is by a betting game. Suppose you believe that P() = 0.8. Then you would be willing to place a bet of $1 against 3$. To see this, note that with probability 0.8 you gain a dollar, and with probability 0.2 you lose $3, so on average this bet is a good deal with expected gain of 20 cents. In fact, you might be even tempted to place a bet of$ against $4. Under this bet the average gain is 0, so you should not mind. However, you would not consider it worthwhi$ against $4 and 10 cents, since that would have negative expected gain. Thus, by finding which bets you are willing to p$

2 Basic Concepts in Probability 2.1 Conditional Probability To use a concrete example, suppose we consider a distribution over a population of students taking a certain course. The space of outcomes is simply the set of all students in the population. Now, suppose that we want to reason about the students' intelligence and their final grade. We can define the event to denote "all students with grade A," and the event to denote "all students with high intelligence." Using our distribution, we can consider the probability of these events, as well as the probability of (the set of intelligent students who got grade A). This, however, does not directly tell us how to update our beliefs given new evidence. Suppose we learn that a student has received the grade A; what does that tell us about her intelligence? This kind of question arises every time we want to use distributions to reason about the real world. More precisely, after learning that an event is true, how do we change our probability conditional about occurring? The answer is via the notion of conditional probability. Formally, the probability conditional probability of given is defined as P(|) = P( ) P() (2.1) That is, the probability that is true given that we know is the relative proportion of outcomes satisfying among these that satisfy . (Note that the conditional probability is not defined when P() = 0.) The conditional probability given an event (say) satisfies the properties of definition 2.1 (see exercise 2.4), and thus it is a probability distribution by its own right. Hence, we can think of the conditioning operation as taking one distribution and returning another

over the same probability space.

2.2 Chain Rule and Bayes Rule From the definition of the conditional distribution, we immediately see that P( ) = P()P(|). (2.2) chain rule This equality is known as the chain rule of conditional probabilities. More generally, if 1, . . . , k are events, then we can write P(1 . . . k) = P(1)P(2 | 1) · · · P(k | 1 . . . k1). (2.3) In other words, we can express the probability of a combination of several events in terms of the probability of the first, the probability of the second given the first, and so on. It is important to notice that we can expand this expression using any order of events — the result will remain the same. Bayes' rule Another immediate consequence of the definition of conditional probability is Bayes' rule P(|) = P(|)P() P()

A more general conditional version of Bayes' rule, where all our probabilities are conditioned on some background event , also holds: P(|) = P(|)P(|) P(|) . Bayes' rule is important in that it allows us to compute the conditional probability P(|) from the "inverse" conditional probability P(|). Example 2.1 Consider the student population, and let Smart denote smart students and GradeA denote students who got grade A. Assume we believe (perhaps based on estimates from past statistics) that P(GradeA | Smart) = 0.6, and now we learn that a particular student received grade A. Can we estimate the probability that the student is smart? According to Bayes' rule, this depends on prior our prior probability for students being smart (before we learn anything about them) and the prior probability of students receiving high grades. For example, suppose that P(Smart) = 0.3 and P(GradeA) = 0.2, then we have that P(Smart | GradeA) = 0.6 0.3/0.2 = 0.9. That is, an A grade strongly suggests that the student is smart. On the other hand, if the test was easier and high grades were more common, say, P(GradeA) = 0.4 then we would get that P(Smart | GradeA) = 0.6 0.3/0.4 = 0.45, which is much less conclusive about the student. Another classic example that shows the importance of this reasoning is in disease screening. To see this, consider the following hypothetical example (none of the mentioned figures are related to real statistics). Example 2.2 Suppose that a tuberculosis (TB) skin test is 95 percent accurate. That is, if the patient is TB-infected, then the test will be positive with probability 0.95, and if the patient is not infected, then the test will be negative with probability 0.95. Now suppose that a person gets a positive test result. What is the probability that he is infected? Naive reasoning suggests that if the test result is wrong 5 percent of the time, then the probability that the subject is infected is 0.95. That is, 95 percent of subjects with positive results have TB. If we consider the problem by applying Bayes' rule, we see that we need to consider the prior probability of TB infection, and the probability of getting positive test result. Suppose that 1 in 1000 of the subjects who get tested is infected. That is, P(TB) = 0.001. What is the probability of getting a positive test result? From our description, we see that 0.001 · 0.95 infected subjects get a positive result, and 0.999·0.05 uninfected subjects get a positive result. Thus, P(Positive) = 0.0509. Applying Bayes' rule, we get that P(TB | Positive) = 0.001·0.95/0.0509 0.0187. Thus, although a subject with a positive test is much more probable to be TB-infected than is a random subject, fewer than 2 percent of these subjects are TB-infected.

3 Random Variables and Joint Distributions 3.1 Motivation Our discussion of probability distributions deals with events. Formally, we can consider any event from the set of measurable events. The description of events is in terms of sets of outcomes. In many cases, however, it would be more natural to consider

attributes of the outcome. For example, if we consider a patient, we might consider attributes such as "age,"

"gender," and "smoking history" that are relevant for assigning probability over possible diseases and symptoms. We would like then consider events such as "age > 55, heavy smoking history, and suers from repeated cough." To use a concrete example, consider again a distribution over a population of students in a course. Suppose that we want to reason about the intelligence of students, their final grades, and so forth. We can use an event such as GradeA to denote the subset of students that received the grade A and use it in our formulation. However, this discussion becomes rather cumbersome if we also want to consider students with grade B, students with grade C, and so on. Instead, we would like to consider a way of directly referring to a student's grade in a clean, mathematical way. The formal machinery for discussing attributes and their values in dierent outcomes are random variable random variables. A random variable is a way of reporting an attribute of the outcome. For example, suppose we have a random variable Grade that reports the final grade of a student, then the statement P (Grade = A) is another notation for P (GradeA).

n the statement P (Grade = A) is another notation for P (GradeA).

3.2 What Is a Random Variable? Formally, a random variable, such as Grade, is defined by a function that associates with each outcome in a value. For example, Grade is defined by a function fGrade that maps each person in to his or her grade (say, one of A, B, or C). The event Grade = A is a shorthand for the event   : fGrade() = A. In our example, we might also have a random variable Intelligence that (for simplicity) takes as values either "high" or "low." In this case, the event "Intelligence = high" refers, as can be expected, to the set of smart (high intelligence) students. Random variables can take dierent sets of values. We can think of categorical (or discrete) random variables that take one of a few values, as in our two preceding examples. We can also talk about random variables that can take infinitely many values (for example, integer or real values), such as Height that denotes a student's height. We use Val(X) to denote the set of values that a random variable X can take. In most of the discussion in this book we examine either categorical random variables or random variables that take real values. We will usually use uppercase roman letters X, Y, Z to denote random variables. In discussing generic random variables, we often use a lowercase letter to refer to a value of a random variable. Thus, we use x to refer to a generic value of X. For example, in statements such as "P (X = x) 0 for all x Val(X)." When we discuss categorical random variables, we use the notation x1, . . . , xk, for k = |Val(X)| (the number of elements in Val(X)), when we need to enumerate the specific values of X, for example, in statements such as kX i =1 P (X = xi) = 1. multinomial The distribution over such a variable is called a multinomial. In the case of a binary-valued distribution random variable X, where Val(X) = false, true, we often use x1 to denote the value true for X, and x0 to denote the value false. The distribution of such a random variable is called a Bernoulli Bernoulli distribution. distribution We also use boldface type to denote sets of random variables. Thus, X, Y , or Z are typically used to denote a set of random variables, while x, y, z denote assignments of values to the

variables in these sets. We extend the definition of Val(X) to refer to sets of variables in the obvious way. Thus, x is always a member of Val(X). For Y X, we use xhY i to refer to the assignment within x to the variables in Y . For two assignments x (to X) and y (to Y), we say that x y if they agree on the variables in their intersection, that is, xhX Y i = yhX Y i. In many cases, the notation P(X = x) is redundant, since the fact that x is a value of X is already reported by our choice of letter. Thus, in many texts on probability, the identity of a random variable is not explicitly mentioned, but can be inferred through the notation used for its value. Thus, we use P(x) as a shorthand for P(X = x) when the identity of the random variable is clear from the context. Another shorthand notation is that Px refers to a sum over all possible values that X can take. Thus, the preceding statement will often appear as Px P(x) = 1. Finally, another standard notation has to do with conjunction. Rather than write P((X = x) (Y = y)), we write P(X = x, Y = y), or just P(x, y).

3.3 Marginal and Joint Distributions Once we define a random variable X, we can consider the distribution over events that can be marginal described using X. This distribution is often referred to as the marginal distribution over the distribution random variable X. We denote this distribution by P(X). Returning to our population example, consider the random variable Intelligence. The marginal distribution over Intelligence assigns probability to specific events such as P(Intelligence = high) and P(Intelligence = low), as well as to the trivial event P(Intelligence high, low). Note that these probabilities are defined by the probability distribution over the original space. For concreteness, suppose that P(Intelligence = high) = 0.3, P(Intelligence = low) = 0.7. If we consider the random variable Grade, we can also define a marginal distribution. This is a distribution over all events that can be described in terms of the Grade variable. In our example, we have that P(Grade = A) = 0.25, P(Grade = B) = 0.37, and P(Grade = C) = 0.38. It should be fairly obvious that the marginal distribution is a probability distribution satisfying the properties of definition 2.1. In fact, the only change is that we restrict our attention to the subsets of S that can be described with the random variable X. In many situations, we are interested in questions that involve the values of several random variables. For example, we might be interested in the event "Intelligence = high and Grade = A." joint distribution To discuss such events, we need to consider the joint distribution over these two random variables. In general, the joint distribution over a set X = X1, . . . , Xn of random variables is denoted by P(X1, . . . , Xn) and is the distribution that assigns probabilities to events that are specified in terms of these random variables. We use to refer to a full assignment to the variables in X , that is, Val(X). The joint distribution of two random variables has to be consistent with the marginal distribution, in that P(x) = P y P(x, y). This relationship is shown in figure 2.1, where we compute the marginal distribution over Grade by summing the probabilities along each row. Similarly, we find the marginal distribution over Intelligence by summing out along each column. The resulting sums are typically written in the row or column margins, whence the term "marginal distribution." Suppose we have a joint distribution over the variables X = X1, . . . , Xn. The most fine-grained events we can discuss using these variables are ones of the form "X1 = x1 and X2 = x2, . . ., and Xn = xn" for a choice of values x1, . . . , xn for all the variables. Moreover,

Intelligence low high A 0.07 0.18 0.25 Grade B 0.28 0.09 0.37 C 0.35 0.03 0.38 0.7 0.3 1 Figure 2.1 Example of a joint distribution P(Intelligence, Grade): Values of Intelligence (columns) and Grade (rows) with the associated marginal distribution on each variable. any two such events must be either identical or disjoint, since they both assign values to all the variables in X . In addition, any event defined using variables in X must be a union of a set of canonical such events. Thus, we are eectively working in a canonical outcome space: a space where each outcome space outcome corresponds to a joint assignment to X1, . . . , Xn. More precisely, all our probability computations remain the same whether we consider the original outcome space (for example, all students), or the canonical space (for example, all combinations of intelligence and grade). atomic outcome We use to denote these atomic outcomes: those assigning a value to each variable in X . For example, if we let X = Intelligence, Grade, there are six atomic outcomes, shown in figure 2.1. The figure also shows one possible joint distribution over these six outcomes. Based on this discussion, from now on we will not explicitly specify the set of outcomes and measurable events, and instead implicitly assume the canonical outcome space.

3.4 Conditional Probability The notion of conditional probability extends to induced distributions over random variables. For conditional example, we use the notation P (Intelligence | Grade = A) to denote the conditional distribution distribution over the events describable by Intelligence given the knowledge that the student's grade is A. Note that the conditional distribution over a random variable given an observation of the value of another one is not the same as the marginal distribution. In our example, P (Intelligence = high) = 0.3, and P (Intelligence = high | Grade = A) = 0.18/0.25 = 0.72. Thus, clearly P (Intelligence | Grade = A) is dierent from the marginal distribution P (Intelligence). The latter distribution represents our prior knowledge about students before learning anything else about a particular student, while the conditional distribution represents our more informed distribution after learning her grade. We will often use the notation P (X | Y) to represent a set of conditional probability distributions. Intuitively, for each value of Y , this object assigns a probability over values of X using the conditional probability. This notation allows us to write the shorthand version of the chain rule: P (X, Y) = P (X)P (Y | X), which can be extended to multiple variables as P (X1, . . . , Xk) = P (X1)P (X2 | X1) · · · P (Xk | X1, . . . , Xk1). (2.5) Similarly, we can state Bayes' rule in terms of conditional probability distributions: P (X | Y) = P (X)P (Y | X) P (Y) . (2.6)

4 Independence and Conditional Independence 4.1 Independence As we mentioned, we usually expect P(|)P(|) to be different from P()P(). That is, learning that is true changes our probability over . However, in some situations equality can occur, so that P(|)=P()P(|)=P(). That is, learning that occurs did not change our probability of .

Definition independent events

We say that an event is independent of event in PP, denoted P()P(), if P(|)=P()P(|)=P() or if P()=0P()=0.

We can also provide an alternative definition for the concept of independence:

Proposition 2.1

A distribution PP satisfies ()() if and only if P()=P()P()P()=P()P().

PROOF Consider first the case where P()=0P()=0; here, we also have P()=0P()=0, and so the equivalence immediately holds. When P()0P()0, we can use the chain rule; we write P()=P(|)P()P()=P(|)P(). Since is independent of , we have that P(|)=P()P(|)=P(). Thus, P()=P()P()P()=P()P(). Conversely, suppose that P()=P()P()P()=P()P(). Then, by definition, we have that

P(|)=P()P()=P()P()P()=P(). P(|)=P()P()=P()P()P()=P(). As an immediate consequence of this alternative definition, we see that independence is a symmetric notion. That is, ( ) implies ( ). Example 2.3 For example, suppose that we toss two coins, and let be the event "the first toss results in a head" and the event "the second toss results in a head." It is not hard to convince ourselves that we expect that these two events to be independent. Learning that is true would not change our probability of . In this case, we see two dierent physical processes (that is, coin tosses) leading to the events, which makes it intuitive that the probabilities of the two are independent. In certain cases, the same process can lead to independent events. For example, consider the event denoting "the die outcome is even" and the event denoting "the die outcome is 1 or 2." It is easy to check that if the die is fair (each of the six possible outcomes has probability 1 6), then these two events are independent.

4.2 Conditional Independence While independence is a useful property, it is not often that we encounter two indepen- dent events. A more common situation is when two events are independent given an additional event. For example, suppose we want to reason about the chance that our student is accepted to graduate studies at Stanford or MIT. Denote by Stanford the event "admitted to Stanford" and by MIT the event "admitted to MIT." In most reasonable distributions, these two events are not independent. If we learn that a student was admitted to Stanford, then our estimate of her probability of being accepted at MIT is now higher, since it is a sign that she is a promising student

Now, suppose that both universities base their decisions only on the student's grade point average (GPA), and we know that our student has a GPA of A. In this case, we might argue that learning that the student was admitted to Stanford should not change the probability that she will be admitted to MIT: Her GPA already tells us the information relevant to her chances of admission to MIT, and finding out about her admission to Stanford does not change that. Formally, the statement is P(MIT | Stanford, GradeA) = P(MIT | GradeA). In this case, we say that MIT is conditionally independent of Stanford given GradeA. Definition 2.3 We say that an event is conditionally independent of event given event in P, denoted conditional independence P |= (|), if P(|) = P(|) or if P( ) = 0. It is easy to extend the arguments we have seen in the case of (unconditional) independencies to give an alternative definition. Proposition 2.2 P satisfies (|) if and only if P(|) = P(|)P(|).

4.3 Independence of Random Variables Until now, we have focused on independence between events. Thus, we can say that two events, such as one toss landing heads and a second also landing heads, are independent. However, we would like to say that any pair of outcomes of the coin tosses is independent. To capture such statements, we can examine the generalization of independence to sets of random variables. Definition 2.4 Let X, Y , Z be sets of random variables. We say that X is conditionally independent of Y given conditional independence Z in a distribution P if P satisfies (X = x Y = y | Z = z) for all values x Val(X), y Val(Y), and z Val(Z). The variables in the set Z are often said to be observed. If the set observed variable Z is empty, then instead of writing (X Y |), we write (X Y) and say that X and Y are marginally independent. marginal independence Thus, an independence statement over random variables is a universal quantification over all possible values of the random variables. The alternative characterization of conditional independence follows immediately: Proposition 2.3 The distribution P satisfies (X Y | Z) if and only if P(X, Y | Z) = P(X | Z)P(Y | Z). Suppose we learn about a conditional independence. Can we conclude other independence properties that must hold in the distribution? We have already seen one such example: symmetry • Symmetry: (X Y | Z) = (Y X | Z).

(2.7) There are several other properties that hold for conditional independence, and that often provide a very clean method for proving important properties about distributions. Some key properties are:

• Decomposition: (X Y , W | Z) = (X Y | Z). (2.8) weak union • Weak union: (X Y , W | Z) = (X Y | Z, W). (2.9) contraction • Contraction: (X W | Z, Y)(X Y | Z) = (X Y , W | Z). (2.10) An additional important property does not hold in general, but it does hold in an important subclass of distributions. Definition 2.5 A distribution P is said to be positive if for all events S such that 6= , we have that positive distribution P() > 0. For positive distributions, we also have the following property: intersection • Intersection: For positive distributions, and for mutually disjoint sets X, Y , Z, W : (X Y | Z, W)(X W | Z, Y) = (X Y , W | Z). (2.11) The proof of these properties is not dicult. For example, to prove Decomposition, assume that (X Y, W | Z) holds. Then, from the definition of conditional independence, we have that P(X, Y, W | Z) = P(X | Z)P(Y, W | Z). Now, using basic rules of probability and arithmetic, we can show P(X, Y | Z) = X w P(X, Y, w | Z) = X w P(X | Z)P(Y, w | Z) = P(X | Z) X w P(Y, w | Z) = P(X | Z)P(Y | Z). The only property we used here is called "reasoning by cases" (see exercise 2.6). We conclude that (X Y | Z).

5 Querying a Distribution Our focus throughout this book is on using a joint probability distribution over multiple random variables to answer queries of interest.

5.1 Probability Queries probability query Perhaps the most common query type is the probability query. Such a query consists of two parts: evidence • The evidence: a subset E of random variables in the model, and an instantiation e to these variables; query variables • the query variables: a subset Y of random variables in the network. Our task is to compute P(Y | E = e), posterior that is, the posterior probability distribution over the values y of Y , conditioned on the fact that distribution E = e. This expression can also be viewed as the marginal over Y , in the distribution we obtain by conditioning on e.

5.2 MAP Queries A second important type of task is that of finding a high-probability joint assignment to some subset of variables. The simplest variant of this type of task is the MAP query (also called MAP assignment most probable explanation (MPE)), whose aim is to find the MAP assignment — the most likely assignment to all of the (non-evidence) variables. More precisely, if we let W = X E, our task is to find the most likely assignment to the variables in W given the evidence E = e: MAP(W | e) = argmax w P(w, e), (2.12) where, in general, argmaxx f(x) represents the value of x for which f(x) is maximal. Note that there might be more than one assignment that has the highest posterior probability. In this case, we can either decide that the MAP task is to return the set of possible assignments, or to return an arbitrary member of that set. It is important to understand the dierence between MAP queries and probability queries. In a MAP query, we are finding the most likely joint assignment to W . To find the most likely assignment to a single variable A, we could simply compute P(A | e) and then pick the most likely value. However, the assignment where each variable individually picks its most likely value can be quite dierent from the most likely joint assignment to all variables simultaneously. This phenomenon can occur even in the simplest case, where we have no evidence. Example 2.4 Consider a two node chain A B where A and B are both binary-valued. Assume that: a0 a1 0.4 0.6 A b0 b1 a0 0.1 0.9 a1 0.5 0.5 (2.13) We can see that P(a1) > P(a0), so that MAP(A) = a1. However, MAP(A, B) = (a0, b1): Both values of B have the same probability given a1. Thus, the most likely assignment containing a1 has probability 0.6 0.5 = 0.3. On the other hand, the distribution over values of B is more skewed given a0, and the most likely assignment (a0, b1) has the probability 0.4 0.9 = 0.36. Thus, we have that argmaxa,b P(a, b) 6= (argmaxa P(a),argmaxb P(b)).

5.3 Marginal MAP Queries To motivate our second query type, let us return to the phenomenon demonstrated in example 2.4. Now, consider a medical diagnosis problem, where the most likely disease has multiple possible symptoms, each of which occurs with some probability, but not an overwhelming probability. On the other hand, a somewhat rarer disease might have only a few symptoms, each of which is very likely given the disease. As in our simple example, the MAP assignment to the data and the symptoms might be higher for the second disease than for the first one. The solution here is to look for the most likely assignment to the disease variable(s) only, rather than the most likely assignment to both the disease and symptom variables. This approach suggests marginal MAP the use of a more general query type. In the marginal MAP query, we have a subset of variables Y that forms our query. The task is to find the most likely assignment to the variables in Y given the evidence E = e: MAP(Y | e) = arg max y P(y | e). If we let Z = X Y E, the marginal MAP task is to compute: MAP(Y | e) = arg max Y X Z P(Y , Z | e). Thus, marginal MAP queries contain both summations and maximizations; in a way, it contains elements of both a conditional probability query and a MAP query. Note that example 2.4 shows that marginal MAP assignments are not monotonic: the most likely assignment MAP(Y1 | e) might be completely dierent from the assignment to Y1 in MAP(Y1, Y2 | e). Thus, in particular, we cannot use a MAP query to give us the correct answer to a marginal MAP query.

6 Continuous Spaces In the previous section, we focused on random variables that have a finite set of possible values. In many situations, we also want to reason about continuous quantities such as weight, height, duration, or cost that take real numbers in IR. When dealing with probabilities over continuous random variables, we have to deal with some technical issues. For example, suppose that we want to reason about a random variable X that can take values in the range between 0 and 1. That is, Val(X) is the interval [0, 1]. Moreover, assume that we want to assign each number in this range equal probability. What would be the probability of a number x? Clearly, since each x has the same probability, and there are infinite number of values, we must have that P(X = x) = 0. This problem appears even if we do not require uniform probability.

6.1 Probability Density Functions How do we define probability over a continuous random variable? We say that a function density function p : IR 7 IR is a probability density function or (PDF) for X if it is a nonnegative integrable

function such that Z Val(X) p(x)dx = 1. That is, the integral over the set of possible values of X is 1. The PDF defines a distribution for X as follows: for any x in our event space: P(X a) = aZ p(x)dx. cumulative The function P is the cumulative distribution for X. We can easily employ the rules of distribution probability to see that by using the density function we can evaluate the probability of other events. For example, P(a X b) = bZa p(x)dx. Intuitively, the value of a PDF p(x) at a point x is the incremental amount that x adds to the cumulative distribution in the integration process. The higher the value of p at and around x, the more mass is added to the cumulative distribution as it passes x. The simplest PDF is the uniform distribution. Definition 2.6 A variable X has a uniform distribution over [a, b], denoted X Unif[a,b] if it has the PDF uniform distribution p(x) = 0 otherwise b 1a b x a. Thus, the probability of any subinterval of [a, b] is proportional its size relative to the size of [a, b]. Note that, if b a < 1, then the density can be greater than 1. Although this looks unintuitive, this situation can occur even in a legal PDF, if the interval over which the value is greater than 1 is not too large. We have only to satisfy the constraint that the total area under the PDF is 1. As a more complex example, consider the Gaussian distribution. Definition 2.7 A random variable X has a Gaussian distribution with mean and variance 2, denoted X Gaussian distribution N ; 2, if it has the PDF p(x) = 21 e (x2 2)2 . standard A standard Gaussian is one with mean 0 and variance 1. Gaussian A Gaussian distribution has a bell-like curve, where the mean parameter controls the location of the peak, that is, the value for which the Gaussian gets its maximum value. The variance parameter 2 determines how peaked the Gaussian is: the smaller the variance, the

more peaked the Gaussian. Figure 2.2 shows the probability density function of a few dierent Gaussian distributions. More technically, the probability density function is specified as an exponential, where the expression in the exponent corresponds to the square of the number of standard deviations that x is away from the mean . The probability of x decreases exponentially with the square of its deviation from the mean, as measured in units of its standard deviation.

6.2 Joint Density Functions The discussion of density functions for a single variable naturally extends for joint distributions of continuous random variables. Definition 2.8 Let P be a joint distribution over continuous random variables X1, . . . , Xn. A function p(x1, . . . , xn) joint density is a joint density function of X1, . . . , Xn if • p(x1, . . . , xn) 0 for all values x1, . . . , xn of X1, . . . , Xn. • p is an integrable function. • For any choice of a1, . . . , an, and b1, . . . , bn, P(a1 X1 b1, . . . , an Xn bn) = b1 Za1 · · · b nZ a n p(x1, . . . , xn)dx1 . . . dxn. Thus, a joint density specifies the probability of any joint event over the variables of interest. Both the uniform distribution and the Gaussian distribution have natural extensions to the multivariate case. The definition of a multivariate uniform distribution is straightforward. We defer the definition of the multivariate Gaussian to section 7.1. From the joint density we can derive the marginal density of any random variable by integrating out the other variables. Thus, for example, if p(x, y) is the joint density of X and Y

then p(x) = Z p(x, y)dy. To see why this equality holds, note that the event a X b is, by definition, equal to the event "a X b and Y ." This rule is the direct analogue of marginalization for discrete variables. Note that, as with discrete probability distributions, we abuse notation a bit and use p to denote both the joint density of X and Y and the marginal density of X. In cases where the distinction is not clear, we use subscripts, so that pX will be the marginal density, of X, and pX,Y the joint density.

6.3 Conditional Density Functions As with discrete random variables, we want to be able to describe conditional distributions of continuous variables. Suppose, for example, we want to define P(Y | X = x). Applying the definition of conditional distribution (equation (2.1)), we run into a problem, since P(X = x) = 0. Thus, the ratio of P(Y, X = x) and P(X = x) is undefined. To avoid this problem, we might consider conditioning on the event x X x + , which can have a positive probability. Now, the conditional probability is well defined. Thus, we might consider the limit of this quantity when 0. We define P(Y | x)

= lim 0 P(Y | x X x +). When does this limit exist? If there is a continuous joint density function p(x, y), then we can derive the form for this term. To do so, consider some event on Y , say a Y b. Recall that P(a Y b | x X x +) = P(a Y b, x X x +) P(x X x +) = Ra b Rx x + p(x0, y)dydx0 Rx x + p(x0)dx0 . When is suciently small, we can approximate x+ Z x p(x0)dx0 2p(x). Using a similar approximation for p(x0, y), we get P(a Y b | x X x +) Ra b 2p(x, y)dy 2p(x) = bZa p(x, y) p(x) dy. We conclude that p(x,y) p(x) is the density of P(Y | X = x).

Let p(x, y) be the joint density of X and Y . The conditional density function of Y given X is conditional density function defined as p(y | x) = p(x, y) p(x) When p(x) = 0, the conditional density is undefined. The conditional density p(y | x) characterizes the conditional distribution P(Y | X = x) we defined earlier. The properties of joint distributions and conditional distributions carry over to joint and conditional density functions. In particular, we have the chain rule p(x, y) = p(x)p(y | x) (2.14) and Bayes' rule p(x | y) = p(x)p(y | x) p(y) . (2.15) As a general statement, whenever we discuss joint distributions of continuous random variables, we discuss properties with respect to the joint density function instead of the joint distribution, as we do in the case of discrete variables. Of particular interest is the notion of (conditional) independence of continuous random variables. Definition 2.10 Let X, Y , and Z be sets of continuous random variables with joint density p(X, Y , Z). We say conditional that X is conditionally independent of Y given Z if independence p(x | z) = p(x | y, z) for all x, y, z such that p(z) > 0.

7 Expectation and Variance 7.1 Expectation expectation Let X be a discrete random variable that takes numerical values; then the expectation of X under the distribution P is IEP[X] = X x x · P(x). If X is a continuous variable, then we use the density function IEP[X] = Z x · p(x)dx. For example, if we consider X to be the outcome of rolling a fair die with probability 1/6 for each outcome, then IE[X] = 1 · 1 6 + 2 · 1 6 + · · · + 6 · 1 6 = 3.5. On the other hand, if we consider a biased die where P(X = 6) = 0.5 and P(X = x) = 0.1 for x < 6, then IE[X] = 1 · 0.1 + · · · + 5 · 0.1 + · · · + 6 · 0.5 = 4.5.

Often we are interested in expectations of a function of a random variable (or several random variables). Thus, we might consider extending the definition to consider the expectation of a functional term such as X2 + 0.5X. Note, however, that any function g of a set of random variables X1, . . . , Xk is essentially defining a new random variable Y : For any outcome , we define the value of Y as g(fX1(), . . . , fXk()). Based on this discussion, we often define new random variables by a functional term. For example Y = X2, or Y = eX. We can also consider functions that map values of one or more categorical random variables to numerical values. One such function that we use quite often is indicator function the indicator function, which we denote 11X = x. This function takes value 1 when X = x, and 0 otherwise. In addition, we often consider expectations of functions of random variables without bothering to name the random variables they define. For example IEP [X + Y]. Nonetheless, we should keep in mind that such a term does refer to an expectation of a random variable. We now turn to examine properties of the expectation of a random variable. First, as can be easily seen, the expectation of a random variable is a linear function in that random variable. Thus, IE[a · X + b] = aIE[X] + b. A more complex situation is when we consider the expectation of a function of several random variables that have some joint behavior. An important property of expectation is that the expectation of a sum of two random variables is the sum of the expectations. Proposition 2.4 IE[X + Y] = IE[X] + IE[Y]. linearity of This property is called linearity of expectation. It is important to stress that this identity is true expectation even when the variables are not independent. As we will see, this property is key in simplifying many seemingly complex problems. Finally, what can we say about the expectation of a product of two random variables? In general, very little: Example 2.5 Consider two random variables X and Y , each of which takes the value +1 with probability 1/2, and the value 1 with probability 1/2. If X and Y are independent, then IE[X · Y] = 0. On the other hand, if X and Y are correlated in that they always take the same value, then IE[X · Y] = 1. However, when X and Y are independent, then, as in our example, we can compute the expectation simply as a product of their individual expectations: Proposition 2.5 If X and Y are independent, then IE[X · Y] = IE[X] · IE[Y]. conditional We often also use the expectation given some evidence. The conditional expectation of X expectation given y is IEP [X | y] = X x x · P(x | y).

7.2 Variance The expectation of X tells us the mean value of X. However, It does not indicate how far X variance deviates from this value. A measure of this deviation is the variance of X. VVarP [X] = IEP h(X IEP [X])2i. Thus, the variance is the expectation of the squared dierence between X and its expected value. It gives us an indication of the spread of values of X around the expected value. An alternative formulation of the variance is VVar[X] = IEX2 (IE[X])2 . (2.16) (see exercise 2.11). Similar to the expectation, we can consider the expectation of a functions of random variables. Proposition 2.6 If X and Y are independent, then VVar[X + Y] = VVar[X] + VVar[Y]. It is straightforward to show that the variance scales as a quadratic function of X. In particular, we have: VVar[a · X + b] = a2VVar[X]. For this reason, we are often interested in the square root of the variance, which is called the standard standard deviation of the random variable. We define deviation X = pVVar[X]. The intuition is that it is improbable to encounter values of X that are farther than several standard deviations from the expected value of X. Thus, X is a normalized measure of "distance" from the expected value of X. As an example consider the Gaussian distribution of definition 2.7. Proposition 2.7 Let X be a random variable with Gaussian distribution N(, 2), then IE[X] = and VVar[X] = 2. Thus, the parameters of the Gaussian distribution specify the expectation and the variance of the distribution. As we can see from the form of the distribution, the density of values of X drops exponentially fast in the distance x . Not all distributions show such a rapid decline in the probability of outcomes that are distant from the expectation. However, even for arbitrary distributions, one can show that there is a decline. Theorem 2.1 (Chebyshev inequality): Chebyshev's inequality P (|X IEP [X]| t) VVarP [X] t2 .

We can restate this inequality in terms of standard deviations: We write t = kX to get P(|X IEP [X]| kX) 1 k2. Thus, for example, the probability of X being more than two standard deviations away from IE[X] is less than 1/4.

27.3 Foundation: Graph

Perhaps the most pervasive concept in this book is the representation of a probability distribution using a graph as a data structure. In this section, we survey some of the basic concepts in graph theory used in the book.

1 Nodes and Edges A graph is a data structure K consisting of a set of nodes and a set of edges. Throughout most this book, we will assume that the set of nodes is X = X1,...,Xn. A pair of nodes Xi,Xj directed edge can be connected by a directed edge Xi Xj or an undirected edge Xi—Xj. Thus, the set undirected edge of edges E is a set of pairs, where each pair is one of Xi Xj, Xj Xi, or Xi—Xj, for Xi,Xj X , i < j. We assume throughout the book that, for each pair of nodes Xi,Xj, at most one type of edge exists; thus, we cannot have both Xi Xj and Xj Xi, nor can we have Xi Xj and Xi—Xj.2 The notation Xi Xj is equivalent to Xj Xi, and the notation X j—Xi is equivalent to Xi—Xj. We use Xi Xj to represent the case where Xi and X j are connected via some edge, whether directed (in any direction) or undirected. In many cases, we want to restrict attention to graphs that contain only edges of one kind directed graph or another. We say that a graph is directed if all edges are either Xi Xj or Xj Xi. We usually denote directed graphs as G. We say that a graph is undirected if all edges are Xi—Xj. undirected graph We denote undirected graphs as H. We sometimes convert a general graph to an undirected graph by ignoring the directions on the edges. Definition 2.11 Given a graph K = (X, E), its undirected version is a graph H = (X, E0) where E0 = X—Y : graph's undirected version X Y E. Whenever we have that Xi Xj E, we say that Xj is the child of Xi in K, and that child Xi is the parent of Xj in K. When we have Xi—Xj E, we say that Xi is a neighbor of parent neighbor Xj in K (and vice versa). We say that X and Y are adjacent whenever X Y E. We use PaX to denote the parents of X, ChX to denote its children, and NbX to denote its neighbors. We define the boundary of X, denoted BoundaryX, to be PaX NbX; for DAGs, this set is boundary simply X's parents, and for undirected graphs X's neighbors.3 Figure 2.3 shows an example of a graph K. There, we have that A is the only parent of C, and F,I are the children of C. The degree only neighbor of C is D, but its adjacent nodes are A,D,F,I. The degree of a node X is the number of edges in which it participates. Its indegree is the number of directed edges Y X. indegree The degree of a graph is the maximal degree of a node in the graph. 2. Note that our definition is somewhat restricted, in that it disallows cycles of length two, where Xi Xj Xi, and allows self-loops where Xi Xi. 3. When the graph is not clear from context, we often add the graph as an additional argument.

2 Subgraphs In many cases, we want to consider only the part of the graph that is associated with a particular subset of the nodes. Definition 2.12 Let K = (X , E), and let X X . We define the induced subgraph K[X] to be the graph (X, E0) induced subgraph where E0 are all the edges X Y E such that X, Y X. For example, figure 2.4a shows the induced subgraph K[C, D, I]. A type of subgraph that is often of particular interest is one that contains all possible edges. Definition 2.13 A subgraph over X is complete if every two nodes in X are connected by some edge. The set X complete subgraph is often called a clique; we say that a clique X is maximal if for any superset of nodes Y X, clique Y is not a clique. Although the subset of nodes X can be arbitrary, we are often interested in sets of nodes that preserve certain aspects of the graph structure. Definition 2.14 We say that a subset of nodes X X is upwardly closed in K if, for any X X, we have that upward closure BoundaryX X. We define the upward closure of X to be the minimal upwardly closed subset

Y that contains X. We define the upwardly closed subgraph of X, denoted K+[X], to be the induced subgraph over Y , K[Y]. For example, the set A, B, C, D, E is the upward closure of the set C in K. The upwardly closed subgraph of C is shown in figure 2.4b. The upwardly closed subgraph of C, D, I is shown in figure 2.4c.

3 Paths and Trails Using the basic notion of edges, we can define dierent types of longer-range connections in the graph.

Definition path

We say that X1,...,XkX1,...,Xk form a path in the graph K=(X,E)K=(X,E) if, for every i=1,...,k1i=1,...,k1, we have that either XiXi+1XiXi+1 or XiXi+1XiXi+1. A path is directed if, for at least one i, we have XiXi+1XiXi+1.

Definition trail

We say that X1,...,XkX1,...,Xk form a trail in the graph K=(X,E)K=(X,E) if, for every i=1,...,k1i=1,...,k1, we have that XiXi+1XiXi+1.

In the graph KK of figure 2.3, A,C,D,E,IA,C,D,E,I is a path, and hence also a trail. On the other hand, A,C,F,G,DA,C,F,G,D is a trail, which is not a path.

Definition connected graph

A graph is connected if for every Xi,XjXi,Xj there is a trail between XiXi and XjXj.

We can now define longer-range relationships in the graph.

Definition ancestor, descendant

We say that XX is an ancestor of YY in K=(X,E)K=(X,E), and that YY is a descendant of XX, if there exists a directed path X1,...,XkX1,...,Xk with X1=XX1=X and Xk=YXk=Y. We use DescendantsXDescendantsX to denote X's descendants, AncestorsXAncestorsX to denote X's ancestors, and NonDescendantsXNonDescendantsX to denote the set of nodes in XDescendantsXXDescendantsX.

In our example graph K, we have that F,G,IF,G,I are descendants of CC. The ancestors of CC are AA, via the path A,C,A,C, and BB, via the path B,E,D,CB,E,D,C.

A final useful notion is that of an ordering of the nodes in a directed graph that is consistent with the directionality its edges.

Definition topological ordering

Let G=(X,E)G=(X,E) be a graph. An ordering of the nodes X1,...,XnX1,...,Xn is a topological ordering relative to KK if, whenever we have XiXjEXiXjE, then i<ji<j.

Appendix A.3.1 presents an algorithm for finding such a topological ordering.

4 Cycles and Loops Note that, in general, we can have a cyclic path that leads from a node to itself, making that node its own descendant.

Definition 2.20 A cycle in K is a directed path X1, . . . , Xk where X1 = Xk. A graph is acyclic if it contains no cycle acyclic cycles. For most of this book, we will restrict attention to graphs that do not allow such cycles, since it is quite dicult to define a coherent probabilistic model over graphs with directed cycles. DAG A directed acyclic graph (DAG) is one of the central concepts in this book, as DAGs are the basic graphical representation that underlies Bayesian networks. For some of this book, we also use acyclic graphs that are partially directed. The graph K of figure 2.3 is acyclic. However, if we add the undirected edge A—E to K, we have a path A, C, D, E, A from A to itself. Clearly, adding a directed edge E A would also lead to a cycle. Note that prohibiting cycles does not imply that there is no trail from a node to itself. For example, K contains several trails: C, D, E, I, C as well as C, D, G, F, C. An acyclic graph containing both directed and undirected edges is called a partially directed PDAG acyclic graph or PDAG. The acyclicity requirement on a PDAG implies that the graph can be chain component decomposed into a directed graph of chain components, where the nodes within each chain component are connected to each other only with undirected edges. The acyclicity of a PDAG guarantees us that we can order the components so that all edges point from lower-numbered components to higher-numbered ones. Definition 2.21 Let K be a PDAG over X . Let K1, . . . , K‘ be a disjoint partition of X such that: • the induced subgraph over Ki contains no directed edges; • for any pair of nodes X Ki and Y Kj for i < j, an edge between X and Y can only be a directed edge X Y . chain component Each component Ki is called a chain component. chain graph Because of its chain structure, a PDAG is also called a chain graph. Example 2.6 In the PDAG of figure 2.3, we have six chain components: A, B, C, D, E, F, G, H, and I. This ordering of the chain components is one of several possible legal orderings. Note that when the PDAG is an undirected graph, the entire graph forms a single chain component. Conversely, when the PDAG is a directed graph (and therefore acyclic), each node in the graph is its own chain component.

Dierent from a cycle is the notion of a loop: Definition 2.22 A loop in K is a trail X1, . . . , Xk where X1 = Xk. A graph is singly connected if it contains loop singly connected no loops. A node in a singly connected graph is called a leaf if it has exactly one adjacent node. leaf A singly connected directed graph is also called a polytree. A singly connected undirected graph is polytree called a forest; if it is also connected, it is called a tree. forest tree We can also define a notion of a forest, or of a tree, for directed graphs. Definition 2.23 A directed graph is a forest if each node has at most one parent. A directed forest is a tree if it is also connected. Note that polytrees are very dierent from trees. For example, figure 2.5 shows a graph that is a polytree but is not a tree, because several nodes have more than one parent. As we will discuss later in the book, loops in the graph increase the computational cost of various tasks. We conclude this section with a final definition relating to loops in the graph. This definition will play an important role in evaluating the cost of reasoning using graph-based representations. Definition 2.24 Let X1—X2— · · · —Xk—X1 be a loop in the graph; a chord in the loop is an edge connecting chordal graph Xi and Xj for two nonconsecutive nodes Xi, Xj. An undirected graph H is said to be chordal if any loop X1—X2— · · · —Xk—X1 for k 4 has a chord. Thus, for example, a loop A—B—C—D—A (as in figure 1.1b) is nonchordal, but adding an edge A—C would render it chordal. In other words, in a chordal graph, the longest "minimal loop" (one that has no shortcut) is a triangle. Thus, chordal graphs are often also called triangulated triangulated. graph We can extend the notion of chordal graphs to graphs that contain directed edges. Definition 2.25 A graph K is said to be chordal if its underlying undirected graph is chordal.

27.4 Bayesian Network

A Bayesian network is a graphical model that encodes probabilistic relationships among variables of interest. When used in conjunction with statistical techniques, the graphical model has several advantages for data analysis. One, because the model encodes dependencies among all variables, it readily handles situations where some data entries are missing. Two, a Bayesian network can be used to learn causal relationships, and hence can be used to gain understanding about a problem domain and to predict the consequences of intervention.

Three, because the model has both a causal and probabilistic semantics, it is an ideal representation for combining prior knowledge (which often comes in causal form) and data. Four, Bayesian statistical methods in conjunction with bayesian networks offer an efficient and principled approach for avoiding the overfitting of data. In this paper, we discuss methods for constructing Bayesian networks from prior knowledge and summarize Bayesian statistical methods for using data to improve these models. With regard to the latter task, we describe methods for learning both the parameters and structure of a Bayesian network, including techniques for learning with incomplete data. In addition, we relate Bayesian-network methods for learning to techniques for supervised and unsupervised learning. We illustrate the graphical-modeling approach using a real-world case study.

A Non-Causal Bayesian Network Example Figure 1 shows a simple Bayesian network, which consists of only two nodes and one link. It represents the JPD of the variables Eye Color and Hair Color in a population of students (Snee, 1974). In this case, the conditional probabilities of Hair Color given the values of its parent node, Eye Color, are provided in a CPT. It is important to point out that this Bayesian network does not contain any causal assumptions, i.e. we have no knowledge of the causal order between the variables. Thus, the interpretation of this network should be merely statistical (informational).

A Causal Network Example Figure 2 illustrates another simple yet typical Bayesian network. In contrast to the statistical relationships in Figure 1, the diagram in Figure 2 describes the causal relationships among the seasons of the year (X1X1), whether it is raining (X2X2), whether the sprinkler is on (X3X3), whether the pavement is wet (X4X4), and whether the pavement is slippery (X5X5). Here, the absence of a direct link between X1X1 and X5X5, for example, captures our understanding that there is no direct influence of season on slipperiness. The influence is mediated by the wetness of the pavement (if freezing were a possibility, a direct link could be added).

A Dynamic Bayesian Network Example Entities that live in a changing environment must keep track of variables whose values change over time. Dynamic Bayesian networks capture this process by representing multiple copies of the state variables, one for each time step. A set of variables Xt1Xt1 and XtXt denotes the world state at times t-1 and t respectively. A set of evidence variables Et denotes the observations available at time t. The sensor model P(Et|Xt)P(Et|Xt) is encoded in the conditional probability distributions for the observable variables, given the state variables. The transition model P(Xt|Xt1)P(Xt|Xt1) relates the state at time t-1 to the state at time t. Keeping track of the world means computing the current probability distribution over world states given all past observations, i.e. P(Xt|E1,. . . ,Et)P(Xt|E1,. . . ,Et).

Dynamic Bayesian networks (DBN) are a generalization of Hidden Markov Models (HMM) and Kalman Filters (KF). Every HMM and KF can be represented with a DBN. Furthermore, the DBN representation of an HMM is much more compact and, thus, much better understandable. The nodes in the HMM represent the states of the system, whereas the nodes in the DBN represent the dimensions of the system. For example, the HMM representation of the valve system in Figure 2.3 is made of 26 nodes and 36 arcs, versus 9 nodes and 11 arcs in the DBN (Weber and Jouffe, 2003).

27.5 Template Models for Bayesian Networks

In many cases, we need to model distributions that have a recurring structure. In this module, we describe representations for two such situations. One is temporal scenarios, where we want to model a probabilistic structure that holds constant over time; here, we use Hidden Markov Models, or, more generally, Dynamic Bayesian Networks. The other is aimed at scenarios that involve multiple similar entities, each of whose properties is governed by a similar model; here, we use Plate Models.

Temporal Models Our focus in this section is on modeling dynamic settings, where we are interested in reasoning about the state of the world as it evolves over time. We can model such settings in terms of a system state system state, whose value at time t is a snapshot of the relevant attributes (hidden or observed) of the system at time t. We assume that the system state is represented, as usual, as an assignment of values to some set of random variables X . We use X (t) i to represent the instantiation of the variable Xi at time t. Note that Xi itself is no longer a variable that takes a value; rather, it is a template variable template variable. This template is instantiated at dierent points in time t, and each Xi (t) is a variable that takes a value in Val(Xi). For a set of variables X X , we use X (t1:t2) (t1 $<$ t2) to denote the set of variables X (t) : t [t1,t2]. As usual, we use the notation x(t:t0) for an assignment of values to this set of variables.

Each "possible world" in our probability space is now a trajectory: an assignment of values to each variable X (t) i for each relevant time t. Our goal therefore is to represent a joint distribution over such trajectories. Clearly, the space of possible trajectories is a very complex probability space, so representing such a distribution can be very dicult. We therefore make a series of simplifying assumptions that help make this representational problem more tractable.

Dynamic Bayesian Networks

Directed Probabilistic Models for Object-Relational Domains Based on the framework described in the previous section, we now describe template-based representation languages that can encode directed probabilistic models.

Plate Models We begin our discussion by presenting the plate model, the simplest and best-established of the object-relational frameworks. Although restricted in several important ways, the plate modeling framework is perhaps the approach that has been most commonly used in practice, notably for encoding the assumptions made in various learning tasks. This framework also provides an excellent starting point for describing the key ideas of template-based languages and for motivating some of the extensions that have been pursued in richer languages.

In the plate formalism, object types are called plates. The fact that multiple objects in the class share the same set of attributes and same probabilistic model is the basis for the use of the term "plate," which suggests a stack of identical objects. We begin with some motivating examples and then describe the formal framework.

Examples Example 1 The simplest example of a plate model, shown in figure 6.6, describes multiple random variables generated from the same distribution. In this case, we have a set of random variables X(d) (dD)X(d) (dD) that all have the same domain Val(X) and are sampled from the same distribution. In a plate representation, we encode the fact that these variables are all generated from the same template by drawing only a single node X(d) and enclosing it in a box denoting that d ranges over D, so that we know that the box represents an entire "stack" of these identically distributed variables. This box plate is called a plate, with the analogy that it represents a stack of identical plates.

27.6 Factor Graph

A factor graph is a bipartite graph representing the factorization of a function.

Each edge in graph defines a function

Definition A factor graph is a bipartite graph representing the factorization of a function.

Related Readings [1]: Factor Graph, wikipedia.org

27.7 Inference

This addresses the question of probabilistic inference: how a PGM can be used to answer questions.

Even though a PGM generally describes a very high dimensional distribution, its structure is designed so as to allow questions to be answered efficiently. The course presents both exact and approximate algorithms for different types of inference tasks, and discusses where each could best be applied. The (highly recommended) honors track contains two hands-on programming assignments, in which key routines of the most commonly used exact and approximate algorithms are implemented and applied to a real-world problem.

27.8 Learning

This course addresses the question of learning: how a PGM can be learned from a data set of examples.

The course discusses the key problems of parameter estimation in both directed and undirected models, as well as the structure learning task for directed models. The (highly recommended) honors track contains two hands-on programming assignments, in which key routines of two commonly used learning algorithms are implemented and applied to a real-world problem.

27.9 An Introduction to UnBBayes

UnBBayes is a probabilistic network framework written in Java. It has both a GUI and an API with inference, sampling, learning and evaluation. It supports Bayesian networks, influence diagrams, MSBN, OOBN, HBN, MEBN/PR-OWL, PRM, structure, parameter and incremental learning.

Features Probabilistic Networks: Bayesian Network (BN) Junction Tree Likelihood Weighting Gibbs Influence Diagram (ID) Multiply Sectioned Bayesian Network (MSBN) Hybrid Bayesian Network (HBN) Gaussian Mixture - Propagation under development Object-Oriented Bayesian Network (OOBN) FOL Probabilistic Network: Multi-Entity Bayesian Network (MEBN) Probabilistic

Ontology Language (PR-OWL) Learning Bayesian Network: K2 B CBL-A CBL-B Incremental Learning Sampling Logic Likelihood Weighting Gibbs Classification Performance Evaluation Evaluation using Logic Sampling Evaluation using Likelihood Weighting Sampling Installation Go to https://sourceforge.net/projects/unbbayes/files/latest/dow 4.21.18.$ziptounbbayes-$4.21.18$folderOpenunbbayes-$4.21.18$folder, doubleclicktounbbayes.batunbbayes-$ 4.21.18$open$

Official Videos In this section, I add some official videos from unbbayes team. There are overview

Overview In this video we are going to show the basic function we have in UnBBayes. This is the first of many tutorials we have been creating to support the demand for documentation on how to use UnBBayes. We hope this will help UnBBayes' user community to grow even more.

Bayesian Network In this video we are going to show how to create and compile a Bayesian Network (BN) in UnBBayes. This is our second of many video tutorials we have been creating to support the demand for documentation on how to use UnBBayes. We hope this will help UnBBayes' user community to grow even more.

UnBBayes Performance Evaluation for Multi-Sensor Classification Systems In this video we are going to show how to do a performance evaluation for multi-sensor classification systems in UnBBayes. It has been a while we do not post new videos, but hopefully this third one is just one more of many tutorials we will have available to support the demand for documentation on how to use UnBBayes. We hope this will help UnBBayes' user community to grow even more.

Probabilistic Ontology Modeling Using UnBBayes In this video we discuss how to model probabilistic ontologies using PR-OWL/MEBN in UnBBayes. This session was a video conference between PhD students from the Institute of Business Administration (http://www.iba.edu.pk) and Rommel Carvalho from George Mason University (http://www.gmu.edu).

27.10 Medical Domain Data

We have provided you with a joint probability distribution of symptons, conditions and diseases based on the "flu" example in class. Certain diseases are more likely than others given certain symptons, and a model such as this can be used to help doctors make a diagnosis. (Don't actually use this for diagnosis, though!). The ground-truth joint probability distribution consists of twelve binary random variables and contains 212212 possible configurations (numbered 0 to 4095), which is small enough that you can enumerate them exhaustively. The variables are as follows:

(0) IsSummer true if it is the summer season, false otherwise. (1) HasFlu true if the patient has the flu. (2) HasFoodPoisoning true if the patient has food poisoning. (3) HasHayFever true if patient has hay fever. (4) HasPneumonia true if the patient has pneumonia. (5) HasRespiratoryProblems true if the patient has problems in the respiratory system. (6) HasGastricProblems true if the patient has problems in the gastro-intestinal system. (7) HasRash true if the patient has a skin rash. (8) Coughs true if the patient has a cough. (9) IsFatigued true if the patient is tired and fatigued. (10) Vomits true if the patient has vomited. (11) HasFever true if the patient has a high fever. You can download all the

data here. The archive contains two files:

joint.dat: The true joint probability distribution over the twelve binary variables. Since each variable is binary, we can represent a * full variable assignment as a bitstring. This file lists all $2^12assignments(oneineachline)aspairs"IntegerProbability"where"Integer"$
$0andtrue = 1, anassignmenttoallvariablesresultsina12-bitbinarynumber(withtheindexofthevariablessho$
$Thedatasetconsistsofsamplesfromtheaboveprobabilitydistribution.Eachlineofthefilecontainsacompleteas$

27.11 Optical Word Recognition

We will be studying the computer vision task of recognizing words from images. The task of recognizing words is usually decomposed to recognition of individual characters from their respective images (optical character recognition, OCR), and hence inferring the word. However character recognition is often a very difficult task, and since each character is predicted independent of its neighbors, its results can often contain combinations of characters that may not be possible in English. In this homework we will augment a simple OCR model with additional factors that capture some intuitions based on character co-occurences and image similarities.

The undirected graphical model for recognition of a given word is given in the figure above. It consists of two types of variables:

Image Variables: These are observed images that we need to predict the corresponsing character of, and the number of these image variables for a word is the number of characters in the word. The value of these image variables is an observed image, represented by an integer id (less than 1000). For the description of the model, assume the id of the image at position i is represented by img(i). Character Variables: These are unobserved variables that represent the character prediction for each of the images, and there is one of these for each of the image variables. For our dataset, the domain of these variables is restricted to the ten most frequent characters in the English language (e,t,a,o,i,n,s,h,r,d [ciation]), instead of the complete alphabet. For the discussion below, assume the predicted character at position i is represented by char(i). The model for a word w will consist of len(w) observed image ids, and the same number of unobserved character variables. For a given assignment to these character variables, the model score will be specified using three types of factors:

OCR Factors, oo : These factors capture the predictions of a character-based OCR system, and hence exist between every image variable and its corresponding character variable. The number of these factors of word w is len(w). The value of factor between an image variable and the character variable at position i is dependent on img(i) and char(i), and is stored in ocr.dat file described in the data section. Transition Factors, tt : Since we also want to represent the co-occurence frequencies of the characters in our model, we add these factors between all consecutive character variables. The number of these factors of word w is len(w)-1. The value of factor between two character variables at positions i and i+1 is dependent on char(i) and char(i+1), and is high if char(i+1) is frequently preceded by char(i) in english words. These values are given to you in trans.dat file described in the data section. Skip Factors, ss : Another intuition that we would like to capture in our model is that similar images in a word always represent the same character. Thus our model score should be higher if it predicts the same characters for similar images. These factors exist between every pair of image variables that have the same id, i.e. this factor exist between all i,j, i!=j such that img(i)==img(j). The value of this factor depends on char(i) and char(j), and is 5.0 if char(i)==char(j), and 1.0 otherwise. You can download all the data here. The archive contains the following files:

ocr.dat: Contains the output predictions of a pre-existing OCR system for the set of thousand images. Each row contains three tab separated values "id a prob" and represents the OCR system's probability that image id represents character aa, p(char=a|img=id)=probp(char=a|img=id)=prob. Use these values directly as the value of the factor between image and character variables at position ii, o(image(i)=id,char(i)=a)=probo(image(i)=id,char(i)=a)=prob. Since there are 10 characters and 1000 images, the total number of rows in this file is 10,000. trans.dat: Stores the factor potentials for the transition factors. Each row contains three tab-separated values "a b value" that represents the value of factor when the previous character is "a" and the next character is "b", i.e. (char(i)=a, char(i+1)=b) = value. The number of rows in the file is 100 (10*10). data.dat (and truth.dat): Dataset to run your experiments on (see Core Tasks below). The observed dataset (data.dat) consists observed images of one word on each row. The observed images for a word are represented by a sequence of tab-separated integer ids ("id1 id2 id3"). The true word for these observed set of images is stored the respective row in truth.dat, and is simply a string ("eat"). For the core task (3) below, you should iterate through both the files together to ensure you have the true word along with the observed images. Extra files (bicounts.dat, allwords.dat, allimagesX.dat): These files are not necessary for the core tasks, but may be useful for further fun and your own exploration. allwords.dat and allimagesX.dat are larger versions of data.dat and truth.dat, i.e. they contain all possible words that can be generated from our restricted set of alphabet, and five samples of their observed image sequences (one in each file). You can run inference on these if you like, but is likely to take 15-20 times longer than the small dataset. bicount.dat is in the same format as trans.dat, but instead of storing inexplicable potentials, it stores the joint probability of the co-occurences of the characters. Core Task 1. Graphical Model: Implement the graphical model containing the factors above. For any given assignment to the character variables, your model should be able to calculate the model score. Implemention should allow switching between three models:

OCR model: only contains the OCR factors Transition model: contains OCR and Transition factors Combined model: containing all three types of factors Note: To avoid errors arising from numerical issues, we suggest you represent the factors in the log-space and take sums as much as possible, calculating the log of the model score.

2. Exhaustive Inference: Using the graphical model, write code to perform exhaustive inference, i.e. your code should be able to calculate the probability of any assignment of the character and image variables. To calculate the normalization constant Z for the word w, you will need to go through all possible assignments to the character variables (there will be 10len(w)10len(w) of these).

3. Model Accuracy: Run your model on the data given in the file data.dat. For every word in the dataset, pick the assignment to character variables that has the highest probability according to the model, and treat this as the model prediction for the word. Using the truth given in truth.dat, compare the accuracy of the model predictions using the following three metrics: 1. Character-wise accuracy: Ratio of correctly predicted characters to total number of characters 2.

Word-wise accuracy: Ratio of correctly predicted words to total number of words
3. Average Dataset log-likelihood: For each word given in data.dat, calculate the log of the probability of the true word according to the model. Compute the average of this value for the whole dataset.

Compare all of the three models described in (1) using these three metrics. Also give some examples of words that were incorrect by the OCR model but consequently fixed by the Transition model, and examples of words that were incorrect by the OCR, partially corrected by the Transition model, and then completely fixed by the Combined model.

Chương 28

Học sâu

View online http://magizbox.com/training/deep$_learning/site/$

Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence.

28.1 Get Started

Tensorflow Install tensorflow in Windows Anaconda Anaconda is the leading open data science platform powered by Python. The open source version of Anaconda is a high performance distribution of Python and R and includes over 100 of the most popular Python, R and Scala packages for data science.

Step 1: Download the Anaconda installer

Step 2: Double click the Anaconda installer and follow the prompts to install to the default location.

After a successful installation you will see output like this:

CUDA Toolkit 8.0 The NVIDIA CUDA Toolkit provides a comprehensive development environment for C and C++ developers building GPU-accelerated applications. The CUDA Toolkit includes a compiler for NVIDIA GPUs, math libraries, and tools for debugging and optimizing the performance of your applications. You'll also find programming guides, user manuals, API reference, and other documentation to help you get started quickly accelerating your application with GPUs.

Step 1: Verify the system has a CUDA-capable GPU.

Step 2: Download the NVIDIA CUDA Toolkit.

Step 3: Install the NVIDIA CUDA Toolkit.

Step 4: Test that the installed software runs correctly and communicates with the hardware.

cuDNN The NVIDIA CUDA Deep Neural Network library (cuDNN) is a GPU-accelerated library of primitives for deep neural networks. cuDNN provides highly tuned implementations for standard routines such as forward and backward convolution, pooling, normalization, and activation layers. cuDNN is part of the NVIDIA Deep Learning SDK.

Step 1: Register an NVIDIA developer account

Step 2: Download cuDNN v5.1, you will get file like that cudnn-8.0-windows7-x64-v5.1.zip

Step 3: Copy CUDNN files to CUDA install

Extract your cudnn-8.0-windows7-x64-v5.1.zip file, and copy files to corresponding CUDA folder

In my environment, CUDA installed in C:FilesGPU Computing Toolkit8̌.0, you must copy append three folders bin, include, lib

Install Tensorflow Package CPU TensorFlow environment

conda create –name tensorflow python=3.5 activate tensorflow conda install -y jupyter scipy pip install tensorflow GPU TensorFlow environment

conda create –name tensorflow-gpu python=3.5 activate tensorflow-gpu conda install -y jupyter scipy pip install tensorflow-gpu word2vec Example Step 1: Download word2vec example from github

dir

02/06/2017 11:45 DIR . 02/06/2017 11:45 DIR .. 02/06/2017 10:12 9,144 word2vec$_basic.pyStep2 : Runword2vec_basicexample$

$activatetensorflow - gpu$ python word2vec$_basic.py$

Found and verified text8.zip Data size 17005207 Most common words (+UNK) [['UNK', 418391], ('the', 1061396), ('of', 593677), ('and', 416629), ('one', 411764)] Sample data [5241, 3082, 12, 6, 195, 2, 3136, 46, 59, 156] ['anarchism', 'originated', 'as', 'a', 'term', 'of', 'abuse', 'first', 'used', 'against'] 3082 originated -> 5241 anarchism 3082 originated -> 12 as 12 as -> 6 a 12 as -> 3082 originated 6 a -> 195 term 6 a -> 12 as 195 term -> 2 of 195 term -> 6 a Initialized Average loss at step 0 : 288.173675537 Nearest to its: nasl, tinkering, derivational, yachts, emigrated, fatalism, kingston, kochi, Nearest to into: streetcars, neglecting, deutschlands, lecture, realignment, bligh, donau, medalists, Nearest to state: canterbury, exceptions, disaffection, crete, westernmost, earthly, organize, richland,

28.2 Tài liệu Deep Learning

Lang thang thế nào lại thấy trang này My Reading List for Deep Learning! của một anh ở Microsoft. Trong đó, (đương nhiên) có Deep Learning của thánh Yoshua Bengio, có một vụ hay nữa là bài review "Deep Learning" của mấy thánh Yann Lecun, Yoshua Bengio, Geoffrey Hinton trên tạp chí Nature. Ngoài ra còn có nhiều tài liệu hữu ích khác.

28.3 Các layer trong deep learning

28.3.1 Sparse Layers

nn.Embedding (hướng dẫn)

grep code: Shawn1993/cnn-text-classification-pytorch

Đóng vai trò như một lookup table, map một word với dense vector tương ứng

28.3.2 Convolution Layers

nn.Conv1d, nn.Conv2d, nn.Conv3d)

grep code: Shawn1993/cnn-text-classification-pytorch, galsang/CNN-sentence-classification-pytorch

Các tham số trong Convolution Layer

* *kernel_size* (hay là filter size)

Đối với NLP, *kernel_size* thường bằng $region\_size * word\_dim$ (đối với conv1d) hay ($region\_size$, $word\_dim$) đối với conv2d

<small>Quá trình tạo feature map đối với region size bằng 2</small> `![](https://media.giphy.com/media/l2QE2y1UQP7vIgiti/giphy.gif)`

* ‘in$_c$*hannels*‘, ‘*out*$_c$*hannels*‘(*lslng*‘*featuremaps*‘)

Kênh (channels) là các cách nhìn (view) khác nhau đối với dữ liệu. Ví dụ, trong ảnh thường có 3 kênh RGB (red, green, blue), có thể áp dụng convolution giữa các kênh. Với văn bản cũng có thể có các kênh khác nhau, như khi có các kênh sử dụng các word embedding khác nhau (word2vec, GloVe), hoặc cùng một câu nhưng biểu diễn ở các ngôn ngữ khác nhau.

* ‘stride‘

Định nghĩa bước nhảy của filter.

`![](http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/11/Screen-Shot-2015-11-05-at-10.18.08-AM-1024x251.png)`

Hình minh họa sự khác biệt giữa các feature map đối với stride=1 và stride=2. Feature map đối với stride = 1 có kích thước là 5, feature map đối với stride = 3 có kích thước là 3. Stride càng lớn thì kích thước của feature map càng nhỏ.

Trong bài báo của Kim 2014, ‘stride = 1‘ đối với ‘nn.conv2d‘ và ‘stride = word$_d$*im*‘*Øivi*‘*nn.conv1d*‘

Toàn bộ tham số của mạng CNN trong bài báo Kim 2014, `![](http://d3kbpzbmcynnmx.cloudfront.net/wp-content/uploads/2015/11/Screen-Shot-2015-11-06-at-8.03.47-AM.png)`

| Description | Values | |————————|——————| | input word vectors | Google word2vec | | filter region size | (3, 4, 5) | | feature maps | 100 | | activation function | ReLU | | pooling | 1-max pooling | | dropout rate | 0.5 | | $latexlamp;s = 22$ norm constraint | 3 |

Đọc thêm:

* [Lecture 13: Convolutional Neural Networks (for NLP). CS224n-2017](http://web.stanford.edu/class/cs2 2017-lecture13-CNNs.pdf) * [DeepNLP-models-Pytorch - 8. Convolutional Neural Networks](https://nbviewer.jupyter.org/github/DSKSD/DeepNLP-models-Pytorch/blob/master/notebooks/08.CNN-for-Text-Classification.ipynb) * [A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification. Zhang 2015](https://arxiv.org/pdf/1510.03820.pdf)

28.4 Recurrent Neural Networks

What are RNNs? The idea behind RNNs is to make use of sequential information. In a traditional neural network we assume that all inputs (and outputs) are independent of each other. But for many tasks that's a very bad idea. If you want to predict the next word in a sentence you better know which words came before it. RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being depended on the previous computations. Another way to think about RNNs is that they have a "memory" which captures information about what has been calculated so far. In theory

RNNs can make use of information in arbitrarily long sequences, but in practice they are limited to looking back only a few steps (more on this later). Here is what a typical RNN looks like:

A recurrent neural network and the unfolding in time of the computation involved in its forward computation

A recurrent neural network and the unfolding in time of the computation involved in its forward computation. Source: Nature The above diagram shows a RNN being unrolled (or unfolded) into a full network. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word. The formulas that govern the computation happening in a RNN are as follows:

xtxt is the input at time step tt. For example, x1x1 could be a one-hot vector corresponding to the second word of a sentence. stst is the hidden state at time step tt. It's the "memory" of the network. stst is calculated based on the previous hidden state and the input at the current step: st=f(Uxt+Wst1)st=f(Uxt+Wst1). The function ff usually is a nonlinearity such as tanh or ReLU. s1s1, which is required to calculate the first hidden state, is typically initialized to all zeroes. otot is the output at step tt. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary. ot=softmax(Vst)ot=softmax(Vst). There are a few things to note here:

You can think of the hidden state stst as the memory of the network. stst captures information about what happened in all the previous time steps. The output at step otot is calculated solely based on the memory at time tt. As briefly mentioned above, it's a bit more complicated in practice because stst typically can't capture information from too many time steps ago. Unlike a traditional deep neural network, which uses different parameters at each layer, a RNN shares the same parameters (UU, VV, WW above) across all steps. This reflects the fact that we are performing the same task at each step, just with different inputs. This greatly reduces the total number of parameters we need to learn. The above diagram has outputs at each time step, but depending on the task this may not be necessary. For example, when predicting the sentiment of a sentence we may only care about the final output, not the sentiment after each word. Similarly, we may not need inputs at each time step. The main feature of an RNN is its hidden state, which captures some information about a sequence. What can RNNs do? RNNs have shown great success in many NLP tasks. At this point I should mention that the most commonly used type of RNNs are LSTMs, which are much better at capturing long-term dependencies than vanilla RNNs are. But don't worry, LSTMs are essentially the same thing as the RNN we will develop in this tutorial, they just have a different way of computing the hidden state. We'll cover LSTMs in more detail in a later post. Here are some example applications of RNNs in NLP (by non means an exhaustive list).

Language Modeling and Generating Text Given a sequence of words we want to predict the probability of each word given the previous words. Language Models allow us to measure how likely a sentence is, which is an important input for Machine Translation (since high-probability sentences are typically correct). A side-effect of being able to predict the next word is that we get a generative model, which allows us to generate new text by sampling from the output probabilities. And depending on what our training data is we can generate all kinds of stuff. In Language Modeling our input is typically a sequence of words (encoded as one-hot vectors for example), and our output is the sequence of predicted words. When training the network we set ot=xt+1ot=xt+1 since we want the output at step tt to be the actual next word.

Research papers about Language Modeling and Generating Text:

Recurrent neural network based language model Extensions of Recurrent neural network based language model Generating Text with Recurrent Neural Networks Machine Translation Machine Translation is similar to language modeling in that our input is a sequence of words in our source language (e.g. German). We want to output a sequence of words in our target language (e.g. English). A key difference is that our output only starts after we have seen the complete input, because the first word of our translated sentences may require information captured from the complete input sequence.

RNN for Machine Translation

RNN for Machine Translation. Image Source: http://cs224d.stanford.edu/lectures/CS224d-Lecture8.pdf

Research papers about Machine Translation:

A Recursive Recurrent Neural Network for Statistical Machine Translation Sequence to Sequence Learning with Neural Networks Joint Language and Translation Modeling with Recurrent Neural Networks Speech Recognition Given an input sequence of acoustic signals from a sound wave, we can predict a sequence of phonetic segments together with their probabilities.

Research papers about Speech Recognition:

Towards End-to-End Speech Recognition with Recurrent Neural Networks Generating Image Descriptions Together with convolutional Neural Networks, RNNs have been used as part of a model to generate descriptions for unlabeled images. It's quite amazing how well this seems to work. The combined model even aligns the generated words with features found in the images.

Deep Visual-Semantic Alignments for Generating Image Descriptions. Source: http://cs.stanford.edu/people/karpathy/deepimagesent/

Training RNNs Training a RNN is similar to training a traditional Neural Network. We also use the backpropagation algorithm, but with a little twist. Because the parameters are shared by all time steps in the network, the gradient at each output depends not only on the calculations of the current time step, but also the previous time steps. For example, in order to calculate the gradient at t=4t=4 we would need to backpropagate 3 steps and sum up the gradients. This is called Backpropagation Through Time (BPTT). If this doesn't make a whole lot of sense yet, don't worry, we'll have a whole post on the gory details. For now, just be aware of the fact that vanilla RNNs trained with BPTT have difficulties learning long-term dependencies (e.g. dependencies between steps that are far apart) due to what is called the vanishing/exploding gradient problem. There exists some machinery to deal with these problems, and certain types of RNNs (like LSTMs) were specifically designed to get around them.

RNN Extensions Over the years researchers have developed more sophisticated types of RNNs to deal with some of the shortcomings of the vanilla RNN model. We will cover them in more detail in a later post, but I want this section to serve as a brief overview so that you are familiar with the taxonomy of models.

Bidirectional RNNs are based on the idea that the output at time t may not only depend on the previous elements in the sequence, but also future elements. For example, to predict a missing word in a sequence you want to look at both the left and the right context. Bidirectional RNNs are quite simple. They are just two RNNs stacked on top of each other. The output is then computed based on the hidden state of both RNNs.

Deep (Bidirectional) RNNs are similar to Bidirectional RNNs, only that we now have multiple layers per time step. In practice this gives us a higher learning capacity (but we also need a lot of training data).

Deep Bidirectional RNN LSTM networks are quite popular these days and we briefly talked about them above. LSTMs don't have a fundamentally different architecture from RNNs, but they use a different function to compute the hidden state. The memory in LSTMs are called cells and you can think of them as black boxes that take as input the previous state ht1ht1 and current input xtxt. Internally these cells decide what to keep in (and what to erase from) memory. They then combine the previous state, the current memory, and the input. It turns out that these types of units are very efficient at capturing long-term dependencies. LSTMs can be quite confusing in the beginning but if you're interested in learning more this post has an excellent explanation.

Conclusion So far so good. I hope you've gotten a basic understanding of what RNNs are and what they can do. In the next post we'll implement a first version of our language model RNN using Python and Theano. Please leave questions in the comments!

BTS

22/11/2017 - Phải nói quyển này hơi nặng so với mình. Nhưng thôi cứ cố gắng vậy. 24/11/2017 - Từ hôm nay, mỗi ngày sẽ ghi chú một phần (rất rất nhỏ) về Deep Learning [tại đây](https://docs.google.com/document/d/1KxDrw5s6uYHNLda7t0rhp0RM$_TlUGxydQ-Qi1JOPFr8/edit?usp = sharing$)

[1] : $[UnderstandingConvolutionalNeuralNetworksforNLP](http : //www.wildml.com/2015/11/unde$ $convolutional-neural-networks-for-nlp)$[2] : $http : //pytorch.org/docs/master/nn.html$

Chương 29

Xử lý ngôn ngữ tự nhiên

View online http://magizbox.com/training/natural$_language_processing/site/$

05/01/2018: "điên đầu" với Sphinx và HTK

HTK thì đã bỏ rồi vì quá lằng nhằng.

Sphinx thì setup được đối với dữ liệu nhỏ rồi. Nhưng không thể làm nó hoạt động với dữ liệu của VIVOS. Chắc hôm nay sẽ switch sang Kaldi vậy.

26/12/2017: Automatic Speech Recognition 100

Sau mấy ngày "vật lộn" với code base của Truong Do, thì cuối cùng cũng produce voice được. Cảm giác rất thú vị. Quyết định làm luôn ASR. Tìm mãi chẳng thấy code base đâu (chắc do lĩnh vực mới nên không có kinh nghiệm). May quá lại có bạn frankydotid có project về nhận diện tiếng Indonesia ở [github](https://github.com/frankydotid/Indonesian-Speech-Recognition). Trong README.md bạn đấy bảo là phải cần đọc HTK Book. Tốt quá đang cần cơ bản.

20/12/2017: Text to speech 100

Cảm ơn project rất hay của [bạn Truong Do ở vais](https://vais.vn/vi/tai-ve/hts$_for_vietnamese/), nukhngcprojectnychcmnhphimtrtnhiuthigianmic\emptyset cphinbntexttospeech\emptyset utin.$

Tóm lại thì việc sinh ra tiếng nói từ text gồm 4 giai đoạn

1. Sinh ra features từ file wav sử dụng tool sptk 2. Tạo một lab, trong đó có dữ liệu huấn luyện (những đặc trưng của âm thanh được trích xuất từ bước 1), text đầu vào 3. Sử dụng htk để train dữ liệu từ thư mục lab, đầu ra là một model 4. Sử dụng model để sinh ra output với text đầu vào, dùng hts$_engine\emptyset decode, ktqu\emptyset cwavfiles.$

Phù. 4 bước đơn giản thế này thôi mà không biết. Lục cả internet ra mãi chẳng hiểu, cuối cùng file phân tích file ‘train.sh‘ của bạn Truong Do mới hiểu. Ahihi

24/11/2017: Nhánh của Trí tuệ nhân tạo mà hiện tại mình đang theo đuổi. Project hiện tại là [underthesea](https://github.com/magizbox/underthesea). Với mục đích là xây dựng một toolkit cho xử lý ngôn ngữ tự nhiên tiếng Việt.

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages.

29.1 Introduction to Natural Language Processing

Natural language processing (NLP) is a field of computer science, artificial intelligence, and computational linguistics concerned with the interactions between computers and human (natural) languages.

NLP is related to the area of human–computer interaction. Many challenges in NLP involve: natural language understanding, enabling computers to derive meaning from human or natural language input; and others involve natural language generation.

The input and output of an NLP system can be either speech or written text.

Components of NLP

There are two components of NLP as given

Natural Language Understanding (NLU): this task mapping the given input in natural language into useful representations and analyzing different aspects of the language. Natural Language Generation (NLG): In the process of producing meaningful phrases and sentences in the form of natural language form some internal representation. It involves text planning retrieve the relevant content from knowledge base, sentence planning choose required words, forming meaningful phrases, setting tone of the sentence, text realization map sentence plan into sentence structure. Difficulties

Natural Language has an extremely rich form and structure. It is very ambiguous. There can be different levels of ambiguity

Lexical ambiguity: it is at very primitive level such as word-level. For example, treating the word "board" as noun or verb? Syntax level ambiguity: A sentence be parsed in different ways. For example, "He lifted the beetle with the red cap?" - did he use cap to lift the beetle or he lifted a beetle that had red cap? Referential ambiguity: referring to something using pronouns. For example, Rima went to Gauri. She said "I am tired". - Exactly who is tired? One input can mean different meanings. Many inputs can mean the same thing.

29.2 Natural Language Processing Tasks

The analysis of natural language is broken into various board levels such as phonological, morphological, syntactic, semantic, pragmatic and discourse analysis.

Phonological Analysis Phonology is analysis of spoken language. Therefore, it deals with speech recognition and generation. The core task of speech recognition and generation system is to take an acoustic waveform as input and produce as output, a string of words. The phonology is a part of natural language analysis, which deals with it. The area of computational linguistics that deals with speech analysis is computational phonology

Example: Hans Rosling's shortest TED talk

Original Sound

0:00 / 0:52

Text X means unknown but the world is pretty known it's seven billion people have seven stones. One billion can save money to fly abroad on holiday every year. One billion can save money to keep a car or buy a car. And then three billion they save money to pay the by be a bicycle or perhaps a two-wheeler. And two billion they are busy saving money to buy shoes. In the future they will get rich and these people we move over here, these people will move over here, we will have two billion more in the world like this and the question is whether the rich people over there are prepared to be integrated in the world with 10 bilions people. Auto generated sound

0:00 / 0:36

Morphological Analysis It is the most elementary phase of NLP. It deals with the word formation. In this phase, individual words are analyzed according to their components called "morphemes". In addition, non-word taken such as punctuation, etc. are separated from words. Morpheme is basic grammatical building block that makes words.

The study of word structure is refereed to as morphology. In natural language processing, it is done in morphological analysis. The task of breaking a word into its morphemes is called morphological parsing. A morpheme is defined as minimal meaningful unit in a language, which cannot be further broken into smaller units.

Example: word fox consists a single morpheme, as it cannot be further resolved into smaller units. Whereas word cats consists two morphemes, the morpheme "cat" and morpheme "s" indicating plurality.

Here we defined the term meaningful. Though cat can be broken in "c" and "at", but these do not relate with word "cat" in any sense. Thus word "cat" will be dealt with as minimum meaningful unit.

Morphemes are traditionally divided into two types

(i) "free morphemes", that are able to act as words in isolation (e.g., "thing", "permanent", "local") (ii) "bound morphemes", that can operate only as part of other words (e.g., "is" 'ing' etc) The morpheme, which forms the center part of the world, is also called "stem". In English, a word can be made up of one or more morphemes, e.g., word - thing -> stem "think" word - localize -> stem "local", suffix "ize" word - denationalize -> prefix "de", stem "nation", suffix "al", "ize" The computational tool to perform morphological parsing is finite state transducer. A transducer performs it by mapping between the two sets of symbols, and a finite state transducer does it with finite automaton. A transducer normally consists of four parts: recognizer, generator, translator, and relator. The output of the transducer becomes a set of morphemes.

Lexical Analysis In this phase of natural language analysis, validity of words according to lexicon is checked. Lexicon stands for dictionary. It is a collection of all possible valid words of language along with their meaning.

In NLP, the first stage of processing input text is to scan each word in sentence and compute (or look-up) all the relevant linguistic information about that word. The lexicon provides the necessary rules and data for carrying out the first stage analysis.

The details of words, like their type (noun, verb and adverb, and other details of nouns and verb, etc.) are checked.

Lexical analysis is dividing the whole chunk of text into paragraphs, sentences, and words.

Syntactic Analysis Syntax refers to the study of formal relationships between words of sentences. In this phase the validity of a sentence according to grammar rules is checked. To perform the syntactic analysis, the knowledge of grammar and parsing is required. Grammar is formal specification of rules allowable in the language, and parsing is a method of analyzing a sentence to determine its structure according to grammar. The most common grammar used for syntactic analysis for natural languages are context free grammar (CFG) also called phase structure grammar and definite clause grammar. These grammars are described in detail in a separate actions.

Syntactic analysis is done using parsing. Two basic parsing techniques are: top-down parsing and bottom-up parsing.

Semantic Analysis In linguistics, semantic analysis is the process of relating syntactic structures, from the levels of phrases, clauses, sentences and paragraphs to the level of the writing as a whole, to their language-independent meanings. It also involves removing features specific to particular linguistic and cultural contexts, to the extent that such a project is possible.

The elements of idiom and figurative speech, being cultural, are often also converted into relatively invariant meanings in semantic analysis. Semantics, although related to pragmatics, is distinct in that the former deals with word or sentence choice in any given context, while pragmatics considers the unique or particular meaning derived from context or tone. To reiterate in different terms, semantics is about universally coded meaning, and pragmatics the meaning encoded in words that is then interpreted by an audience

Discourse Analysis The meaning of any sentence depends upon the meaning of the sentence just before it. In addition, it also brings about the meaning of immediately succeeding sentence.

Topics of discourse analysis include:

The various levels or dimensions of discourse, such as sounds, gestures, syntrax, the lexicon, style, rhetoric, meanings, speech acts, moves, strategies, turns, and other aspects of interaction Genres of discourse (various types of discourse in politics, the media, education, science, business, etc.) The relations between text (discourse) and context The relations between discourse and power The relations between discourse and interaction The relations between discourse and cognition and memory Pragmatic Analysis During this, what was said is reinterpreted on what it actually meant. It involves deriving those aspects of language which require real world knowledge.

Sentiment Analysis MetaMind, @RichardSocher

Named Entity Recognition KDD 2015 Tutorial: Automatic Entity Recognition and Typing from Massive Text Corpora - A Phrase and Network Mining Approach

Relationship Extraction AlchemyAPI

29.3 Natural Language Processing Applications

Information Retrieval (IR) Information retrieval (IR) is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on metadata or on full-text (or other content-based) indexing.

Information Extraction (IE) Information extraction (IE) is the task of automatically extracting structured information from unstructured and/or semi-structured machine-readable documents. In most of the cases this activity concerns processing human language texts by means of natural language processing (NLP).

Machine Translation Machine translation, sometimes referred to by the abbreviation MT (not to be confused with computer-aided translation, machine-aided human translation (MAHT) or interactive translation) is a sub-field of computational linguistics that investigates the use of software to translate text or speech from one language to another.

Question Answering (QA) Question answering (QA) is a computer science discipline within the fields of information retrieval and natural language processing (NLP), which is concerned with building systems that automatically answer questions posed by humans in a natural language.

29.4 Spelling Correction

For instance, we may wish to retrieve documents containing the term carrot when the user types the query carot. Google reports (http://www.google.com/jobs/britney.html) that the following are all treated as misspellings of the query britney spears: britian spears, britney's spears, brandy spears and prittany spears

We look at two steps to solving this problem: the first based on edit distance and the second based on k-gram overlap. Before getting into the algorithmic details of these methods, we first review how search engines provide spell-correction as part of a user experience.

Implementing spelling correction There are two basic principles underlying most spelling correction algorithms.

Of various alternative correct spellings for a mis-spelled query, choose the nearest one. This demands that we have a notion of nearness or proximity between a pair of queries. When two correctly spelled queries are tied (or nearly tied), select the one that is more common. For instance, grunt and grant both seem equally plausible as corrections for grnt. Then, the algorithm should choose the more common of grunt and grant as the correction. The simplest notion of more common is to consider the number of occurrences of the term in the collection; thus if grunt occurs more often than grant, it would be the chosen correction. A different notion of more common is employed in many search engines, especially on the web. The idea is to use the correction that is most common among queries typed in by other users. The idea here is that if grunt is typed as a query more often than grant, then it is more likely that the user who typed grnt intended to type the query grunt. Corpus Birkbeck spelling error corpus

References How to Write a Spelling Corrector. Peter Norvig. 2007 Statistical Natural Language Processing in Python. Peter Norvig. 2007 Spelling correction. Introduction to Information Retrieval. 2008

29.5 Word Vectors

Discrete Representation Use a taxonomy like WordNet that has hypernyms (is-a) relationships

from nltk.corpus import wordnet as wn panda = wn.synset("panda.n.01") hyper = lambda s: s.hypernyms() list(panda.closure(hyper)) [Synset('procyonid.n.01'), Synset('carnivore.n.01'), Synset('placental.n.01'), Synset('mammal.n.01'), Synset('vertebrate.n.01'), Synset('chordate.n.01'), Synset('animal.n.01'), Synset('organism.n.01'), Synset('living$_thing.n.01'), Synset('w$

Great as resource but missing nuances, e.g. synonyms: adept, expert, good, practiced, proficient, skillful? Missing new words (impossible to keep up to date): wicked, badass, nifty, crack, ace, wizard, genius, ninjia Subjective Requires human labor to create and adapt Hard to compute accurate word similarity Word2Vec Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located in close proximity to one another in the space.

Word2vec was created by a team of researchers led by Tomas Mikolov at Google. The algorithm has been subsequently analysed and explained by other researchers. Embedding vectors created using the Word2vec algorithm have many advantages compared to earlier algorithms like Latent Semantic Analysis.

Main Idea of Word2Vec

Instead of capturing cooccurrence counts directly,, Predict surrounding words of every word Both are quite similar, see "Glove: Global Vectors for Word Representation" by Pennington et at. (2014) and Levy and Goldberg (2014)... more later. Faster and can easily incorporate a new sentence/document or add a word to the vocabulary. Detail of Word2Vec

Predict surrounding words in a window of length m of every word. Objective function: Maximize the log probability of any context word given the current cenetr word: J()=1Tt=1Tmjm,j0log p(wt+j|wt) J()=1Tt=1Tmjm,j0log p(wt+j|wt) where represents all variables we optimize

Predict surrounding words in a window of length m of every word For p(wt+j|wt)p(wt+j|wt) the simplest first formulation is p(o|c)=exp(uTovc)Ww=1exp(uTwvc) p(o|c)=exp(uoTvc)w=1Wexp(uwTvc) where o is the outside (or output) word id, c is the center word id, u and v are "center" and "outside" vectors of o and c

Every word has two vectors! This is essentially "dynamic" logistic regression Linear Relationships in word2vec

These representations are very good at encoding dimensions of similarity!

Analogies testing dimensions of similarity can be solved quite well just by doing vector subtraction in the embedding space Syntactically

xapplexapplesxcarxcarsxfamilyxfamiliesxapplexapplesxcarxcarsxfamilyxfamilies Similarly for verb and adjective morphological forms Semantically (Semeval 2012 task 2)

xshirtxclothingxchairxfurniturexshirtxclothingxchairxfurniture xkingxmanxqueenxwomanxkingxmanxqueenxwoman GloVe Project

Highlights Training Model Overview GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

Pre-trained Model fastText

Pre-trained word vectors for 294 languages, trained on Wikipedia using fastText. These vectors in dimension 300 were obtained using the skip-gram model described in Bojanowski et al. (2016) with default parameters.

glove

Pre-trained word vectors. This data is made available under the Public Domain Dedication and License v1.0 whose full text can be found at: http://www.opendatacommons.org/licenses

Language: English

Wikipedia 2014 + Gigaword 5 (6B tokens, 400K vocab, uncased, 50d, 100d, 200d, 300d vectors, 822 MB download): glove.6B.zip Common Crawl (42B tokens, 1.9M vocab, uncased, 300d vectors, 1.75 GB download): glove.42B.300d.zip Common Crawl (840B tokens, 2.2M vocab, cased, 300d vectors, 2.03 GB download): glove.840B.300d.zip Twitter (2B tweets, 27B tokens, 1.2M vocab, uncased, 25d, 50d, 100d, 200d vectors, 1.42 GB download): glove.twitter.27B.zip word2vec-GoogleNews-vectors

Language: English

Pre-trained Google News corpus (3 billion running words) word vector model (3 million 300-dimension English word vectors).

Word Analogies Test for linear relationships, examined by Mikolov et at. (2014)

Suggested Readings Simple Word Vector representations: word2vec, GloVe. cs224d.stanford.edu. Last Accessed: 2017-02-01. FastText and Gensim word embeddings. rare-technologies.com. Last Accessed: 2016-08-31. Distributed Representations of Words and Phrases and their Compositionality. papers.nips.cc. Last Accessed: 2013-12-05. Efficient Estimation of Word Representations in Vector Space. arxiv.org. Last Accessed: 2013-01-16

29.6 Conditional Random Fields in Name Entity Recognition

In this tutorial, I will write about how to using CRF++ to train your data for name entity recognition task.

Environment:

Ubuntu 14.04 Install CRF++ Download CRF++-0.58.tar.gz

Extact CRF++-0.58.tar.gz file

Navigate to the location of extracted folder through

Install CRF++ from source

./configure make sudo make install ldconfig Congratulations! CRF++ is install

crf$_learnTrainingCRFTotrainaCRFusingCRF++, youneed2things:$

A template file: where you define features to be considered for training A training data file: where you have data in CoNLL format crf$_learn-ttemplate_filetrain_data_filemodel$

crf$_learn-ttemplatetrain.txtmodelAbinaryofmodelisproduce.$

To test this model, on a testing data

crf$_test-mmodeltestfile > output.txt$

crf$_test-mmodeltest.txt > output.txtReferencesConditionalRandomFields:$ $InstallingCRF++onUbuntuConditionalRandomFieldsTrainingandTestingusingCRF+$
$+$

29.7 Entity Linking

In natural language processing, entity linking, named entity linking (NEL), named entity disambiguation (NED), named entity recognition and disambiguation (NERD) or named entity normalization (NEN) is the task of determining the identity of entities mentioned in text. More precise, it is the task of linking entity mentions to entries in a knowledge base (e.g., DBpedia, Wikipedia)

Entity linking requires a knowledge base containing the entities to which entity mentions can be linked. A popular choice for entity linking on open domain text are knowledge-bases based on Wikipedia, in which each page is regarded as a named entity. NED using Wikipedia entities has been also called wikification (see Wikify! an early entity linking system]). A knowledge base may also be induced automatically from training text or manually built.

NED is different from named entity recognition (NER) in that NER identifies the occurrence or mention of a named entity in text but it does not identify which specific entity it is

Examples Example 1:

For example, given the sentence "Paris is the capital of France", the idea is to determine that "Paris" refers to the city of Paris and not to Paris Hilton or any other entity that could be referred as "Paris".

Example 2:

Give the sentence "In Second Debate, Donald Trump and HIllary Clinton Spar in Bitter, Personal Terms", the idea is to determine that "Donald Trump" refer to an American politician, and "Hillary Clinton" refer to 67th United States Secretary of State from 2009 to 2013.

Architecture

Mention detection: Identification of text snippets that can potentially be linked to entities Candidate selection: Generating a set of candidate entities for each mention Disambiguation: Selecting a single entity (or none) for each mention, based on the context Mention detection

Goal: Detect all "linkable" phrases

Challenges:

Recall oriented: Do not miss any entity that should be link Find entity name variants (e.g. "jlo" is name variant of [Jennifer Lopez]) Filter out inappropriate ones (e.g. "new york" matches >2k different entities) COMMON APPROACH Build a dictionary of entity surface forms entities with all names variants Check all document n-grams against the dictionary the value of n is set typically between 6 and 8 Filter out undesired entities Can be done here or later in the pipeline Examples

Candidate Selection

Goal: Narrow down the space of disambiguation possibilities

Balances between precision and recall (effectiveness vs. efficiency)

Often approached as ranking problem: keeping only candidates above a score/rank threshold for downstream processing.

COMMONNESS Perform the ranking of candidate entities based on their overall popularity, i.e., "most command sense"

Examples

Commonness can be pre-computed and stored in the entity surface form dictionary. Follows a power law with a long tail of extremely unlikely senses;

entities at the tail end of distribution can be safely discarded (e.g., 0.001 is sensible threshold)

Disambiguation

Baseline approach: most common sense

Consider additional types of evidence: prior importance of entities and mentions, contextual similarity between the text surrounding the mention and the candidate entity, coherence among all entity linking decisions in the document.

Combine these signals: using supervised learning or graph-based approaches

Optionally perform pruning: reject low confidence or semantically meaning less annotations.

References “Entity Linking”. wikipedia “Entity Linking”. Krisztian Balog, University of Stavanger, 10th Russian Summer School in Information Retrieval. 2016 “An End-to-End Entity Linking Approach for Tweets”. Ikuya Yamada, Hideaki Takeda, Yoshiyasu Takefuji. 2015

Chương 30

Nhận dạng tiếng nói

Trong hệ thống nhận dạng tiếng nói, tín hiệu âm thanh được thu thập như những mẫu phù hợp cho quá trình xử lý của máy tính và được đưa vào quá trình nhận diện. Đầu ra của hệ thống là một câu phụ đề của câu nói.

Nhận dạng tiếng nói là một nhiệm vụ phức tạp và hệ thống tốt nhất trong nhận dạng tiếng nói rất phức tạp. Có rất nhiều cách tiếp cận cho mỗi thành phần. Trong phần này, người viết chỉ muốn đưa ra một cái nhìn tổng thể về nhận dạng tiếng nói, các khó khăn chính, các thành phần cơ bản, chức năng và tương tác của chúng trong một hệ thống nhận dạng tiếng nói.

Các thành phần của hệ thống nhận dạng tiếng nói

`![](http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/img1.gif)`

Trong bước thứ nhất, trích rút thông tin *Feature Extraction*, các mẫu tín hiệu được tham số hóa. Mục tiêu là trích xuất ra một tập các tham số (đặc trưng) từ tín hiệu có nhiều thông tin hữu ích nhất cho quá trình phân loại. Các đặc trưng chính được trích xuất với điều kiện *thích nghi* với các sự thay đổi của âm thanh và *nhạy cảm* với các nội dung ngôn ngữ.

Trong module phân loại, các vector đặc trưng được ánh xạ với các pattern, được gọi là *mô hình âm học* (acoustic model). Mô hình học thường là HMM được train với toàn bộ từ, hay âm như là một đơn vị ngôn ngữ.

Từ điển phát âm (pronunciation dictionary) định nghĩa cách kết hợp âm cho các ký tự. Nó có thể chứa cách phát âm khác nhau cho cùng một từ. Bảng 1 hiển thị chính xác một từ điển. Từ (graphme) ở cột bên trái ứng với cách phát âm (các âm) ở cột bên phải (các ký tự âm trong bảng được dùng phổ biến đối với tiếng Anh)

| word | pronunciation | |————|—————————| | INCREASE | ih n k r iy s | | INCREASED | ih n k r iy s t | | INCREASES | ih n k r iy s ah z | | INCREASING | ih n k r iy s ih ng | | INCREASINGLY | ih n k r iy s ih ng l iy | | INCREDIBLE | ih n k r eh d ah b ah l |

Mô hình ngôn ngữ (language model) chứa các thông tin về cú pháp. Mục tiêu để dự đoán khả năng một từ xuất hiện sau các từ khác trong một ngôn ngữ. Nói cách khác, xác xuất để một từ k xảy ra sau khi k-1 từ sau đó được định nghĩa bởi $latex P(w_k|w_{k-1}, w_{k-2}, ..., w1)$

Mô hình hóa sub-word với HMMs

Trong các hệ thống ASR, HMMs được dùng để biểu diễn các đơn vị dưới từ (ví dụ như âm). Với ngôn ngữ, thông thường có 40 âm. Số lượng âm phụ thuộc vào từ điển được sử dụng. Số lượng âm phụ thuộc vào từ điển được sử dụng. Mô hình từ có thể được xây dựng bằng cách kết hợp các mô hình dưới từ.

Trong thực tế, khi nhận dạng một âm phụ thuộc rất nhiều vào các âm bên cạnh. Do đó, mô hình âm phụ thuộc ngữ cảnh (*context dependence*) được sử dụng rất phổ biến. Mô hình *biphone* chú ý đến âm bên trái hoặc âm bên phải, mô hình *triphone* chú ý đến cả hai phía, với một âm, các mô hình khác nhau được sử dụng trong ngữ cảnh khác nhau. Hình dưới thể hiện các mô hình monophone, biphone và triphone của từ *bat* (b ae t)

`![](http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/img10.gif)`

Quá trình huấn luyện

Huấn luyện các mô hình monophone

Một mô hình monophone là một mô hình âm học, trong đó không chứa thông tin ngữ cảnh về các âm trước và sau. Nó được sử dụng như thành phần cơ bản cho các mô hình triphone - mô hình sử dụng những thông tin về ngữ cảnh.

Việc huấn luyện sử dụng framework Gaussian Mixture Model/Hidden Markov Model.

Dóng hàng âm thanh trong mô hình âm học

Các tham số trong mô hình âm học được tính toán trong quá trình huấn luyện; tuy nhiên, quá trình này có thể được tối ưu hóa bởi việc lặp lại quá trình huấn luyện và dòng hàng. Còn lại là huấn luyện Viterbi (liên quan đến phương pháp này, nhưng dùng nhiều khối lượng tính toán hơn là thuật toán Forward-Backward và Expectation Maximization). Bằng cách dóng hàng âm thanh - phụ đề với mô hình âm học hiện tại, các thuật toán huấn luyện có thể sử dụng kết quả này để cải thiện và hiệu chỉnh tham số của mô hình. Do đó, mỗi quá trình huấn luyện sẽ theo bởi một bước dóng hàng trong đó âm thanh và văn bản được dóng hàng lại.

Huấn luyện các mô hình triphone

Trong khi các mô hình monophone đơn giản biểu diễn các đặc trưng âm thanh như một đơn âm, trong khi các âm vị sẽ thay đổi đáng kể phụ thuộc vào ngữ cảnh. Mô hình triphone thể hiện một âm trong ngữ cảnh với hai âm bên cạnh.

Đến đây, một vấn đề là không phải tất cả các đơn vị triphone được thể hiện trong dữ liệu huấn luyên. Có tất cả (of phonemes)$^3 triphone, nhngchcmttpthcstntitrongdliu.Hnna, cc\emptyset nvxyr$

Dóng hàng các mô hình âm học và huấn luyện lại các mô hình triphone

Lặp lại các bước dòng hàng âm thanh và huấn luyện các mô hình triphone với các thuật toán huấn luyện để hiệu chỉnh mô hình. Các phương pháp phổ biến là delta+delta-delta, LDA-MLLT và SAT. Các giải thuật dóng hàng bao gồm dóng hàng cho từng người nói và FMLLR.

Các thuật toán huấn luyện

Huấn luyện delta+delta-delta tính các đặc trưng delta và double-delta, hay các hệ số động, để thêm vào các đặc trưng MFCC. Delta và delta-delta là các đặc trưng số học, tính các đạo hàm bậc 1 và 2 của tín hiệu. Do đó, phép tính toán này thường được thực hiện trên một window của các đặc trưng vector. Trong khi một window của hai đặc trưng vector có thể hiệu quả, nó là các xấp xỉ thô (giống như delta-diffrence là một xấp xỉ thô của đạo hàm). Đặc trưng delta được tính toán trong các window của các đặc trưng cơ bản, trong khi delta-delta được tính toán trong các window của đặc trựng delta.

LDA-MLLT viết tắt của Linear Discriminant Analysis - Maximum Likelihood Linear Transform. Linear Discriminant Analysis lấy các đặc trưng vector và xây dựng các trạng thái HMM, nhưng giảm thiểu không gian vector. Maximum Likelihood Linear Transfrom lấy các đặc trưng được giảm từ LDA, và thực hiện các biến đổi đối với từng người nói. MLLT sau đó thực hiện một bước chuẩn hóa, để giảm sự khác biệt giữa các người nói.

SAT viết tắt của Speaker Adaptive Training. SAT cũng thực hiện các chuẩn hóa đối với người nói bằng cách thực hiện biến đổi trên mỗi người nói. Kết quả của quá trình này đồng nhất và chuẩn hóa hơn, cho phép mô hình có thể sử dụng những tham số này để giảm thiểu sự biến đổi của âm, đối với từng người nói hoặc môi trường thu.

Các thuật toán dóng hàng

Thuật toán dòng hàng luôn luôn cố định, trong đó các kịch bản chấp nhận các loại đầu vào âm học khác nhau. Dòng hàng đối với từng người nói, sẽ tách biệt thông tin giữa các người nói trong quá trình dóng hàng.

fMLLR viết tắt của Feature Space Maximum Likelihood Linear Regression. Sau quá trình huấn luyện SAT, các mô hình âm học không huấn luyện trên các đặc trưng ban đầu, mà đối với các đặc trưng chuẩn hóa theo người nói. Với quá trình dóng hàng, xóa bỏ sự khác biệt giữa người nói (bằng cách nghịch đạo ma trận fMLLR), sau đó loại bỏ nó khỏi mô hình *bằng cách nhân ma trận nghịch đảo với đặc trưng vector). Mô hình âm học quasi-speaker-independent có thể sử dụng trong quá trình dóng hàng.

Dóng hàng (Forced Alignment)

Hệ thống nhận dạng tiếng nói sử dụng một máy tìm kiếm bên cạnh mô hình âm học và ngôn ngữ trong đó chứa tập các từ, âm và tập dữ liệu để đối chiếu với dữ liệu âm thanh cho câu nói. Máy tìm kiếm này sử dụng các đặc trưng được trích xuất bởi dữ liệu âm thanh để xác định sự xuất hiện của từ, âm và đưa ra kết quả.

![](https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/

Quá trình dòng hàng cũng tương tự như vậy, nhưng khác ở một điểm quan trong. Thay vì đưa vào tập các từ có thể để tìm kiếm, máy tìm kiếm đưa vào đoạn phụ đề tương ứng với câu nói. Hệ thống sau đó dóng hàng dữ liệu văn bản với dữ liệu âm thanh, xác định đoạn nào trong âm thanh tương ứng với từ cụ thể nào trong dữ liệu văn bản.

![](https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/

Dóng hàng có thể sử dụng để dóng âm trong dữ liệu với bản với dữ liệu âm thanh, giống như hình dưới đây, các âm được xác định trong từng đoạn của âm thanh.

![](https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/

Hidden Markov Model

Hidden Markov Model (HMM) là mô hình trọng số với các trọng số ở cung, chỉ khả năng xuất hiện của cung.

Một trong những ứng dụng của HMM, là phán đoán chuỗi các trạng thái thay đổi, dựa vào chuỗi các quan sát

Các trọng số trong trạng thái gọi là observation likelihood, các trọng số ở cung gọi là transition likelihood.

Sau đây là một ví dụ:

* Thời tiết trong một ngày có thể là NÓNG hoặc LẠNH * Khi trời NÓNG, 20* Khi trời NÓNG, 30* `![https://qph.ec.quoracdn.net/main-qimg-a6744f9e17e59f3729d6fef02d54391b.webp]( qimg-a6744f9e17e59f3729d6fef02d54391b.webp)`

Giờ, giả sử chung ta quan sát trong 3 ngày, bạn dùng 1,2,3 viên đá. Thời tiết có khả năng diễn ra như thế nào?

Đến đây chúng ta dùng thuật toán Viterbi. Về cơ bản, nó là dynamic programming với hai chiều [state, position$_in_sequence$]

Gọi S là trạng thái hiện tại HOT, COLD trong quan sát i, S' là trạng thái trước đó, và A là lượng đá tiêu thụ 1, 2, 3 trong quan sát i

$latexViterbi[S, i] = Viterbi[S', i-1] * p(S|S') * p(A|S)$

$latexV[S, i] = V[S', i-1] *$ transition$_likelihood * observation_likelihood$

HMM được sử dụng trong các hệ thống thỏa mãn

1. Có hữu hạn các trạng thái nội tại (internal state), là nguyên nhân của các sự kiện (external events) (các quan sát) 2. Trạng thái nội tại không quan sát được (hidden) 3. Trạng thái hiện tại chỉ phụ thuộc vào trạng thái trước đó (qúa trình Markov)

Wow! George nhanh chóng liên hệ vụ của anh đấy với mô hình HMM. George nhận ra rằng CCTV footage từ các cặp có thể coi như là chuỗi quan sát được, anh đấy có thể dùng mô hình và sử dụng nó để phát hiện hành vị ẩn mà Bob và William hoạt động.

3 vấn đề cơ bản được Jack Ferguson giới thiệu trong những năm 1960

Vấn đề 1 (Likelihood): Cho một HMM $\lambda = (A, B)$ và một chuỗi quan sát O, xác định likelihood $P(O|\lambda)$

Vấn đề 2 (Decoding): Cho một chuỗi quan sát O, và một HMM $\lambda = (A, B)$, xác định chuỗi ẩn Q tốt nhất

Vấn đề 3 (Learning): Cho một chuỗi quan sát O, một tập các trạng thái trong HMM, học các tham số A và B

Likelihood Computation

Vấn đề đầu tiên là tính xác suất xảy ra của một chuỗi quan sát. Ví dụ, trong bài toán ăn đá ở hình 9.3, xác suất xảy ra chuỗi *3 1 3* là bao nhiêu?

Tính toán Likelihood: Chuỗi một HMM $\lambda = (A, B)$, và mỗi chuỗi quan sát O, xác định likelihood $P(O|\lambda)$

Thuật toán Forward, nếu sử dụng Bayes rule, để tính likelihood, cần khối lượng tính toán N^T với N là số trạng thái có thể có và T là chiều dài chuỗi quan sát. Ví dụ trong bài toán gán nhãn có N=10 nhãn, chiều dài của chuỗi trung bình là 28, thì cần 10^{28} bước tính toán. Một giải thuật với hiệu quả $O(N^2T)$ được đề xuất với tên gọi **forward algorithm**

Tài liệu tham khảo

* http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/node1.html * https://www.isip.piconepress.com

$http://www.igi.tugraz.at/lehre/CI/SS08/tutorials/ASR/node1.html*https://www.isip.piconepress.com/projects/speech/software/tutorials/production/fundamentals/v1.0/section$

$https://www.quora.com/What-is-a-simple-explanation-of-the-Hidden-Markov-Model-algorithm$

Chương 31

Phân loại văn bản

<h3>Naive Bayes Classifier</h3> Tham khảo thư viện <a href="http://scikit-learn.org/stable/modules/naive$_{b}ayes.html”target = ”_{b}lank”rel = ”noopener” >$
$Scikit - learn < /a >$

Xét bài toán classification với C classes 1,2,. . . ,C. Tính xác suất để 1 điểm dữ liệu rơi vào class C ta có công thức: $latexP(\frac{c}{x})$. Tức tính xác suất để đầu ra là class C biết rằng đầu vào là vector x. Việc xác định class của điểm dữ liệu đó bằng cách chọn ra class có xác suất cao nhất:<p style="text-align:center;"> c = argmax($latexP(\frac{c}{x})$) với c 1,. . . ,C</p> Sử dụng quy tắc Bayes:<p style="text-align:center;"> c = argmax($latexP(\frac{c}{x})$) = argmax($latexP(\frac{P(\frac{c}{x})P(x)}{P(x)})$ = argmax($latexP(\frac{P(\frac{c}{x})}{P(c)})$)</p>

<h4>Các phân phối thường dùng</h4> Gaussian Naive Bayes Mô hình này được sử dụng chủ yếu trong loại dữ liệu mà các thành phần là các biến liên tục. Multinomial Naive Bayes Mô hình này chủ yếu được sử dụng trong phân loại văn bản mà feature vectors được tính bằng Bags of Words. Lúc này, mỗi văn bản được biểu diễn bởi một vector có độ dài d chính là số từ trong từ điển. Giá trị của thành phần thứ i trong mỗi vector chính là số lần từ thứ i xuất hiện trong văn bản đó. Khi đó, $latexP(\frac{x_i}{c})$ tỉ lệ với tần suất từ thứ i xuất hiện trong các văn bản của class c:<p style="text-align:center;"> $latexP(\frac{x_i}{c}) = latex\frac{Nx_i}{Nc}$</p> <p style="padding-left:70px;">Trong đó:</p> <p style="padding-left:90px;">$latexNx_i$ là tổng số lần từ thứ i xuất hiện trong các văn bản của class c, nó được tính là tổng của tất cả các thành phần thứ i của các feature vectors ứng với class c.</p> <p style="padding-left:90px;">$latexNc$ là tổng số từ (kể cả lặp) xuất hiện trong class c. Hay bằng tổng độ dài của toàn bộ các văn bản thuộc vào class c.</p> Nếu có một từ mới chưa bao giờ xuất hiện trong class c thì biểu thức trên sẽ bằng 0, điều này dẫn đến vế phải của c bằng 0. Bernoulli Naive Bayes Mô hình này được áp dụng cho các loại dữ liệu mà mỗi thành phần là một giá trị binary. Ví dụ: cũng với loại văn bản nhưng thay vì đếm tổng số lần xuất hiện của 1 từ trong văn bản, ta chỉ cần quan tâm từ đó có xuất hiện hay không. Khi đó: $latexP(\frac{x_i}{c}) = latexP(\frac{i}{c})x_i + (1\ \ latexP(\frac{i}{c})(1\ \ latexx_i))$ Với $latexP(\frac{i}{c})$ là xác suất từ thứ i xuất hiện trong các văn bản của class c.

Chương 32

Pytorch

Bí kíp luyện công

(cập nhật 08/12/2017): cảm giác [talk](http://videolectures.net/deeplearning2017$_c hintala_t orch/)caanhSo$

Sau khi nghe bài này thì hâm mộ luôn anh Soumith Chintala, tìm loạt bài anh trình bày luôn

* [PyTorch: Fast Differentiable Dynamic Graphs in Python with a Tensor JIT](https://www.youtube.com/watch?v=DBVLcgq2Eg0amp;t=2s), Strange Loop Sep 2017 * [Keynote: PyTorch: Framework for fast, dynamic deep learning and scientific computing](https://www.youtube.com/watch?v=LAMwEJZqesUamp;t=66s), EuroSciPy Aug 2017

So sánh giữa Tensorflow và Pytorch?

Có 2 điều cần phải nói khi mọi người luôn luôn so sánh giữa Tensorflow và Pytorch. (1) Tensorflow khiến mọi người "không thoải mái" (2) Pytorch thực sự là một đối thủ trên bàn cân. Một trong những câu trả lời hay nhất mình tìm được là của anh Hieu Pham (Google Brain) [trả lời trên quora (25/11/2017)](https://www.quora.com/What-are-your-reviews-between-PyTorch-and-TensorFlow/answer/Hieu-Pham-20?srid=5O2u). Điều quan trọng nhất trong câu trả lời này là *"Dùng Pytorch rất sướng cho nghiên cứu, nhưng scale lên mức business thì Tensorflow là lựa chọn tốt hơn"*

Behind The Scene

(15/11/2017) Hôm nay bắt đầu thử nghiệm pytorch với project thần thánh classification sử dụng cnn https://github.com/Shawn1993/cnn-text-classification-pytorch

Cảm giác đầu tiên là make it run khá đơn giản

‵‵‵ conda create -n test-torch python=3.5 pip install http://download.pytorch.org/whl/cu80/torch-0.2.0.post3-cp35-cp35m-manylinux1$_x86_64.whlpipinstalltorchvisionpipinstalltorchtext$‵‵‵

Thế là ‘main.py‘ chạy! Hay thật. Còn phải vọc để bạn này chạy với CUDA nữa.

Cài đặt CUDA trong ubuntu 16.04

Kiểm tra VGA

‵‵‵ $lspci|grepVGA01:00.0VGAcompatiblecontroller:NVIDIACorporationGM204[GeForceGTX980]($

Kiểm tra CUDA đã cài đặt trong Ubuntu [1]

‵‵‵ $nvcc--versionnvcc:NVIDIA(R)CudacompilerdriverCopyright(c)2005-2016NVIDIACorporationBuiltonSun_Sep_{42}2:14:01_CDT_2016Cudacompilationtools,release8.0,V8.0.44$‵‵‵

Kiểm tra pytorch chạy với cuda ‘test$_cuda.py$‘

‵‵‵python import torch print("Cuda:", torch.cuda.is$_available()$)‵‵‵

"' $pythontest_cuda.pyCUDA : True$"'

Chỉ cần cài đặt thành công CUDA là pytorch tự work luôn. Ngon thật!

Ngày X

Chẳng hiểu sao update system kiểu nào mà hôm nay lại không sử dụng được CUDA 'torch.cuda.is$_a vailable() = False'.Saukhidnglnh'torch.Tensor().cuda()'thgpli$

"' AssertionError: The NVIDIA driver on your system is too old (found version 8000). Please update your GPU driver by downloading and installing a new version from the URL: http://www.nvidia.com/Download/index.aspx Alternatively, go to: https://pytorch.org/binaries to install a PyTorch version that has been compiled with your version of the CUDA driver. "'

Kiểm tra lại thì mình đang dùng nvidia-361, làm thử theo [link này](http://www.linuxandubuntu.com/hom to-install-latest-nvidia-drivers-in-linux) để update NVIDIA, chưa biết kết quả ra sao?

May quá, sau khi update lên nvida-387 là ok. Haha

Ngày 2

Hôm qua đã bắt đầu implement một nn với pytorch rồi. Hướng dẫn ở [Deep Learning with PyTorch: A 60 Minute Blitz](http://pytorch.org/tutorials/beginner/deep$_learning_60min_blitz.h$

Hướng dẫn implement các mạng neural với pytorch rất hay tại [PyTorch-Tutorial](https://github.com/MorvanZhou/PyTorch-Tutorial)

(lượm lặt) Trang này [Awesome-pytorch-list](https://github.com/bharathgs/Awesome-pytorch-list) chứa rất nhiều link hay về pytorch như tập hợp các thư viện liên quan, các hướng dẫn và ví dụ sau đó là các cài đặt của các paper sử dụng pytorch.

(lượm lặt) Loạt video hướng dẫn pytorch [PyTorchZeroToAll](https://www.youtube.com/watch?v=SKq-pmkekTkamp;list=PLlMkM4tgfjnJ3I-dbhO9JTw7gNty6o$_2 m)catcgiSungKimtrnyoutube.$

Bước tiếp theo là visualize loss và graph trong tensorboard, sử dụng [tensorboard$_logger](https://github.com/TeamHG-Memex/tensorboard_logger)khhay.$

"' pip install tensorboard$_loggerpipinstalltensorboard$"'

Chạy tensorboard server

"' tensorboard –log-dir=runs "'

Ngày 3: Vấn đề kỹ thuật

Hôm qua cố gắng implement một phần thuật toán CNN cho bài toán phân lớp văn bản. Vấn đề đầu tiên là biểu diễn sentence thế nào. Cảm giác load word vector vào khá chậm. Mà thằng tách từ của underthesea cũng chậm kinh khủng.

Một vài link tham khảo về bài toán CNN: [Implementing a CNN for Text Classification in TensorFlow](http://www.wildml.com/2015/12/implementing-a-cnn-for-text-classification-in-tensorflow/), [Text classification using CNN : Example](https://agarnitin86.git classification-cnn)

[1] : $https://askubuntu.com/questions/799184/how-can-i-install-cuda-on-ubuntu-16-04$

Chương 33

Big Data

View online http://magizbox.com/training/bigdata/site/

Big Data QA 1. What is "Big Data"? 1 https://www.youtube.com/watch?v=TzxmjbL-i4Y

2. How big is big data? 2
3. How much data is "Big Data"? 3
4. What are characteristics of "Big Data"? 4
5. What is big data ecosystem? 5
6. What is big data landscape 6
7. What are benefits of big data? 7

https://www.youtube.com/watch?v=TzxmjbL-i4Y
http://scoop.intel.com/what-happens-in-an-internet-minute/
http://www.quora.com/How-much-data-is-Big-Data
https://en.wikipedia.org/wiki/Big$_{d}ataCharacteristics$
http://www.clearpeaks.com/blog/big-data/big-data-ecosystem-spark-and-tableau

https://vladimerbotsvadze.wordpress.com/2015/01/28/the-big-data-landscape-technology-businessintelligence-analytics/
http://blog.galaxyweblinks.com/big-data-with-bigger-benefits/

33.1 Distribution Storage

33.1.1 HDFS

The Hadoop Distributed File System (HDFS) — a subproject of the Apache Hadoop project—is a distributed, highly fault-tolerant file system designed to run on low-cost commodity hardware. HDFS provides high-throughput access to application data and is suitable for applications with large data sets. This article explores the primary features of HDFS and provides a high-level view of the HDFS architecture. : sequenceiq/hadoop-docker

Big Data Stack: HDFS, Kibana, ElasticSearch, Neo4J, Apache Spark

33.1.2 HBase

Apache HBase™ is the Hadoop database, a distributed, scalable, big data store. Download Apache HBase™ Click here to download Apache HBase™.

1. When Would I Use Apache HBase? 1 HBase isn't suitable for every problem.

First, make sure you have enough data. If you have hundreds of millions or billions of rows, then HBase is a good candidate. If you only have a few thousand/million rows, then using a traditional RDBMS might be a better choice due to the fact that all of your data might wind up on a single node (or two) and the rest of the cluster may be sitting idle.

Second, make sure you can live without all the extra features that an RDBMS provides (e.g., typed columns, secondary indexes, transactions, advanced query languages, etc.) An application built against an RDBMS cannot be "ported" to HBase by simply changing a JDBC driver, for example. Consider moving from an RDBMS to HBase as a complete redesign as opposed to a port.

Third, make sure you have enough hardware. Even HDFS doesn't do well with anything less than 5 DataNodes (due to things such as HDFS block replication which has a default of 3), plus a NameNode.

HBase can run quite well stand-alone on a laptop - but this should be considered a development configuration only.

2. Features 2 Linear and modular scalability. Strictly consistent reads and writes. Automatic and configurable sharding of tables Automatic failover support between RegionServers. Convenient base classes for backing Hadoop MapReduce jobs with Apache HBase tables. Easy to use Java API for client access. Block cache and Bloom Filters for real-time queries. Query predicate push down via server side Filters Thrift gateway and a REST-ful Web service that supports XML, Protobuf, and binary data encoding options Extensible jruby-based (JIRB) shell Support for exporting metrics via the Hadoop metrics subsystem to files or Ganglia; or via JMX 3. Architecture

HBase Shell [code lang="shell"]

list all table list [/code]

Up Running 1. Download HBase 0.94.27 (HBase 0.98 won't work)

[code lang="shell"] wget https://www.apache.org/dist/hbase/hbase-0.94.27/hbase-0.94.27.tar.gz tar -xzf hbase-0.94.27.tar.gz [/code]

2. Setup 1. edit $HBASE_ROOT/conf/hbase-site.xmlandadd$

[code lang="xml"] hbase.rootdir file:///full/path/to/where/the/data/should/be/stored hbase.cluster.distributed false [/code]

3. Verify Go to http://localhost:60010 to see if HBase is running.

When Should I Use HBase? HBase Config HBase Remote 1. Change /etc/hosts [code] 127.0.0.1 [username] [server$_ip]hbase.io[/code]$

Example

[code] 127.0.0.1 crawler 192.168.0.151 hbase.io [/code]

2. Change hostname [code] hostname hbase.io [/code]

3. Change region servers Edit $HBASE_ROOT/conf/regionservers$

[code] hbase.io [/code]

4. Change $HABSE_ROOT/conf/hbase-site.xml[codelang="xml"title="hbase-site.xml"] <?xml-stylesheettype="text/xsl"href="configuration.xsl"?> hbase.rootdirfile:///home/username/Downloads/hbase/datahbase.cluster.distributedfalsehbase.zookee$
$unsecurehbase.rpc.timeout2592000000[/code]$

Docker HBase 0.94

Image: https://github.com/Banno/docker-hbase-standalone

[code] docker run -d -p 2181:2181 -p 60000:60000 -p 60010:60010 -p 60020:60020 -p 60030:60030 banno/hbase-standalone [/code]

Compose

[code] hbase.vmware: build: ./docker-hbase-standalone/. command: "/opt/hbase/hbase-0.94.15-cdh4.7.0/bin/hbase master start" hostname: hbase.vmware ports: - 2181:2181 - 60000:60000 - 60010:60010 - 60020:60020 - 60030:60030 volumes: - ./docker-hbase-standalone/hbase-0.94.15-cdh4.7.0:/opt/hbase/hbase-0.94.15-cdh4.7.0 - ./data/hbase:/tmp/hbase-root/hbase /code]

33.2 Distribution Computing

33.2.1 Apache Spark

Apache Spark is an open-source cluster computing framework originally developed in the AMPLab at UC Berkeley. In contrast to Hadoop's two-stage disk-based MapReduce paradigm, Spark's in-memory primitives provide performance up to 100 times faster for certain applications. By allowing user programs to load data into a cluster's memory and query it repeatedly, Spark is well-suited to machine learning algorithms. Installation Requirements: Hadoop, YARN

Install Hadoop

Insatll YARN

Install Java

Verification Tutorial From Pandas to Apache Spark's DataFrame

Big Data Stack: HDFS, Kibana, ElasticSearch, Neo4J, Apache Spark

Apache Spark: Tutorials Beginners Guide: Apache Spark Machine Learning with Large Data

Spark and Spark Streaming Unit Testing Recipes for Running Spark Streaming Applications in Production- Databricks

Spark Streaming

Spark and Spark Streaming Unit Testing Recipes for Running Spark Streaming Applications in Production- Databricks

33.3 Components

33.3.1 Ambari

The Apache Ambari project is aimed at making Hadoop management simpler by developing software for provisioning, managing, and monitoring Apache Hadoop clusters. Ambari provides an intuitive, easy-to-use Hadoop management web UI backed by its RESTful APIs.

Ambari enables System Administrators to:

Provision a Hadoop Cluster

Ambari provides a step-by-step wizard for installing Hadoop services across any number of hosts. Ambari handles configuration of Hadoop services for the cluster. Manage a Hadoop Cluster

Ambari provides central management for starting, stopping, and reconfiguring Hadoop services across the entire cluster. Monitor a Hadoop Cluster

Ambari provides a dashboard for monitoring health and status of the Hadoop cluster. Ambari leverages Ambari Metrics System for metrics collection. Ambari leverages Ambari Alert Framework for system alerting and will notify you when your attention is needed (e.g., a node goes down, remaining disk space is low, etc). Ambari enables Application Developers and System Integrators to:

Easily integrate Hadoop provisioning, management, and monitoring capabilities to their own applications with the Ambari REST APIs. Docker

Receipts:

Image: sequenceiq/ambari (git) Multinode cluster with Ambari 1.7.0 1 Get the docker images

[code] docker pull sequenceiq/ambari:1.7.0 [/code]

Get ambari-functions [code] curl -Lo .amb j.mp/docker-ambari-170 . .amb [/code]

Create your cluster – automated

[code] amb-deploy-cluster 3 [/code]

Multinode cluster with Ambari 1.7.0

33.3.2 Kibana

Kibana is an open source data visualization plugin for Elasticsearch. It provides visualization capabilities on top of the content indexed on an Elasticsearch cluster. Users can create bar, line and scatter plots, or pie charts and maps on top of large volumes of data.

33.3.3 Logstash

https://www.digitalocean.com/community/tutorials/how-to-use-logstash-and-kibana-to-centralize-logs-on-centos-6

33.3.4 Elasticsearch

Elasticsearch is a search server based on Lucene. It provides a distributed, multitenant-capable full-text search engine with a RESTful web interface and schema-free JSON documents. Elasticsearch is developed in Java and is released as open source under the terms of the Apache License. Elasticsearch is the second most popular enterprise search engine 1. Basic Concenpts Relational Database Elasticsearch Database Index Table Type Row Document Column Field Schema Mapping 2. Index Query Get all indices $/_statsSearchAPI1SearchAll/bank/_search?q = *hits.hits\breve{}actualarrayofsearchresults(defaultstofirst10documents)$

Query Language elasticsearch provides a full Query DSL based on JSON to define queries.

curl -XPOST /bank/$_search//matchall, limit10offset10"query" : "match_all" : , "from" : 10, "size" : 10$

// select fields "query": "match$_all" : ,_source : ["account_number", "balance"]"size" :$ 10

// where account equals 20 "query": "match": "account$_number" : 20Filter$

curl -XPOST elastic:9200/index/type/$_search-d'"query" : "filtered" : "query" : "term" : "feature" : 1,$

curl -XPOST elastic:9200/index/type/$_search-d'"query" : "filtered" : "query" : "term" : "feature" : 1,$

curl -XPOST localhost:9200/test "mappings" : "$_default" : "_timestamp" : "enabled" : true, "store" : true$

Easy, fast, performant No need for special queries Only applicable when one-to-one relationships are maintained Nested

Nested docs are stored in the same Lucene block as each other, which helps read/query performance. Reading a nested doc is faster than the equivalent parent/child. Updating a single field in a nested document (parent or nested children) forces ES to reindex the entire nested document. This can be very expensive for large nested docs "Cross referencing" nested documents is impossible Best suited for data that does not change frequently Parent/Child

Updating a child doc does not affect the parent or any other children, which can potentially save a lot of indexing on large docs Children are stored separately from the parent, but are routed to the same shard. So parent/children are slightly less performance on read/query than nested Parent/child mappings have a bit extra memory overhead, since ES maintains a "join" list in memory Sorting/scoring can be difficult with Parent/Child since the Has Child/Has Parent operations can be opaque at times Denormalization

You get to manage all the relations yourself! Most flexible, most administrative overhead May be more or less performant depending on your setup 4. Backup Elastic Dump 5 Tools for moving and saving indicies.

bin/elasticdump –input=http://localhost:9200/index$_1 - -output = http : //localhost : 9200/index_{1b}ackup--type = data--scrollTime = 100Alias6curl- XPOST'http : //localhost : 9200/_aliases'-d'quot; actionsquot; : [quot; removequot; : quot; indexquot; : quot;$ $EnginesRankingofSearchEngines$

The Search API http://stackoverflow.com/a/17146144/772391 http://stackoverflow.com/a/23407367/77 https://www.elastic.co/guide/en/elasticsearch/guide/current/modeling-your-data.html https://github.com/taskrabbit/elasticsearch-dump https://www.elastic.co/guide/en/elasticsearch/reference/ aliases.html https://www.elastic.co/guide/en/elasticsearch/reference/current/modules-scripting.html Elasticsearch tutorial series 1: Metric Aggregations with Social Network Data Table of content

Avg, Max, Min, Sum Aggregation Cardinality Aggregation Stats Aggregation Extended Stats Aggregation Percentile Aggregation Percentile Ranks Aggregation Top hits Aggregation Avg, Max, Min, Sum, Count Aggregation Doc: Avg Aggregation, Doc: Max Aggregation, Doc: Min Aggregation

Get max, min, avg, sum, count about number of likes, shares, comments

Request

POST /facebook$_crawler/post/_search"aggs" : "sum_like" : "sum" : "field" : "num_like", "min_like" : "mi$

"aggregations": "avg$_comment" : "value" : 75.23860589812332, "min_like" : "value" : 0, "avg_like" : "value" : 1761974365266098.2, "sum_like" : "value" : 3238508883359088600, "max_sha$ $"value" : 30407, "max_comment" : "value" : 11000, "sum_share" : "value" : 117844, "max_like" : "value" : 2751488761761411000, "avg_share" : "value" : 250.19957537154988, "sum_comment" : "value" : 28064, "min_comment" : "value" : 2, "min_share" : "value" : 1CardinalityAggregationCardinality$

Get total of users

Request

POST /facebook$_crawler/post/_search"aggs" : "num_authors" : "cardinality" : "field" : "from.fb_id"Res$

"aggregations": "num$_authors" : "value" : 7385StatsAggregationDoc : StatsAggregation$

Basic Stats of like, share comment

Request

POST /facebook$_crawler/post/_search"aggs" : "shares" : "stats" : "field" : "num_share", "likes" : "stats$

"aggregations": "shares": "count": 471, "min": 1, "max": 30407, "avg": 250.19957537154988, "sum": 117844 , "comments": "count": 373, "min": 2, "max": 11000, "avg": 75.23860589812332, "sum": 28064 , "likes": "count": 1838, "min": 0, "max": 2751488761761411000, "avg": 1761974365266098.2, "sum": 3238508883359088600 Extended Stats Aggregation Extended Stats Aggregation

Stats of like, share comment with more metrics, such as sum, std$_deviation, std_deviation_bounds, variance$

Request

POST /facebook$_crawler/post/_search"aggs" : "like_stats" : "extended_stats" : "field" : "num_like", "shar$

"aggregations": "like$_stats" : "count" : 1838, "min" : 0, "max" : 2751488761761411000, "avg" : 176197436$
$"count" : 471, "min" : 1, "max" : 30407, "avg" : 250.19957537154988, "sum" : 117844, "sum_of_squares" : 176$
$"count" : 373, "min" : 2, "max" : 11000, "avg" : 75.23860589812332, "sum" : 28064, "sum_of_squares" : 131531$
$PercentilesAggregation$

Comment, Like, Share Percentiles

Request

POST /facebook$_crawler/post/_search"aggs" : "like_percentiles" : "percentiles" : "field" : "num_like", "s$

"aggregations": "like$_percentiles" : "values" : "1.0" : 0, "5.0" : 0, "25.0" : 4, "50.0" : 18.35, "75.0" : 72.5357$
$"values" : "1.0" : 2, "5.0" : 2, "25.0" : 5, "50.0" : 10, "75.0" : 26, "95.0" : 139.39999999999998, "99.0" : 1000, "s$
$"values" : "1.0" : 1, "5.0" : 1, "25.0" : 1, "50.0" : 4, "75.0" : 25, "95.0" : 251.5, "99.0" : 5560.3LikePercentilesu$

Request

POST /facebook$_crawler/post/_search"aggs" : "share_percentiles" : "percentiles" : "field" : "num_share"$

"aggregations": "share$_percentiles" : "values" : "0.0" : 1, "10.0" : 1, "80.0" : 37.33333333333333, "90.0" : 9$
$PercentileRanksAggregation$

How like, share, comment distribute

Request

POST /facebook$_crawler/post/_search"aggs" : "like_percentile_ranks" : "percentile_ranks" : "field" : "nur$

"aggregations": "share$_percentile_ranks" : "values" : "10.0" : 60.438782731776364, "100.0" : 89.915074309$
$"values" : "10.0" : 39.281828073993466, "100.0" : 79.39530545624125, "1000.0" : 90.98349676683587, "10000.0$
$"values" : "10.0" : 49.865951742627345, "100.0" : 92.18395545473294, "1000.0" : 98.92761394101876, "10000.0$

Top hits Aggregation Doc: Top hits Aggregation

Example

Request

Response

An Aggregation Doc: Link

Config elasticsearch.yml

discovery.zen.minimum$_master_nodes : 1discovery.zen.ping.multicast.enabled :$
$falsediscovery.zen.ping.unicast.hosts : ["localhost"]$

network.host: 0.0.0.0 http.cors.enabled: true http.cors.allow-origin: '*' script.inline: on script.indexed: on Docker Image

https://hub.docker.com/r/$_/elasticsearch/$

Run

docker run -d -v "$PWD/esdata" : /usr/share/elasticsearch/dataelasticsearchDockerFolder$

elasticsearch/ config elasticsearch.yml Dockerfile Dockerfile

FROM elasticsearch:2.2.0

ADD config/elasticsearch.yml /elasticsearch/config/elasticsearch.yml Compose

elasticsearch: build: ./elasticsearch/. ports: - 9200:9200 - 9300:9300 volumes: - ./data/elasticsearch:/usr/share/elasticsearch/data Elasticsearch: Search Ignore Accents The ICU 1 2 analysis plug-in for Elasticsearch uses the International Components for Unicode (ICU) libraries to provide a rich set of tools for dealing with Unicode. These include the icu$_tokenizer, whichisparticularlyusefulforAsianlanguages, andanumbero$

Step 1: Install ICU-Plugin 3 cd /usr/share/elasticsearch sudo bin/plugin install analysis-icu Step 2: Create an analyzer setting: "settings": "analysis": "analyzer": "vnanalysis": "tokenizer": "icu$_tokenizer", "filter" : ["icu_folding", "icu_normalizer"]Step3 :$
$Createyourindex, createafieldwithtypestringandanalyzerisvnanalysisyouhavecreated"key" :$
$"type" : "string", "analyzer" : "vnanalysis"Step4 : SearchwithsensePOST/your_index/your_doc_type/_searc$
$ImportCSVtoElasticsearchhttps : //gist.github.com/clemsos/$8668698

Install lastest Elasticdump with NVM As a matter of best practice we'll update our packages:

apt-get update The build-essential package should already be installed, however, we're going still going to include it in our command for installation:

apt-get install build-essential libssl-dev To install or update nvm, you can use the install script using cURL:

curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.31.0/install.sh | bash if you have below problem or after you type nvm ls-remote command it result N/A: curl: (77) error setting certificate verify locations: CAfile: /etc/pki/tls/certs/ca-bundle.crt CApath: none

head to this 1:

or Wget:

wget -qO- https://raw.githubusercontent.com/creationix/nvm/v0.31.0/install.sh | bash Don't forget to restart your terminal

Then you use the following command to list available versions of nodejs

nvm ls-remote To download, compile, and install the latest v5.0.x release of node, do this:

nvm install 5.0 And then in any new shell just use the installed version:

nvm use 5.0 Or you can just run it:

nvm run 5.0 –version Or, you can run any arbitrary command in a subshell with the desired version of node:

nvm exec 4.2 node –version You can also get the path to the executable to where it was installed:

nvm which 5.0 Node Version Manager

how to solve https problem

ICU plug-in Github

Installing the ICU plug-in

33.3.5 Neo4J

version: 2.3.1

Neo4j is an open-source graph database, implemented in Java. The developers describe Neo4j as "embedded, disk-based, fully transactional Java persistence engine that stores data structured in graphs rather than in tables". Neo4j is the most popular graph database.

Installation

Docker Docker Image: https://hub.docker.com/r/library/neo4j/

Run these below command to open neo4j

clone datahub project git clone https://github.com/magizbox/datahub.git

change folder to datahub directory cd datahub

set your config in docker-compose.yml

run docker docker-compose up

Cypher

Schema Discovery List all nodes label, list all relation type

> START n=node(*) RETURN distinct labels(n)

> match n-[r]-() return distinct type(r) UI Way: Click to Overtab in Neo4j Browser

Sample 10 entities > MATCH (n:Entity) RETURN n, rand() as random ORDER BY random LIMIT 10 Group By http://www.markhneedham.com/blog/2013/02/17/neo4jcypher-sql-style-group-by-functionality/

Graph Algorithms

shortestPath, dijkstra

POST http://localhost:7474/db/data/node/72/paths

Headers Accept: application/json Authorization: Basic bmVvNGo6cGFzc3dk

Body "to" : "http://localhost:7474/db/data/node/77", "max$_depth" : 5, "relationships" : "type" : "FRIEND", "direction" : "out", "algorithm" : "shortestPath" GraphAnalystic$

pagerank, closeness$_centrality, betweenness_centrality, triangle_count, connected_components, strongly_conn$

Client

In this article you will know how to connect to neo4j database from python.

Python Client We can use Py2neo to connect to neo4j from python.

Py2neo is a client library and comprehensive toolkit for working with Neo4j from within Python applications and from the command line. The core library has no external dependencies and has been carefully designed to be easy and intuitive to use.

Snippets to connect, create, add nodes, add relationship and update property

from py2neo import authenticate, Graph, Node, Relationship connect to graph authenticate("localhost:7474", "neo4j", "passwd") graph = Graph("http://localhost:7474/db/data/")

create unique graph.schema.create$_uniqueness_constraint('Person', 'name')$

add nodes graph.create(Node.cast('Person', "name": "Alice")) graph.create(Node.cast('Person', "name": "Bob"))

add relationship source = graph.merge$_one("Person", "name", "Alice") target = graph.merge_one("Person", "name", "Bob") graph.create_unique(Relationship(source, "FRIEND", target))$

update property alice = graph.merge$_one("Person", "name", "Alice") alice["age"] = 30 alice.push()$

33.4 Web Crawling

33.4.1 Introduction

Web Crawler Static Crawler

Apache Nutch Dynamic Crawler

nutch-selenium Intelligent Extractor

boilerpipe Web Content Extraction Through Machine Learning Priority Crawler, Social Crawler

Features a crawler must provide We list the desiderata for web crawlers in two categories: features that web crawlers must provide, followed by features they should provide.

Robustness:

The Web contains servers that create spider traps, which are generators of web pages that mislead crawlers into getting stuck fetching an infinite number of pages in a particular domain. Crawlers must be designed to be resilient to such traps. Not all such traps are malicious; some are the inadvertent side-effect of faulty website development.

Politeness:

Web servers have both implicit and explicit policies regulating the rate at which a crawler can visit them. These politeness policies must be respected.

Features a crawler should provide Distributed The crawler should have the ability to execute in a distributed fashion across multiple machines.

Scalable

The crawler architecture should permit scaling up the crawl rate by adding extra machines and bandwidth.

Performance and efficiency

The crawl system should make efficient use of various system resources including processor, storage and network bandwidth.

Quality

Given that a significant fraction of all web pages are of poor utility for serving user query needs, the crawler should be biased towards fetching "useful" pages first.

Freshness

In many applications, the crawler should operate in continuous mode: it should obtain fresh copies of previously fetched pages. A search engine crawler, for instance, can thus ensure that the search engine's index contains a fairly current representation of each indexed web page. For such continuous crawling, a crawler should be able to crawl a page with a frequency that approximates the rate of change of that page.

Extensible

Crawlers should be designed to be extensible in many ways - to cope with new data formats, new fetch protocols, and so on. This demands that the crawler architecture be modular.

Crawling The basic operation of any hypertext crawler (whether for the Web, an intranet or other hypertext document collection) is as follows.

The crawler begins with one or more URLs that constitute a seed set. It picks a URL from this seed set, then fetches the web page at that URL. The fetched page is then parsed, to extract both the text and the links from the page (each of which points to another URL). The extracted text is fed to a text indexer. The extracted links (URLs) are then added to a URL frontier, which at all times consists of URLs whose corresponding pages have yet to be fetched by the crawler. Initially, the URL frontier contains the seed set; as pages are fetched, the corresponding URLs are deleted from the URL frontier. The entire process may be viewed as traversing the web graph. In continuous crawling, the URL of a fetched page is added back to the frontier for fetching again in the future. This seemingly simple recursive traversal of the web graph is complicated by the many demands on a practical web crawling system: the crawler has to be distributed, scalable, efficient, polite, robust and extensible while fetching pages of high quality. We examine the effects of each of these issues. Our treatment follows the design of the Mercator crawler that has formed the basis of a number of research and commercial crawlers. As a reference point, fetching a billion pages (a small fraction of the static Web at present) in a month-long crawl requires fetching several hundred pages each second. We will see how to use a multi-threaded design to address several bottlenecks in the overall crawler system in order to attain this fetch rate.

Before proceeding to this detailed description, we reiterate for readers who may attempt to build crawlers of some basic properties any non-professional crawler should satisfy:

Only one connection should be open to any given host at a time. A waiting time of a few seconds should occur between successive requests to a host. Politeness restrictions should be obeyed.

A New Approach to Dynamic Crawler Build a crawler system for dynamic websites is not easy task. While you can use a web browser automator (like selenium), or event when you can integrate selenium with nutch (by using nutch-selenium). These solutions are still hard to develop, hard to test and hard to manage sessions because we still "translate" our process to languages (such as java or python)

I suppose a new approach for this problem. Instead of using a web browser automator, we can inject native javascript codes into browser (via extension or add-on).The advantages of this approach is we can easily inject third party libraries (like jquery (for dom selector), Run.js (for complicated process) and APIs that supported by browsers). And we can take advance of debugging tool and testing framework in javascript world.

If you want to know about more details, feel free to contact me.

33.4.2 Scrapy

Scrapy An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

Build and run your web spiders $pipinstallscrapy$ cat > myspider.py «EOF import scrapy

class BlogSpider(scrapy.Spider): name = 'blogspider' start$_urls = ['https://blog.scrapinghub.com']$

def parse(self, response): for title in response.css('h2.entry-title'): yield 'title': title.css('a ::text').extract$_first()$

next$_page = response.css('div.prev-post > a :: attr(href)').extract_first()ifnext_page : yieldscrapy.Request(response.urljoin(next_page), callback = self.parse)EOF$
scrapy runspider myspider.py Deploy them to Scrapy Cloud $shubloginInsertyourScrapinghubAPIKey :< API_KEY >$

Deploy the spider to Scrapy Cloud

$shubdeploy$

Schedule the spider for execution shub schedule blogspider Spider blogspider scheduled, watch it running here: https://app.scrapinghub.com/p/26731/job/1/8

Retrieve the scraped data $shubitems$26731/1/8"$title$" : "$ImprovedFrontera : WebCrawlingatScalewith$

33.4.3 Apache Nutch

Highly extensible, highly scalable Web crawler 1 Nutch is a well matured, production ready Web crawler. Nutch 1.x enables fine grained configuration, relying on Apache Hadoop™ data structures, which are great for batch processing.

History

Usecases

1. Features 1 1. Transparency Nutch is open source, so anyone can see how the ranking algorithms work. With commercial search engines, the precise details of the algorithms are secret so you can never know why a particular search result is ranked as it is. Furthermore, some search engines allow rankings to be based on payments, rather than on the relevance of the site's contents. Nutch is a good fit for academic and government organizations, where the perception of fairness of rankings may be more important.

2. Understanding We don't have the source code to Google, so Nutch is probably the best we have. It's interesting to see how a large search engine works. Nutch has been built using ideas from academia and industry: for instance, core parts of Nutch are currently being re-implemented to use the MapReduce.

Map Reduce distributed processing model, which emerged from Google Labs last year. And Nutch is attractive for researchers who want to try out new search algorithms, since it is so easy to extend.

3. Extensibility Don't like the way other search engines display their results? Write your own search engine–using Nutch! Nutch is very flexible: it can be customized and incorporated into your application. For developers, Nutch is a great platform for adding search to heterogeneous collections of information, and being able to customize the search interface, or extend the out-of-the-box functionality through the plugin mechanism. For example, you can integrate it into your site to add a search capability.

Process 5 0. initialize CrawlDb, inject seed URLs Repeat generate-fetch-update cycle n times:

1. The Injector takes all the URLs of the nutch.txt file and adds them to the CrawlDB. As a central part of Nutch, the CrawlDB maintains information on all known URLs (fetch schedule, fetch status, metadata, . . .).

2. Based on the data of CrawlDB, the Generator creates a fetchlist and places it in a newly created Segment directory.

3. Next, the Fetcher gets the content of the URLs on the fetchlist and writes it back to the Segment directory. This step usually is the most time-consuming one.

4. Now the Parser processes the content of each web page and for example omits all html tags. If the crawl functions as an update or an extension to an already existing one (e.g. depth of 3), the Updater would add the new data to the CrawlDB as a next step.

5. Before indexing, all the links need to be inverted by Link Inverter, which takes into account that not the number of outgoing links of a web page is of interest, but rather the number of inbound links. This is quite similar to how Google PageRank works and is important for the scoring function. The inverted links are saved in the Linkdb.

6-7. Using data from all possible sources (CrawlDB, LinkDB and Segments), the Indexer creates an index and saves it within the Solr directory. For indexing, the popular Lucene library is used. Now, the user can search for information regarding the crawled web pages via Solr.

Installation Requirements

1. OpenJDK 7

2. Nutch 2.3 RC (yes, you need 2.3, 2.2 will not work)

wget https://archive.apache.org/dist/nutch/2.3/apache-nutch-2.3-src.tar.gz tar -xzf apache-nutch-2.3-src.tar.gz 3. HBase 0.94.27 (HBase 0.98 won't work)

wget https://www.apache.org/dist/hbase/hbase-0.94.27/hbase-0.94.27.tar.gz tar -xzf hbase-0.94.27.tar.gz 4. ElasticSearch 1.7

wget https://download.elastic.co/elasticsearch/elasticsearch/elasticsearch-1.7.0.tar.gz tar -xzf elasticsearch-1.7.0.tar.gz Other Options: nutch-2.3, hbase-0.94.26, ElasticSearch 1.4

Setup HBase 1. edit $HBASE_ROOT/conf/hbase - site.xmlandadd$

<configuration> <property> <name>hbase.rootdir</name> <value>file:///full/path/to/where/the/da
</property> <property> <name>hbase.cluster.distributed</name> <value>false</value>
</property> </configuration> 2. edit $HBASE_ROOT/conf/hbase-env.shandenableJAVA_HOMEandseti$

- export JAVA$_HOME = /usr/java/jdk1.6.0/ + exportJAVA_HOME = /usr/lib/jvm/java-7-openjdk-amd64/Thisstepmightseemredundant, butevenwithJAVA_HOMEbeingse$

3. kick off HBase:

HBASE$_ROOT/bin/start-hbase.shConfigureNutch1.EnabletheHBasedependencyin$NUTCH$_ROOT/iv$

<dependency org="org.apache.gora" name="gora-hbase" rev="0.5" conf="*->default" /> 2. Configure the HBase adapter by editing the $NUTCH_ROOT/conf/gora.properties$

-gora.datastore.default=org.apache.gora.mock.store.MockDataStore +gora.datastore.default=org.apache.
3. Build Nutch

cdNUTCH$_ROOTantcleanantruntimeThiscantakeawhileandcreates$NUTCH$_ROOT/runtime/local.$

4. configure Nutch by editing $NUTCH_ROOT/runtime/local/conf/nutch-site.xml$

<configuration> <property> <name>http.agent.name</name> <value>mycrawlername</value> <!– this can be changed to something more sane if you like –> </property> <property> <name>http.robots.agents</name> <value>mycrawlername</value> <!– this is the robot name we're looking for in robots.txt files –> </property> <property> <name>storage.data.store.class</name> <value>org.apache.gora.hbase.store.HBaseStore</va </property> <property> <name>plugin.includes</name> <!– do NOT enable the parse-html plugin, if you want proper HTML parsing. Use something like parse-tika! –> <value> protocol-httpclient|urlfilter-regex|parse-(text|tika|js)|index-(basic|anchor)|query-(basic|site|url)|response-(json|xml)|summary-basic|scoring-opic|urlnormalizer-(pass|regex|basic)|indexer-elastic </value> </property> <property> <name>db.ignore.external.links</name> <value>true</value> <!– do not leave the seeded domains (optional) –> </property> <property> <name>elastic.host</name> <value>localhost</value> <!– where is ElasticSearch listening –> </property> </configuration> or you configure Nutch by editing $NUTCH_ROOT/runtime/local/conf/nutch-site.xml$

<configuration> <property> <name>plugin.includes</name> <!– do × NOT enable the parse-html plugin, if you want proper HTML parsing. Use something like parse-tika! –> <value> protocol-http|protocol-httpclient|urlfilter-regex| parse-(text|tika|js)|index-(basic|anchor)|query-(basic|site|url)|response-(json|xml)| summary-basic|scoring-opic|urlnormalizer-(pass|regex|basic)|indexer-elastic| index-metadata|index-more </value> </property> <property> <name>db.ignore.external.links</name> <value>true</value> <!– do not leave the seeded domains (optional) –> </property>

<!– elasticsearch index properties –> <property> <name>elastic.host</name> <value>localhost</value> <description>The hostname to send documents to using TransportClient. Either host and port must be defined or cluster. </description> </property>

<property> <name>elastic.port</name> <value>9300</value> <description> The port to connect to using TransportClient. </description> </property> <property> <name>elastic.index</name> <value>nutch</value> <description> The name of the elasticsearch index. Will normally be autocreated if it doesn't exist. </description> </property> <!– end index –> </configuration> 5. configure HBase integration by editing $NUTCH_ROOT/runtime/local/conf/hbase-site.xml$

<?xml version="1.0" encoding="UTF-8"?> <configuration> <property> <name>hbase.rootdir</name> <value>file:///full/path/to/where/the/data/should/be/stored</value> <!– same path as you've given for HBase above –> </property> <property> <name>hbase.cluster.distributed</name> <value>false</value> </property> </configuration> or you configure HBase integration by editing $NUTCH_ROOT/runtime/local/conf/hbase-site.xml$:

<configuration> <property> <name>hbase.rootdir</name> <value>file:///$PATH/database < /value >< /property >< property >< name > hbase.cluster.distributed <$

$/name >< value > false < /value >< /property >< property >< name >$ $hbase.zookeeper.quorum < /name >< value > hbase.io < /value >< /property ><$ $property >< name > zookeeper.znode.parent < /name >< value > /hbase -$ $unsecure < /value >< /property >< property >< name > hbase.rpc.timeout <$ $/name >< value > 2592000000 < /value >< /property >< /configuration >$ $That'sit.Everythingisnowsetuptocrawlwebsites.$

Run Nutch 1. Create an empty directory. Add a textfile containing a list of seed URLs

$mkdirseed$ echo "https://www.website.com" » seed/urls.txt $echo"https :$ $//www.another.com" >> seed/urls.txt$ echo "https://www.example.com" » seed/urls.txt Inject them into Nutch by giving a file URL (!)

NUTCH$_{R}OOT/runtime/local/bin/nutchinjectfile : ///path/to/seed/2.GenerateanewsetofURLstofe$

This is is based on both the injected URLs as well as outdated URLs in the Nutch crawl db.

NUTCH$_{R}OOT/runtime/local/bin/nutchgenerate-topN10Theabovecommandwillcreatejobbatchesfor1$

3. Fetch the URLs. We are not clustering, so we can simply fetch all batches:

NUTCH$_{R}OOT/runtime/local/bin/nutchfetch-all4.Nowweparseallfetchedpages :$

NUTCH$_{R}OOT/runtime/local/bin/nutchparse-all5.Laststep : UpdateNutch'sinternaldatabase :$

NUTCH$_{R}OOT/runtime/local/bin/nutchupdatedb-allOnthefirstrun, thiswillonlycrawltheinjectedUR$

6. Putting Documents into ElasticSearch

NUTCH$_{R}OOT/runtime/local/bin/nutchindex-allConfigurationCrawlnutchviaproxy$

Change $NUTCH_{R}OOT/runtime/local/conf/nutch - site.xml$

<configuration> <property> <name>http.proxy.host</name> <value>192.168.80.1</value> <description>The proxy hostname. If empty, no proxy is used.</description> </property> <property> <name>http.proxy.port</name> <value>port</value> <description>The proxy port.</description> </property> <property> <name>http.proxy.username</nar <value>username</value> <description>Username for proxy. This will be used by 'protocol-httpclient', if the proxy server requests basic, digest and/or NTLM authentication. To use this, 'protocol-httpclient' must be present in the value of 'plugin.includes' property. NOTE: For NTLM authentication, do not prefix the username with the domain, i.e. 'susam' is correct whereas 'DOMAIN-susam' is incorrect. </description> </property> <property> <name>http.proxy.password</name> <value>password</value> <description>Password for proxy. This will be used by 'protocol-httpclient', if the proxy server requests basic, digest and/or NTLM authentication. To use this, 'protocol-httpclient' must be present in the value of 'plugin.includes' property. </description> </property> </configuration> Nutch Plugins Extension Points In writing a plugin, you're actually providing one or more extensions of the existing extension-points . The core Nutch extension-points are themselves defined in a plugin, the NutchExtensionPoints plugin (they are listed in the NutchExtensionPoints plugin.xml file). Each extension-point defines an interface that must be implemented by the extension. The core extension points are:

Point Description Example IndexWriter Writes crawled data to a specific indexing backends (Solr, ElasticSearch, a CVS file, etc.). IndexingFilter Permits one to add metadata to the indexed fields. All plugins found which implement this extension point are run sequentially on the parse (from javadoc). Parser Parser implementations read through fetched documents in order to extract data to be indexed. This is what you need to implement if you want Nutch to be able to parse a new type of content, or extract more data from currently parseable content. HtmlParseFilter Permits one to add additional metadata to HTML parses (from javadoc). Protocol Protocol implementations allow Nutch to use different protocols (ftp, http, etc.) to fetch documents. URLFilter URLFilter implementations limit the URLs that Nutch attempts to fetch. The RegexURLFilter distributed with Nutch provides a great deal of control over what URLs Nutch crawls, however if you have very complicated rules about what URLs you want to crawl, you can write your own implementation. URLNormalizer Interface used to convert URLs to normal form and optionally perform substitutions. ScoringFilter A contract defining behavior of scoring plugins. A scoring filter will manipulate scoring variables in CrawlDatum and in resulting search indexes. Filters can be chained in a specific order, to provide multi-stage scoring adjustments. SegmentMergeFilter Interface used to filter segments during segment merge. It allows filtering on more sophisticated criteria than just URLs. In particular it allows filtering based on metadata collected while parsing page. Getting Nutch to Use a Plugin In order to get Nutch to use a given plugin, you need to edit your conf/nutch-site.xml file and add the name of the plugin to the list of plugin.includes. Additionally we are required to add the various build configurations to build.xml in the plugin directory.

Develop nutch plugins Project structure of a plugin plugin-name plugin.xml build.xml ivy.xml src org apache nutch indexer uml-meta source folder URLMetaIndexingFilter.java scoring uml-meta source folder URLMetaScoringFilter.java test org apache nutch indexer uml-meta test folder URLMetaIndexingFilterTest.java scoring uml-meta test folder URLMetaScoringFilterTest.java Follow this link to read develop nutch plugins

Architecture

Architectures

Data Structure The web database is a specialized persistent data structure for mirroring the structure and properties of the web graph being crawled. It persists as long as the web graph that is being crawled (and re-crawled) exists, which may be months or years. The WebDB is used only by the crawler and does not play any role during searching. The WebDB stores two types of entities: pages and links.

A page represents a page on the Web, and is indexed by its URL and the MD5 hash of its contents. Other pertinent information is stored, too, including

the number of links in the page (also called outlinks); fetch information (such as when the page is due to be refetched); the page's score, which is a measure of how important the page is (for example, one measure of importance awards high scores to pages that are linked to from many other pages). A link represents a link from one web page (the source) to another (the target). In the WebDB web graph, the nodes are pages and the edges are links.

A segment is a collection of pages fetched and indexed by the crawler in a single run. The fetchlist for a segment is a list of URLs for the crawler to fetch, and is generated from the WebDB. The fetcher output is the data retrieved from the pages in the fetchlist. The fetcher output for the segment is indexed and the index is stored in the segment. Any given segment has a limited lifespan, since it is obsolete as soon as all of its pages have been re-crawled. The default re-fetch interval is 30 days, so it is usually a good idea to delete segments older than this, particularly as they take up so much disk space. Segments are named by the date and time they were created, so it's easy to tell how old they are.

The index is the inverted index of all of the pages the system has retrieved, and is created by merging all of the individual segment indexes. Nutch uses Lucene for its indexing, so all of the Lucene tools and APIs are available to interact with the generated index. Since this has the potential to cause confusion, it is worth mentioning that the Lucene index format has a concept of segments, too, and these are different from Nutch segments. A Lucene segment is a portion of a Lucene index, whereas a Nutch segment is a fetched and indexed portion of the WebDB.

View gora-hbase-mapping.xml for more details

Config

Config nutch run intellij Copy file

copy all the files in the runtime/conf on out/test/apache-Nutch-2.3 and out/production/apache-Nutch-2.3

add these lines to file $NUTCH_SRC/out/test/nutch-site.xml$

<property> <name>plugin.folders</name> <value><nutch$_src > /build/plugins < /value >< /property > RunnutchinintellijRun- > EditConfigurations...- > addpathagrs : pathtofilelistlinkscrawlerDevNutchinIntellijReceipts : IntellIJ14, ApacheNutch2.3$

1. Get Nutch source

wget http://www.eu.apache.org/dist/nutch/2.3/apache-nutch-2.3-src.tar.gz tar -xzf apache-nutch-2.3-src.tar.gz 2. Import Nutch source in IntellIJ

[wonderplugin$_slider id = $"1"]

3. Get Dependencies by Ant

[wonderplugin$_slider id = $"3"]

4. Import Dependencies to IntellIJ

[wonderplugin$_slider id = $"4"]

Nutch Dev 1.Intasll java in ubuntu

-Downloads java version .zip

http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html -Create folder jvm

sudo mkdir /usr/lib/jvm/ -Cd to folder downloads java version .zip

sudo mv jdk1.7.0$_x//usr/lib/jvm/jdk1.7.0_x - Runcommandline$

sudo update-alternatives –install /usr/bin/java java /usr/lib/jvm/jdk1.7.0$_x/jre/bin/java0- Tetsversionjava$

java -version 2.Intasll ant in ubuntu

-Downloads ant

http://ant.apache.org/manualdownload.cgi -Add path ant vao file environment

sudo nano /etc/environment $ANT_ROOT/bin - Runcommandline$

source /etc/environment ant -version 3.Intasll hbase in ubuntu

-Downloads and extract hbase 0.94.27

https://archive.apache.org/dist/hbase/hbase-0.94.27/ -Edit file $HABSE_ROOT/conf/hbase- site.xml$

<configuration> <property> <name>hbase.rootdir</name> <value>file:///$PATH_DATA_BASE/datab$ $/value >< /property >< property >< name > hbase.cluster.distributed < /name >< value > false < /value >< /property >< property >< name > hbase.zookeeper.quorum < /name >< value > hbase.io < /value >< /property >< property >< name > zookeeper.znode.parent < /name >< value > /hbase - unsecure < /value >< /property >< property >< name > hbase.rpc.timeout <$

$/name >< value >$ 2592000000 $< /value >< /property >< /configuration >$ $-Editfile$HBASE$_ROOT/conf/hbase - env.sh$

export JAVA$_HOME$ =PATH$_JAVA_HOME-Editfile$HBASE$_ROOT/conf/regionservers$

hbase.io.nutch -Edit file hosts in ubuntu

sudo nano /etc/hosts ip hbase.io.nutch -Edit file hostname in ubuntu

sudo nano /etc/hostname hbase.io.nutch -Run and stop hbase in ubuntu

Run hbase : cd $HBASE_ROOT/bin./start-hbase.shStophbase$: cdHBASE$_ROOT/bin./stop-hbase.sh * Errorinintasllhbase$

- Error regionserver localhost(Edit file hosts and file host name) - Error client no remote server intasll hbase(Turn off file firewall) 4.Build nutch in ant

-Downloads and extract nutch

http://nutch.apache.org/ -Edit file $NUTCH_ROOT/ivy/ivy.xml$

<dependency org="org.apache.gora" name="gora-hbase" rev="0.5" conf="*->default" /> -Edit file $NUTCH_ROOT/ivy/ivysettings.xml$

<property name="repo.maven.org" value="http://repo1.maven.org/maven2/" override="false"/>

<property name = "repo.maven.org" value = "http://maven.oschina.net/content/groups/public/" override = "false" /> -Edit file $NUTCH_ROOT/conf/nutch - site.xml$

<configuration> <property> <name>plugin.folders</name> <value>$NUTCH_ROOT/build/plugins < /value >< /property >< property >< name > http.agent.name < /name >< value > mycrawlername < /value ><!--thiscanbechangedtosomethingmoresaneifyoulike-- >< /property >< property >< name > http.robots.agents < /name >< value > mycrawlername < /value ><!--thisistherobotnamewe'relookingforinrobots.txtfiles-- >< /property >< property >< name > storage.data.store.class < /name >< value > org.apache.gora.hbase.store.HBaseStore < /value >< /property >< property >< name > plugin.includes < /name ><!--doNOTenabletheparse-htmlplugin, ifyouwantproperHTMLparsing.Usesomethinglikeparse-tika!-- >< value > protocol - http|protocol - httpclient|urlfilter - regex|parse - (text|tika|js)|index-(basic|anchor)|query-(basic|site|url)|response-(json|xml)|summary-basic|scoring-opic|urlnormalizer-(pass|regex|basic)|indexer-elastic|index-metadata|index - more < /value >< /property >< property >< name > db.ignore.external.links < /name >< value > true < /value ><!--donotleavetheseededdomains(optiona - >< /property >$

<!– elasticsearch index properties –> <property> <name>elastic.host</name> <value>localhost</value> <description>The hostname to send documents to using TransportClient. Either host and port must be defined or cluster. </description> </property>

<property> <name>elastic.port</name> <value>9300</value> <description> The port to connect to using TransportClient. </description> </property> <property> <name>elastic.index</name> <value>nutch</value> <description> The name of the elasticsearch index. Will normally be autocreated if it doesn't exist. </description> </property> <!– end index –>

<property> <name>http.proxy.host</name> <value>192.168.80.1</value> </property> <property> <name>http.proxy.port</name> <value>8080</value> </property> <property> <name>http.proxy.username</name> <value>user1</value> </property> <property> <name>http.proxy.password</name> <value>user1</value> </property> </configuration> -Edit file file $NUTCH_ROOT/conf/gora.property$

gora.datastore.default=org.apache.gora.hbase.store.HBaseStore -Build nucth

ant runtime or ant eclipse -verbose -Create file links

-Run nutch

cd $NUTCH_ROOT/runtime/local/binruninject: ./nutchinjectfile: ///\mathrm{PATH}_LIKNSrungenerate: ./nutchgenerate{-}topN10runfetch: ./nutchfetch{-}allrunparse: ./nutchparse{-}allrunupdatedb: ./nutchupdatedb - all - Downloadsandextractelastic$

https://www.elastic.co/downloads/elasticsearch -Run elastic

cd $ELASTIC/bin./elasticsearch - Indexdatainelastic$

cd $NUTCH_ROOT/runtime/binrunindex: ./nutchindex{-}all5.Runnutchintellij$

Change $NUTCH_ROOT/runtime/local/conf/hbase - site.xml$

<configuration> <property> <name>hbase.rootdir</name> <value>file:///home/hainv/Downloads/cr </property> <property> <name>hbase.cluster.distributed</name> <value>false</value> </property> <property> <name>hbase.zookeeper.quorum</name> <value>hbase.io</value> </property> <property> <name>zookeeper.znode.parent</name> <value>/hbase-unsecure</value> </property> <property> <name>hbase.rpc.timeout</name> <value>2592000000</value> </property> </configuration> Nutch plugin intellij 1.Structure nutch :[1] 2.Run nutch intellij Downloads nucth2.3:http://nutch.apache.org/downloads.html Editing file $NUTCH_ROOT/ivy/ivysettings.xml$

<ivysettings> <property name="oss.sonatype.org" value="http://oss.sonatype.org/content/repositories/ override="false"/> <property name = "repo.maven.org" value = "http://maven.oschina.net/content/groups/ override = "false" /> <property name="repository.apache.org" value="https://repository.apache.org/conten override="false"/> <property name="maven2.pattern" value="[organisation]/[module]/[revision]/[module]-[revision]"/> <property name="maven2.pattern.ext" value="$maven2.pattern.[ext]"/ >< !{-}{-}pullinthelocalrepository{-}{-} >< includeurl = "$ivy.default.conf.dir/ivyconf-local.xml"/> <settings defaultResolver="default"/> <resolvers> <ibiblio name="maven2" root="$repo.maven.org"pattern = "$maven2.pattern.ext" m2compatible="true" /> <ibiblio name="apache-snapshot" root="$repository.apache.org"changingPattern = ".*{-}SNAPSHOT"m2compatible = "true"/ >< ibiblioname = "restlet"root = "http://maven.restlet.org"pattern = "$maven2.pattern.ext" m2compatible="true" /> <ibiblio name="sonatype" root="$oss.sonatype.org"pattern = "$maven2.pattern.ext" m2compatible="true" />

<chain name="default" dual="true"> <resolver ref="local"/> <resolver ref="maven2"/> <resolver ref="sonatype"/> <resolver ref="apache-snapshot"/> </chain> <chain name="internal"> <resolver ref="local"/> </chain> <chain name="external"> <resolver ref="maven2"/> <resolver ref="sonatype"/> </chain> <chain name="external-and-snapshots"> <resolver ref="maven2"/> <resolver ref="apache-snapshot"/> <resolver ref="sonatype"/> </chain> <chain name="restletchain"> <resolver ref="restlet"/> </chain> </resolvers> <modules> <module organisation="org.apache.nutch" name=".*" resolver="internal"/> <module organisation="org.restlet" name=".*" resolver="restletchain"/> <module organisation="org.restlet.jse" name=".*" resolver="restletchain"/> </modules> </ivysettings> Editing file $NUTCH_ROOT/ivy/ivy.xml$

<dependency org="org.apache.gora" name="gora-hbase" rev="0.5" conf="*->default" /> Editing file $NUCTH_ROOT/conf/gora.properties$

gora.datastore.default=org.apache.gora.hbase.store.HBaseStore Editing file $NUTCH_ROOT/conf/nutch_site.xml$

<configuration> <property> <name>plugin.folders</name> <value>$NUTCH_ROOT/build/plugins < /value >< /property >< property >< name > http.agent.name < /name >< value > mycrawlername < /value ><!{-}{-}thiscanbechangedtosomethingmoresaneifyoulike{-}{-} >< /property >< property >< name > http.robots.agents < /name >< value > mycrawlername < /value ><!{-}{-}thisistherobotnamewe'relookingforinrobots.txtfiles{-}{-} >< /property >< property >< name > storage.data.store.class < /name >< value > org.apache.gora.hbase.store.HBaseStore < /value >< /property ><$

$property >< name > plugin.includes < /name ><!--doNOTenabletheparse-htmlplugin, ifyouwantproperHTMLparsing.Usesomethinglikeparse-tika!-- >< value > protocol-httpclient|urlfilter-regex|parse-(text|tika|js)|index-(basic|anchor)|query-(basic|site|url)|response-(json|xml)|summary-basic|scoring-opic|urlnormalizer-(pass|regex|basic)|indexer-elastic < /value >< /property >< property >< name > db.ignore.external.links < /name >< value > true < /value ><! - -donotleavetheseededdomains(optional) - - >< /property >< property >< name > elastic.host < /name >< value > localhost < /value >< ! - -whereisElasticSearchlistening - - >< /property >$

<property> <name>http.proxy.host</name> <value>192.168.80.1</value> <description>The proxy hostname. If empty, no proxy is used.</description> </property> <property> <name>http.proxy.port</name> <value>8080</value> <description>The proxy port.</description> </property> <property> <name>http.proxy.username</nam <value>user1</value> <description>Username for proxy. This will be used by 'protocol-httpclient', if the proxy server requests basic, digest and/or NTLM authentication. To use this, 'protocol-httpclient' must be present in the value of 'plugin.includes' property. NOTE: For NTLM authentication, do not prefix the username with the domain, i.e. 'susam' is correct whereas 'DOMAINsusam' is incorrect. </description> </property> <property> <name>http.proxy.password</name> <value>user1</value> <description>Password for proxy. This will be used by 'protocol-httpclient', if the proxy server requests basic, digest and/or NTLM authentication. To use this, 'protocol-httpclient' must be present in the value of 'plugin.includes' property. </description> </property> </configuration> Editing file $NUCTH_ROOT/conf/hbase-site.xml$

<configuration> <property> <name>hbase.rootdir</name> <value>file:///home/rombk/Downloads/c </property> <property> <name>hbase.cluster.distributed</name> <value>false</value> </property> <property> <name>hbase.zookeeper.quorum</name> <value>hbase.io</value> </property> <property> <name>zookeeper.znode.parent</name> <value>/hbase-unsecure</value> </property> <property> <name>hbase.rpc.timeout</name> <value>2592000000</value> </property> </configuration> Run terminal

ant eclipse -verbose Import nucth intellij

3.Run plugin creativecommons Sample plugins that parse and index Creative Commons medadata.1 Step 1. Create folder creativecommons in path $NUTCH_HOME/out/test/$

Step 2. Create file nutch-site.xml in folder $NUTCH_HOME/out/test/creativecommonsandaddcontent$

<?xml version="1.0"?> <?xml-stylesheet type="text/xsl" href="configuration.xsl"?> <!– Put site-specific property overrides in this file. –> <configuration> <property> <name>plugin.folders</name> <value>$NUTCH_HOME/build/plugins < /value >< /property >< property >< name > http.agent.name < /name >< value > mycrawlername < /value ><!--thiscanbechangedtosomethingmoresaneifyoulike-- >< /property >< property >< name > http.robots.agents < /name >< value > mycrawlername < /value ><!--thisistherobotnamewe'relookingforinrobots.txtfiles-- >< /property >< property >< name > storage.data.store.class < /name >< value > org.apache.gora.hbase.store.HBaseStore < /value >< /property >< property >< name > plugin.includes < /name ><!--doNOTenabletheparse-htmlplugin, ifyouwantproperHTMLparsing.Usesomethinglikeparse-tika!-- >< value > indexer-elastic|creativecommons|parse-html < /value >< /property >< property >< name > db.ignore.external.links < /name >< value > true < /value ><!--donotleavetheseededdomains(optional)-- >< /property >< property >< name > elastic.host < /name >< value >$

localhost < /value ><!−−whereisElasticSearchlistening−− >< /property ><
! − −configproxy − − >< property >< name > http.proxy.host < /name ><
value >< hosts >< /value >< description > Theproxyhostname.Ifempty, noproxyisused. <
/description >< /property >< property >< name > http.proxy.port <
/name >< value >< port >< /value >< description > Theproxyport. <
/description >< /property >< property >< name > http.proxy.username <
/name >< value >< user1 >< /value >< description > Usernameforproxy.Thiswillbeusedby′protocol−
httpclient′, iftheproxyserverrequestsbasic, digestand/orNTLMauthentication.Tousethis,′ protocol−
httpclient′mustbepresentinthevalueof′plugin.includes′property.NOTE : ForNTLMauthentication, donot
/description >< /property >< property >< name > http.proxy.password <
/name >< value >< user1 >< /value >< description > Passwordforproxy.Thiswillbeusedby′protocol−
httpclient′, iftheproxyserverrequestsbasic, digestand/orNTLMauthentication.Tousethis,′ protocol−
httpclient′mustbepresentinthevalueof′plugin.includes′property. < /description ><
/property >< /configuration > 2.RunpluginfeedPluginfeedparsingofrssError :
ParsingofRSSfeedsfails(tejasp)[2]andreadfileNUTCH$_R$OOT/CHANFES.txt

Phần V

Linh tinh

Chương 34

Nghiên cứu

Các công cụ

[Google Scholar](https://scholar.google.com.vn/) vẫn là lựa chọn tốt

* Tìm kiếm tác giả theo lĩnh vực nghiên cứu và quốc gia: sử dụng filter label: + đuôi * ví dụ: [danh sách các nhà nghiên cứu Việt Nam thuộc lĩnh vực xử lý ngôn ngữ tự nhiên (label:natural$_{l}anguage_{p}rocessing + .vn)](https : //scholar.google.com.vn/citations?hl = enamp; view_{o}p = search_{a}uthorsamp; mauthors = label * danhschnyØspxptheolngtrchdn$

Bên cạnh đó còn có [semanticscholar](https://www.semanticscholar.org/) (một project của [allenai](http://allenai.org/)) với các killer features

* [Tìm kiếm các bài báo khoa học với từ khóa và filter theo năm, tên hội nghị](https://www.semanticscholar.org/search?venue* [Xem những người ảnh hưởng, ảnh hưởng bởi một nhà nghiên cứu, cũng như xem co-author, journals và conferences mà một nhà nghiên cứu hay gửi bài](https://www.semanticscholar.org/author/Christopher-D-Manning/1812612)

Mendeley rất tốt cho việc quản lý và lưu trữ. Tuy nhiên điểm hạn chế lại là không lưu thông tin về citation

Các hội nghị tốt về xử lý ngôn ngữ tự nhiên

* Rank A: ACL, EACL, NAACL, EMNLP, CoNLL * Rank B: SemEval

Các tạp chí

* [Computational Linguistics (CL)](http://www.mitpressjournals.org/loi/coli)

Câu chuyện của Scihub

Sci-Hub được tạo ra vào ngày 5 tháng 9 năm 2011, do nhà nghiên cứu đến từ Kazakhstan, [Alexandra Elbakyan](https://en.wikipedia.org/wiki/Alexandra$_{E}lbakyan$)

Hãy nghe chia sẻ của cô về sự ra đời của Sci-Hub

> Khi tôi còn là một sinh viên tại Đại học Kazakhstan, tôi không có quyền truy cập vào bất kỳ tài liệu nghiên cứu. Những bai bao tôi cần cho dự án nghiên cứu của tôi. Thanh toán 32 USD thi thât la điên rô khi bạn cần phải đọc lướt hoặc đọc hàng chục hoặc hàng trăm tơ để làm nghiên cứu. Tôi có được những bai bao nhơ vao trôm chúng. Sau đó tôi thấy có rất nhiều và rất nhiều nhà nghiên cứu (thậm chí không phải sinh viên, nhưng các nhà nghiên cứu trường đại học) giống như tôi, đặc biệt là ở các nước đang phát triển. Họ đã tạo ra các cộng đồng trực tuyến (diễn đàn) để giải quyết vấn đề này. Tôi là một thành viên tích cực trong một cộng đồng như vậy ở Nga. Ở đây ai cần có một bài nghiên cứu, nhưng không thể trả tiền cho nó, có thể đặt một yêu cầu và các thành viên

khác, những người có thể có được những giấy sẽ gửi nó cho miễn phí qua email. Tôi có thể lấy bất cứ bai nao, vì vậy tôi đã giải quyết nhiều yêu cầu và người ta luôn rất biết ơn sự giúp đỡ của tôi. Sau đó, tôi tạo Sci-Hub.org, một trang web mà chỉ đơn giản là làm cho quá trình này tự động và các trang web ngay lập tức đã trở thành phổ biến.

Về phần mình, là một nhà nghiên cứu trẻ, đương nhiên phải đọc liên tục. Các báo cáo ở Việt Nam về xử lý ngôn ngữ tự nhiên thì thường không tải lên các trang mở như arxiv.org, các kỷ yếu hội nghị cũng không public các proceedings. Thật sự scihub đã giúp mình rất nhiều.

Scihub bị chặn*

Vào thời điểm này (12/2017), scihub bị chặn quyết liệt. Hóng được trên page facebook của scihub các cách truy cập scihub. Đã thử các domain khác như .tw, .hk. Mọi chuyện vẫn ổn cho đến hôm nay (21/12/2017), không thể truy cập vào nữa.

Đành phải cài tor để truy cập vào scihub ở địa chỉ http://scihub22266oqcxt.onion/https://dl.acm.org/citati
Và mọi chuyện lại ổn.

Làm sao để nghiên cứu tốt

* Làm việc mỗi ngày * Viết nhật ký nghiên cứu mỗi tuần (tổng kết công việc tuần trước, các ý tưởng mới, kế hoạch tuần này) * Cập nhật các kết quả từ các hội nghị, tạp chí

Sách giáo khoa

* [Machine Learning Yearning, by Andrew Ng](https://gallery.mailchimp.com/dc3a7ef4d750c0abfc19202a3

Lượm lặt

* Review các khóa học Deep Learning: https://www.kdnuggets.com/2017/10/3-popular-courses-deep-learning.html

Mở đầu

(01/11/2017) Không biết mình có phải làm nghiên cứu không nữa? Vừa kiêm phát triển, vừa đọc paper mỗi ngày. Thôi, cứ (miễn cưỡng) cho là nghiên cứu viên đi.

Chương 35

Nghề lập trình

Chân kinh con đường lập trình: [Teach Yourself Programming in Ten Years. Peter Norvig](http://norvig.com/21-days.html)

Trang web hữu ích

* Chia sẻ thú vị: [15 năm lập trình ở Việt Nam](https://vozforums.com/showthread.php?t=3431312) của Blanic (vozfourm) * Trang web chứa cheatsheet so sánh các ngôn ngữ lập trình và công nghệ http://hyperpolyglot.org/

01/11/2017

Vậy là đã vào nghề (đi làm full time trả lương) được 3 năm rưỡi rồi. Thời gian trôi qua nhanh như *ó chạy ngoài đồng thật. Tâm đắc nhất với câu trong một quyển gì đó của anh lead HR google. Có 4 level của nghề nghiệp. 1 là thỏa mãn được yêu cầu cả bản. 2 là dự đoán được tương lai. 3 là cá nhân hóa (ý nói là tận tình với các khách hàng). 4 là phiêu diêu tự tại. Hay thật! Bao giờ mới được vào mức 4 đây.

Chương 36

Latex

15/12/2017:

Hôm nay tự nhiên nổi hứng vẽ hình trên latex. Thấy blog này là một guide line khá tốt về viết blog phần mềm. Quyết định cài latex

Theo [hướng dẫn này](http://milq.github.io/install-latex-ubuntu-debian/)

"' sudo apt-get install texlive-full sudo apt-get install texmaker "'

Tìm được ngay bên này https://www.overleaf.com/ có vẻ rất hay luôn

Hướng dẫn cực kì cơ bản http://www.math.uni-leipzig.de/ hellmund/LaTeX/pgf-tut.pdf

Chương trình đầu tiên, vẽ diagram cho LanguageFlow

```
\documentclass[border=10pt]{standalone}
\usepackage{verbatim}
\begin{comment}
\end{comment}
\usepackage{tikz}
\begin{document}
\begin{tikzpicture}
    \node[draw] (model) at (0, 0) {Model Folder};
    \node[draw] (analyze) at (6, 0) {Analyze Folder};
    \node[draw] (board) at (3,2) {Board};
    \node[draw] (logger) at (3, -2) {Logger};

    \path[->, densely dotted] (board.east)
        edge [out=0, in=90]
        node[fill=white, pos=.5] {\tiny (1) init}
        (analyze.north) ;
    \path[->, densely dotted] (board.south)
        edge [out=-90, in=180]
        node[fill=white, pos=.3] {\tiny (2) serve}
        (analyze.west) ;
        \path[->, densely dotted] (logger.west)
        edge [out=180, in=-90]
        node[fill=white, pos=.7] {\tiny (1) read}
        (model.south) ;
        \path[->, densely dotted] (logger.east)
```

```
        edge [out=0, in=-90]
        node[fill=white, pos=.7] {\tiny (2) write}
        (analyze.south) ;
\end{tikzpicture}
\end{document}
```

Doc! Doc! Doc! https://en.wikibooks.org/wiki/LaTeX/PGF/TikZ

Chương 37

Chào hàng

16/01/2018 Bố khỉ. Hôm nay gửi lời mời kết bạn đến một thằng làm research về speech mà nó "chửi" mình không biết pitch. Tổ sư. Tuy nhiên, nó cũng dạy mình một bài học hay về pitch.

Chửi nó là vậy nhưng lần sau sẽ phải đầu tư nhiều hơn cho các lời pitch.

Vẫn không ưa Huyền Chíp như ngày nào, nhưng [bài này](https://www.facebook.com/notes/huyen-chip/k

Tóm lại skill này có 4 phần

1. Ngôn ngữ không trau chuốt 2. Giới thiệu bản thân không tốt 3. Không chỉ ra cho người nhận rằng họ sẽ được gì 4. Không có phương án hành động

Đối với email, thì cần triển khai thế này

* [Chào hỏi] * [Giới thiệu bản thân một cách nào đó để người đọc quan tâm đến bạn] * [Giải thích lý do bạn biết đến người này và bạn ấn tượng thế nào với họ – ai cũng thích được nghe khen] * [Bạn muốn gì từ người đó và họ sẽ được gì từ việc này] * [Kết thúc]

Chương 38

Phát triển phần mềm

* Phát triển phần mềm là một việc đau khổ. Từ việc quản lý code và version, packing, documentation. Dưới đây là lượm lặt những nguyên tắc cơ bản của mình.

Quản lý phiên bản

Việc đánh số phiên bản các thay đổi của phần mềm khi có hàm được thêm, lỗi được sửa, hay các phiên bản tiền phát hành cần thống nhất theo chuẩn của [semversion]. Điều này giúp nhóm có thể tương tác dễ hơn với người dùng cuối.

![](https://raw.githubusercontent.com/magizbox/magizbox/master/wordpress/phat$_{t}rien_{p}han_{m}em/versi$

Đánh số phiên bản

Phiên bản được đánh theo chuẩn của [semversion](https://semver.org/).

* Mỗi khi một bug được sửa, phiên bản sẽ tăng lên một patch * Mỗi khi có một hàm mới được thêm, phiên bản sẽ tăng lên một patch. * Khi một phiên bản mới được phát hành, phiên bản sẽ tăng lên một minor. * Trước khi phát hành, bắt đầu với x.y.z-rc, x.y.z-rc.1, x.y.z-rc.2. Cuối cùng mới là x.y.z * Mỗi khi phiên bản rc lỗi, khi public lại, đặt phiên bản alpha x.y.z-alpha.t (một phương án tốt hơn là cài đặt thông qua github)

Đánh số phiên bản trên git

Ở nhánh develop, mỗi lần merge sẽ được đánh version theo PATCH, thể hiện một bug được sửa hoặc một thay đổi của hàm

Ở nhánh master, mỗi lần release sẽ được thêm các chỉ như x.y1.0-rc, x.y1.0-rc.1, x.y1.0-rc, x.y1.0

Vẫn còn lăn tăn:

* Hiện tại theo workflow này thì chưa cần sử dụng alpha, beta (chắc là khi đó đã có lượt người sử dụng mới cần đến những phiên bản như thế này)

Tải phần mềm lên pypi

Làm theo hướng dẫn [tại đây](http://peterdowns.com/posts/first-time-with-pypi.html)

1. Cấu hình file '.pypirc' 2. Upload lên pypi

"' python setup.py sdist upload -r pypi "'

Chương 39

Phương pháp làm việc

Xây dựng phương pháp làm việc là một điều không đơn giản. Với kinh nghiệm 3 năm làm việc, trải qua 2 project. Mà vẫn chưa produce được sản phẩm cho khách hàng. Thiết nghĩ mình nên viết phương pháp làm việc ra để xem xét lại. Có lẽ sẽ có ích cho mọi người.

Làm sao để làm việc hiệu quả, hay xây dựng phương pháp làm việc hữu ích? Câu trả lời ngắn gọn là "Một công cụ không bao giờ đủ".

<!–more–>

Nội dung

1. [Làm sao để đánh giá công việc trong khoảng thời gian dài hạn?](section1) 2. [Làm sao để quản lý project?](section2) 3. [Làm sao để công việc trôi chảy?](section3) 4. [Làm sao để xem xét lại quá trình làm việc?](section4)

<p id="section1">nbsp;</p>

Làm sao để đánh giá công việc trong khoảng thời gian dài hạn?

Câu trả lời OKR (Objectives and Key Results)

!\[\](https://image.slidesharecdn.com/20170829-ale2017-okr-170829060043/95/agile-leadership-and-goal-management-with-objectives-key-results-okrs-ale-2017-prague-23-638.jpg?cb=1503986541) *OKR Framework*

Đầu mỗi quý , nên dành vài ngày cho việc xây dựng mục tiêu và những kết quả quan trọng cho quý tới. Cũng như review lại kết quả quý trước.

Bước 1: Xây dựng mục tiêu cá nhân (Objectives)

Bước 2: Xây dựng các Key Results cho mục tiêu này

Bước 3: Lên kế hoạch để hiện thực hóa các Key Results

<p id="section2">nbsp;</p>

Làm sao để quản lý một project

Meistertask

!\[\](https://focus.meisterlabs.com/wp-content/uploads/2015/03/Projectboard$_{M}T_{E}N- 1.png) * MeisterTask *$

<p id="section3">nbsp;</p>

Làm sao để công việc trôi chảy?

Có vẻ trello là công cụ thích hợp

Bước 1: Tạo một team với một cái tên thật ấn tượng (của mình là Strong Coder)

Trong phần Description của team, nên viết Objectives and Key Results của quý này

Sau đây là một ví dụ

"' Objectives and Key Results

-> Build Vietnamese Sentiment Analysis -> Develop underthesea -> Deep Learning Book "'

Bước 2: Đầu mỗi tuần, tạo một board với tên là thời gian ứng với tuần đó (của mình là '2017 | Fight 02 (11/12 - 16/12)')

Board này sẽ gồm 5 mục: "TODO", "PROGRESSING", "Early Fight", "Late Fight", "HABBIT", được lấy cảm hứng từ Kanban Board

![](https://mktgcdn.leankit.com/uploads/images/general/$_xLarge/kanban_guide_print_KPO_bleed_board2.jpg$
$TrelloBoardexample*$

* Mỗi khi không có việc gì làm, xem xét card trong "TODO" * [FOCUS] tập trung làm việc trong "PROGRESSING" * Xem xét lại thói quen làm việc với "HABBIT"

Một Card cho Trello cần có

* Tên công việc (title) * Độ quan trọng (thể hiện ở label xanh (chưa quan trọng), vàng (bình thường), đỏ (quan trọng)) * Hạn chót của công việc (due date)

Sắp xếp TODO theo thứ tự độ quan trọng và Due date

<p id="section4">nbsp;</p>

Làm sao để xem xét lại quá trình làm việc?

Nhật lý làm việc hàng tuần . Việc này lên được thực hiện vào đầu tuần . Có 3 nội dung quan trọng trong nhật ký làm việc (ngoài gió mây trăng cảm xúc, quan hệ với đồng nghiệp...)

* Kết quả công việc tuần này * Những công việc chưa làm? Lý do tại sao chưa hoàn thành? * Dự định cho tuần tới

Đang nghiên cứu

Làm sao để lưu lại các ý tưởng, công việc cần làm?: Dùng chức năng checklist của card trong meister. Khi có ý tưởng mới, sẽ thêm một mục trong checklist

Làm sao để tập trung vào công việc quan trọng?: Dùng chức năng tag của meister, mỗi một công việc sẽ được đánh sao (với các mức 5 sao, 3 sao, 1 sao), thể hiện mức độ quan trọng của công việc. Mỗi một sprint nên chỉ tập trung vào 10 star, một product backlog chỉ nên có 30 star.

Tài liệu của dự án: Sử dụng Google Drive, tài liệu mô tả dự án sẽ được link vào card tương ứng trong meister.

Tài liệu tham khảo

Goodfellow, Ian / Bengio, Yoshua / Courville, Aaron (2016): *Deep Learning.* , MIT Press.

Hai, Do (2018): *Một số phân phối phổ biến*
.

Chỉ mục