COMPOSITION NOTEBOOK

DATA STRUCTURE

100 sheets · 200 pages
6.32" x 8.17" in

dsoop

COOL

SUPER

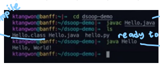
Lesson 1 Hello, world

* every Java programe has to live inside a class

```
public class File name {        must have inorder to run and call other def.?
                                  ↙ The entry point
    public static void main(string[] args) {      //psvm
    or ~~~~f
        System.out.println("Hello, World!");      //sout
    }
}
```

How it works

File.java —compile/javac→ File.class —run/java→ Result



out put of java compile
→ what .java that we wanna compile
ready to run
→ would like to run a class called Hello

// comment

/* block comment

Phython to Java in 45 mins

Ex: C° → F°

```
public class BasicTempConvert {

    public static void main(String args[]) {

        double tC = 32.75;

        double tF = 32.0 + tC * 9.0/5.0;

        System.out.printf(tF);

    }

}
```

Type Declaration

- need to declare type of a varieble fist
- a variable can't be replaced

```
int x = 5;
int a;
a = 28;
```

byte short int long		// integers
char		// character
boolean		// true / false Boolean
float double		// floating-point numb

System.out.printf("~~ %d ~~", variable)

↳ ~~ variable value ~~

S————p~ln (byte. MIN_VALUE) — check min or max — MAX

Static typing

- can never change type of variables
- Expressions have a type

Ex: int x = max(1,3) + "3" won't work

Math Operators

% remainder 22 % 6 = 4

/ division int division 8/5 = 1 double division 8.0/5.0 = 1.6 mixed division 8.0/5 = 1.6

Mixed mode operations

int a;	int a,b;	type casting either the numerator or denominator
double b, c;	double c;	c = (double) b/a
a = 3;	b = 21;	c = 21.0/5
b = 5.1;	a = 5;	c = 4.2
c = a+b;	c = b/a; → 4.0	but c = (double)(b/4) is 4.0 ← done first

order of Operations : PEMDAS () a^n (* or ÷) (+ or −) (L→R)

Functions (technically, static methods)

public static void name(String arg[])

make this visible outside of the class, without it, main won't be visible as an entry pt to ur program

Static vs. standard methods

- Static methods or class methods : extent of our understanding, our beloved function
- (standard) methods : **Next Week**

indicates the return type of the function. void says the fuction doesn't return anything useful (None in python)

a fuction can return an int, a double, etc

Function name & argumemts (Camel-case naming)

parameters are in parens : typeName varName and connect with ,

public static int add (int a, int b, int c) Take 3 parameters & return int

return is inside a function like python don't forget ;

Conditionals

```
if (<boolean expression>){
    ...
}
```

Curly Braces

multiple statements in response to a single condition.

```
int x = 5;
if (x < 10) {
    System.out.println("I shall in——");
    x = x + 10;
}
System.out.println(x);
```

- statements are grouped by braces, not by indentation.

Else, Else if

```
int cat = 20;
if (cat >= 50) {
    System.out.println(" ~~")
} else if (cat >= 10) {
    system.out.println("~~")
} else {
    System.out.println("~~")
}
```

While loops

```
int bottles = 5;
while (bottles > 0) {
    System.out.println(bottles + "bottles of soda on the wall.");
    bottles = bottles - 1;
}
System.out.println("Done.")
```

Arrays

like lists in python but have a fixed size that need to be declared first and can't change.

```
# python                         // Java
numb = [3,1,4,2,8]               int[] numb = new int[]{3,1,4,2,8};
print(numb[2])  # 4              System.out.println(numb[2]);
```

Length

System.out.println(numb.length); ← for arrays

numb.length() → for string

For loops

```
int[] numb = new int[] {2,1,4,2,8};
int total = 0;
for (int num : numb) {        → for i in numb:
    total += num;
    System.out.println("---" + num);   // num assigned i
}
System.out.println(total);
```

Break and Continue

continue	skips the rest of the current iteration of the loop (jump to the top)
break	terminates the inner most loop.

While

```
int index = 0;
while (index < numb.length) {
    System.out.println("--- " + numbers[index]);
    index++;
}
```

Lesson 2 Futher Java Basics

i++ → i = i+1

C-style for

```
for (init; condition; update){
    ~~~
}

for (int i=0; i < numbers.length; index++) {
    ~~~
}
```

Primitive Arrays

```
int[] numbers = {
```

A quick primer on strings

- String is similar to list
- Strings are immutable

2 features for now

- Fetch the i-th character of a string
- Select a substring of a string

```
String st = "Hello,world!";
// use .charAt(i) to get the i-th elements.
system.out.println("Index" + 2 + "has" + st.charAt(2));
// use .substring to make a substring
String shortStr = st.substring(2,5);
System.out.println("shortStr:" + shortStr);
```

* method .charAt(index) returns a char, not a string

Type casting

① Widening Casting (Auto) small → large

byte → short → char → int → long → float → double

```
int numb = 5;   → 5
double numbd = numb;   → 5.0
```

② Narrowing casting large → small

double → float → long → int → char → short → byte

```
double numbd = 5;
int numb = (int) numbd;
```

Char ' '

• variables that form to string

String " "

capital S (class name actully)

• messages

String to numb

```
int:    String a = "100";
        int b = Integer.parseInt(a);
double: int c = Double.parseDouble(a);
```

Int to string

```
int a = 5;
String b = String.valueOf(a);   // for #
```

Check types T|F

```
boolean result = b instanceof Integer;
```

S out → F

Scanner import from keyboard

```
import jav.util.Scanner;
{
   //  type    name    Declare class     input
   Scanner sc = new Scanner(system.in);

   String name = sc.nextLine();   // รับค่า จากแป้นพิมพ์ไปที่ name

   system.out.print("put ur name: ")   // same line no ln

   System.out.print("year: ")

   int year = sc.nextInt();
```

Increment / Decreament

++a	+1 to a b4 use
a++	use a then +1
--b	-1 to b b4 use
b--	use b then -1

Orders

()

++, --

* / %

+, -

< <= > >=

== !=

&&

||

= += -= *= /= %=

Left to Right

If, elseif, else

elseif( ); {~}

```
if (condition){

}
```

use && || !

```
&&  and   if (a>b && a>c){
          }
||  or    if (a>b || a>c){
          }
!   not   if (!(a>b)){
          }
```

Ternary Operator short form of if...else

```
varible = (condition)? it's true do this : it's false do this;
```

Switch ... case

like if but choose only 1 way to work through case.

```
int x = 0;
switch(condition){        // x

    case variable1: do what;      // 0 → yes
        break;
    ⋮
    case variableN: do what;      // 1 → No
        break;

    default: what to do when it does not match with any above cases.

}
return → yes
```

To jump off this switch need to use "break" if not it'll keep going

break out of the loop

continue stop and back to the top of the loop

```
if(i==5) break; ✓
if(i==5){
  break;   ✓
}
```

While not know rounds

```
while (condition){
    ~~;
}
```

For loop Know rounds

```
int[] variable = new int[]{ , , }
for (int i : variable){     // for i in numb:
    :
}


for (init; condition; update){
    ~~
}
```

Do-While do 1 first and consider the condition b4 redo it.

run 1 time and then see the while condition
if it's True, then it'll redo again

```
do{
    when it's true;
}while (condition);
```

Non-primitive Data Types

Arrays like lists

```
type[] name = new type[]{..,...,..};

type[] name = {..,..,..};
```

Empty arrays `type[] name = new type[size];`

length `name.length`

obtain a string representation of arrays `Arrays.toString(Array's name)`

access elements

```
pick     name[index]

domath   name[index]++;
         name[index] += #;
change   name[index] = new;
```

access elements by for loop

```
for (int i = 0; i< name.length; i++){
    ~~
}
```

access elements by for each

```
for( type i : name ){     // for i in name
    ~~
}
```

copy

```
import java.util.Arrays;
type[] name = Arrays.copyOf(thatArrayName);
```

convert

```
Arrays.toString(name)

type new = name.toString()
```

String

pick i[th] char `name.charAt(i)`

substring `name.substring(index, index)`

int to string `type[] new = String.valueOf(name);`

Big number! turn any # type to a BigInteger

```
import java.math.BigInteger;

BigInteger name = BigInteger.valueOf(value);
```

multipy 2 BigIntegers

```
name.multiply(BigInteger.valueOf(#))
```

S3: Class

- is a template that defines the form of an obj.
- object are instance of a class
- a class is a set of plans that specify how to build an object.

Anatomy of a class: instance variable & methods or either one.

```
Class Classname {
    // declare instance variables
    type var1;
    type var2;
    //...
    typ varN;
    // declare methods
    type method1(parameters) {
    }
    //...
    type methodN(parameters) {
    }
}
```

- In general, a class doesn't require a main method. It's needed only when the class is the starting pt of ur program.
- Program can have multiple class.

Defining a Class

Ex:
```
class Vehicle {
    int passengers;
    int fuelCap;
    int mpg;
}
```
no need public static void main here cus we define the class, but use main in the other file that wanna declare an obj of type Vehicle.

have 3 members

After creating an object, need to write new Vehicle()

```
Vehicle minivan = new Vehicle(); // a vehical obj called minivan.
```

Then minivan becomes an instance of Vehicle.

Every Vehicle obj contain its own copies of passengers, fuelCap, mpg.

Object Name.member name = ww;
use dot (.)

```
// This should go into Vehicle.java
public class Vehicle {
    int passengers;  // number of passengers
    int fuelCap;     // fuel capacity in gallons
    int mpg;         // fuel consumption in miles per gallon
}

// This class declares an object of type Vehicle.
public class VehicleDemo {
    public static void main(String args[]) {
        Vehicle minivan = new Vehicle();
        int range;
        // assign values to fields in minivan
        minivan.passengers = 7;
        minivan.fuelCap = 16;
        minivan.mpg = 21;
        // compute the range assuming a full tank of gas
        range = minivan.fuelCap * minivan.mpg;
        System.out.println("Minivan can carry " + minivan.passengers +
                           " with a range of " + range);
    }
}
```
IMPORTANT!

name of a public class = File name!

the dot operator can access the methods.

S3: References

- is a piece of data that points where another data is.

3 key Principles

1. Everything we store is a sequence of bits (encoded appropriately). 10010000 11 01 ~
2. Primitive vs. Reference types.
   - Java distinguishes these types by how it's stored.

Primitive: boolean byte char short int long float double (lower-case type)

Reference: arrays, class, ..., String

- For primitive type, variables store the actual data.
- For reference type, the variables store the location of the actual data. (storing pointer/ref)

Ex:
```
Class Feline {
    int w;
}
Feline a = new Feline();
a.w = 5;
Feline b = a;
b.w = 9; // so b.w=9 and a.w=9
```
a: , b: → w:

Ex:
```
int x = 5;
int y = x;
y = 9; // now y=9 and x=5
```
x: 101...001, y: 101...001

check bits

Ex:
```
String x = "He" + "llo"
String y = "Hel" + "lo"
```
x != y

name2.equal(name)

x == y ✓

3. The Rule of Equal (RoE)

y = x : copy all the bits from x to y

- if x,y are a primitive type, the actual data is copied.
- if x,y are a reference type, the reference (bits) is copied.

S3 2D Arrays

Full nxn Grid

```
int [][] grid = new int[n][n]

for (int i ~
    for (int j ~
        grid [i][j]= ~
```

uneven Grid

```
int[][] tri = new int[] // n row of int[]

for (int i ~
    tri[i]= new int[ ];
    for (int j ~
        tri [i][j]= ~
```

Initializing Nested Arrays

```
int[][] name = new int[][]{
    new int[] {~},
    new int[] {~},
    new int[] {~}
};
```

or

```
int[][] name = new int[][]{
    {~},
    {~},
    {~}
};
```

```
Int Node next;
public Int Node (int head, Int Node next){
    this.head = head;
    this.next = next;
}
// return size of list
public int size(){ //recursive
    if (this.next == null) // we're by ourselves
        return 1
    else
        return 1 + tis.next.size(); // the size of remainder
```

this.head
this.next
the box

```
public String toString(){
    if (this.next == null)
        return Integer.toString(this.head);
    return Integer.toString(this.head) + "," + this.next.toString();
// this.head + ", " + (the rest of the list)
```

Philosophizing: Fixed Block vs. Chain

Options:

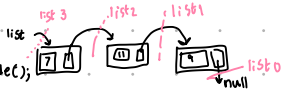
Fixed block - consecutive memory (Array List) | 2 | 1 | 5 | 7 |

Chain of items (LinkedList)

To ponder:

- Add → easy for chain, hard for block
- Access → easy for block, hard for chain
- Remove → easy for chain, hard for block

Build a list [7,11,9]




```
IntNod list= new IntNode();
list.head = 7;          [7] → null
list.next = new IntNode(); [7] → [ ]
list.next.head = 11;    [7] → [11] → null
list.next.next = new IntNode();
list.next.next.head = 9; [7] → [11] → [9] → null
```

backward

```
Int Node list0 = null;
Int Node list1 = new IntNode(head:9, list0);
Int Node list2 = new IntNode(head:11, list1);
Int Node list3 = new IntNode(head:7, list2);
```

Walking a list

- public String toString() returns the content of list
- public int size() returns the length of the list.

list2.toString() → "11, 9"
list2.size() → 2
list1.size() → 1

Recursive view

A list is either
- storing a single #: 7 | null
- a # followed by a list: 5 | →

add delta to all list member

```
public IntNode incrList(int delta) {
    if (this.next == null)
        return new IntNode(this.head+delta, null);
    return new IntNode(this.head+delta, this.next.incrList(delta));
}
```

Iterative View

we need: current location

stopping condition

logic to move to the next node.

```
public int iterSize() {
    IntNode current = this;
    int mySize = 0;
    while (current != null) {
        mySize ++;
        current = current.next;
    }
    return mySize;
}
```

call by name.iterSize()

Overloading and Static

Method Overloading

- In the same class, methods can have the same name, Java distinguish using signatures.
- same method name with diff data types.

it's ok to overload name as they have diff signature,

calculate(double,int, double)

FooBar(int,double)

variable names won't impact

Constructor Overloading

For assigning variables

method: private public static <T> void name(T variable){ ...}

<T> name(T variable){...} // construct or

<T> name(T variable){...} // reg

Static Methods and Variables

class: class A <T,U>

class C<T> implement I<T> {...}

- a class member that independent, no need to used through an obj.
- need to declare static keyword.
- it can be accessed even b4 any obj in class are created and w/o ref to any obj.

static variable

instance variable

- If define outside the class: Timer.count

or inside is fine but can't Time.instVar1 cus Java does n't know what obj we are talking about

- all obj of class share same count

- called through its class name w/o any obj . can't access this
- Static methods/variables are properties of a class, not specific obj
- methods/variables are called with respect to an obj.

Using .copy()

```
public IntNode copy(){
    if (this.next == null)
        return new IntNode(this.head, next: null);
    return new IntNode(this.head,this.next.copy());
}
```

```
class SLList {
    private IntNode first;
    public SLList() { first = null;} → ?
                                       first → null
    void addFirst(int x){
        // to stick x to the front of the list
        first = new IntNode(x,first);
    }
    int getFirst(){
        // return the element at the front of the list.
        return first.head;
    }
```

Int Node

sentinel is assign as 0th element on the list cus we don't want it to be null but sen.rest = null

we won't use

IntNode sentinel = new IntNode(any #, null);

addFirst: sentinel.rest = new IntNode(x, sentinel.rest)

go through: IntNode p = sentinel.rest; → start w/ real 1st

while(p != null) { ... ; p = p.rest }

Generic Lists

Type parameter — so later u can call T as SLList type

```
// the type T is a parameter, allowing us to defer type selection
class SLList<T> {

    private class Node {
        T head;  // use T instead of int, so head stores data of type T
        Node rest;

        public Node(T h, Node r) { this.head = h; this.rest = r; }
    }
    // the front of the list, so users doesn't have to update this themselves.
    // remember: IntNode was the class we wrote previously.
    private Node first;

    // two constructors to make initializing a list easier
    public SLList() { first = null; }
    public SLList(T x) { first = new Node(x, null); }


    // so that users don't have to manage the inner bookkeeping themselves
    public void addFirst(T x) {
        first = new Node(x, first);
    }

    public T getFirst() {
        return first.head;
    }
}
```

desired type

SLList < Double > list1 = new SLList();

int -> Integer	double -> Double	char -> Character	
boolean -> Boolean	long -> Long	byte -> Byte	
short -> Short	float -> Float		

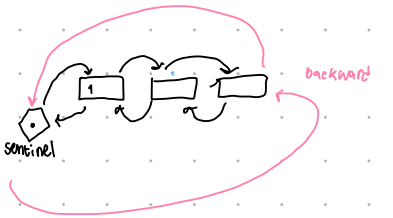
Stacks & Queues

- linear data structures
- Flexible size
- the diff is how elements are removed

A Stack allows elements to be inserted (push) into and deleted (pop) from the top

LIFO (Last-in, first-out)

A queue allows elements to be inserted (enqueue) into one side (rear) of the list, and deleteed (dequeue) from the front. FIFO (First-in, First-out)



A list → conservative memory (prim. array) → jump to any index easily

↳ ArrayList<E> → built-in array-based resizable list (doubling trick is used)

↳ LinkedList<E> → a built-in doubly-linked list

Polymorphism an obj can assume many different forms

list: a linked list, an array list

Java Interfaces

public interface Series {

Real-World Example

A List<E> is an interface (contract/specification)

An ArrayList<E> is a class satisfy the List<E> interface.

A is a class satisfy the List<E> interface.

```
Class A {
    public A(double secretF) { F = secretF;}   constructor w/ one parameter
    public A() { this(1502.0);}                constructor w/ 0 parameter
}
```

List<Integer> list1 = new ArrayList<>(); ✓

Interface — a class kind that will be implemented

- only has method signature, no body
- all methods will be implemented
- every method will be public, either write or not.

Vehicle.java

```
public interface Vehicle {

    int cashRate();

    int efassRate();

}
```

Sedan.Java

```
public class Sedan implements Vehicle {

    public int cashRate() {
        return 40;
    }
    public int efassRate() {
        return 35;
    }
}
```

main

```
public static void main(String[] args){

    Sedan s = new Sedan();
    System.out.println(s.cashRate());

    Vehicle v = new Sedan();
    System.out.println(v.efassRate());
}
```

(or use interface as type; implement; class name)

- use **implements** for interface
- use **extends** in other cases

Inheritance

- a class that allows the other class to link method or properties.

Person.Java → superclass / parent class

```
public class Person {
    private String firstName, lastName;
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
}
```

Student.Java → derived class

```
public class Student extends Person {
    property in Student class only
    private String major;
    public void setMajor(String major){
        this.major = major;
    }
}
```

not allow to inherihetance: Person + Student

```
public final class ExchangeStudent extends Student {
    private String partnerUni;
    public void setPartnerUni(String name) {
        this.partnerUni = name;
    }
    ⋮
}
```

main

```
public static void main(String[] args){
    Student S1 = new Student();
    S1.setFirstName("Peter");
}
```

if public abstract class Person → it can't `Person P1 = new Person()`

(abstract: allow other class to inherence, can't directly create an instant of the class)

Tips About Inheritance and Interfaces

- When a class B extends from a class A (class B extends A)
  - we can't just call a private instance of A in B
- If we recreate same method name from A in B write @Override above the new method

use super()

- can be use only inside the method of a child class calling a parent class
- change in B, which extents A

1. calling parent method `super.method();` when writing @Override
2. use in the same class method + on the first line. Ex: Animal method
   - use `super();` to call public method(){} in A, there is no need to write super(); // when it's in
   - use `super(parameter);` specify what parameter is called

```
A: Animal(String age, String name) { this.age = age; this.name = name;}
B: Animal(String age, String name, String food) { super(age, name); this.food = food;}
```

(super(age, name): exist already; this.food = food: just now)

Boxing and Unboxing

- Reference type is a wrapper type of primitive type

Auto box: int 42 primitive → Integer number class

```
Integer myInt = new Integer(42);
Integer myInt = 42;
```

Unbox: Integer value:42 ⟶ 42 int primitive

Java primitive types	Java Number classes
byte b = 10;	Byte b = new Byte(10);
short	Short
int	Integer
long	Long
float	Float
double	Double

Unboxing Null will lead to an error if the method expect other signature.

Boxed Obj aren't Readily Computable

- can't perform on the wrapped obj, so it needs to auto unwrape first

```
Final int N = 1_000_000_000;
for (Integer iob = 0; iob < N; iob++) {}
```

unbox → increment → unbox → throw original box away

or

```
for (int i=0; i<N; i++) {}
```

→ run faster

Generics and Type Erasure

- type parameters
- every ref type = generic parameter
- for primitive, we have type wrapper

ex: `List<Integer> list = new ArrayList<>();`

Ex: `class A<T>`

`class B<Key, Value, Time>`

`interface I<T>`

`class C<T> implements I<T>{ }`

Diamond Operato <>

ex: `List<Integer> = new ArrayList<>()` (ArrayList <Integer>) safe time

Generics in Methods and Constructors

```
class MyClass<T> {
    <ItemT> MyClass(ItemT arg) {...} // constructor
    static <X> staticMethod(X arg) {...} // static method
    <X> regularMethod(X arg) {...} // regular method
}
```

Bonded Types

- limit type that can be passed to a type parameter.

Ex: want only numbers, not any <T>, write <T extends superClass>

now T must be a superClass.

(number is a super class)

```
class NumberFun<T extends Number> {
    T number;
    NumberFun(T num) { number = num;}
    double getFractional() {
        return number.doubleValue() - number.intValue();
    }
}
```

Wildcards

extend NumberFun w/ a method integralEqual(that) → compare integral part of this & that

```
boolean integralEqual(NumberFun<T> that){
    return this.number.intValue() == that.number.intValue();
}
```

(intValue(): fix to int anyway)

- can compare only same type of T not <Integer> vs. <Double>

(anytype vs anytype)

(NumberFun<?> that)

(NumberFun<?> extends SuperClass)

Type Erasure

?

L7: Higher-Order Functions and Subtyping

```
int maxIndex(int[] items) {
    if (items.length == 0)
        return -1;
    int maxIndex = 0;
    for (int index=0;index<items.length;index++) {
        if (items[index] > items[maxIndex])
            maxIndex = index;
    }
    return maxIndex;
}
```

We can't use it w/ the other types

ex: Cat[]

```
public class Cat {
    private String name;
    private int weight;
    private int age;

    public Cat(String name, int weight, int age) {
        this.name = name;
        this.weight = weight;
        this.age = age;
    }

    public String getName() { return name; }
    public int getWeight() { return weight; }
    public int getAge() { return age; }
}
```

Unclear to compare weight or name appearing

There are 2 ideas for expressing how two obj are to be compared

① Explicit higher-order comparison function — HoF — user make the choice

- pass in a comparison function to the function directly — compare(x,y)
- can work w/ multiple func. for the same class
- More control and explicitness

Prior to Java 8

- Write our own interface that defines any function that take int and return int

```
public interface IntUnaryFunction { int apply(int x); }
```

must have a public method takes int return int

```
class DoubleFunction implements IntUnaryFunction {
    public int apply(int x) { return 2*x; }
}
```

so now DoubleFunction is IntUnaryFunction already

ex: that use ↑

require any variable that is in

```
int twice(IntUnaryFunction f, int x) {
    return f.apply(f.apply(x));
}
```

f(f(x))

```
sout(twice(new DoubleFunction(), 5));
```

```
def tenX(x):
    return 10*x
def do_twice(f,x):
    return f(f(x))
print(do_twice(tenX,2))
```

f(x) = f(2) = 20
f(f(x)) = f(20) = 200
→ 200

Java 8 and On

- reference types to refer to functions.

```
import java.util.function.*;
class HoFDemoJava8 {
    static int doubleFunc(int x) { return 2*x; }
    static int twice(Function<Integer, Integer> f, int x) {
        return f.apply(f.apply(x));
    }

    public static void main(String[] args) {
        int result = twice(HoFDemoJava8::doubleFunc, 5);
        System.out.println(result);
    }
}
```

create a func. that pass into another func.

call

int→return int like

Function<T,T>

a function that take T return

yields

reference to this func.

ClassName :: methodname

Write maxIndex using HoF

```
import java.util.function.BiFunction;

public class maxIndex {
    public static void main(String[] args) {
        double[] a = new double[]{1.2,6.7,33.3,4.3,19.3};
    }
    static <T> int maxIndex(T[] items,
                            BiFunction<T, T, Boolean> isLarger){ //takes2inputs:T,T. return Boolean
        if(items.length==0){return -1;}
        int maxIndex = 0;
        for(int i = 0;i<items.length;i++){
            if(isLarger.apply(items[i],items[maxIndex]))
                maxIndex = i;
        }
        return maxIndex;
    }
}
```

replace item at index > item at maxIndex

BiFunction<T,T,Boolean> take T,T return Boolean

```
1  class MaxDemo {
2      static boolean isLargerByWeight(Cat x, Cat y) {
3          return x.getWeight() > y.getWeight();
4      }
5      public static void main(String[] args) {
6          Cat[] items = ... // list of cats omitted
7          int sampleRet;
8          sampleRet = maxIndex(items, Maximum::isLargerByWeight);
9          sampleRet = maxIndex(
10             items,
11             // OR: declare a function in place
12             (Cat x, Cat y) -> x.getWeight() > y.getWeight()
13         );
14     }
15 }
```

② Subtyping polymorphism comparison — provide a default

- the obj that we're working w/ have certain methods that allow for comparing them — x.largerThan(y)
- Class implementation makes the choice
- Grounded on a default
- More concise code as long as the default makes sense

Ex: For cats, compairing weight,

```
if (items[index].isLargerThan(items[maxIndex]))
```

How we know that the item obj has a .isLargerThan? we needs it in the method

Steps

- define an interface that promises such a method
- tell Java that you only want items of that kind

promise the method isLargerThan take <T> return boolean

```
public interface HasIsLarger<T> {
    boolean isLargerThan(T that);
}
```

Write maxIndex to accept only items conformed to this interface

```
static<T extends HasIsLarger<T>> int maxIndex(T items[]) {
    if (items.length == 0)
        return -1;

    int maxIndex = 0;
    for (int index=0;index<items.length;index++) {
        if (items[index].isLargerThan(items[maxIndex]))
            maxIndex = index;
    }

    return maxIndex;
}
```

How to update Cat class to perform HasIsLarger

declare that class implement the interface + implements the promised method

```
public class Cat implements HasIsLarger<Cat> {
    // existing code omitted
    public boolean isLargerThan(Cat that) {
        return this.weight > that.weight;
    }
}
```

Build Tools

automate the compiling source files → runnable (binary) code

Wishlist

- hit one button → all files (diff directories) are compiled in the right order
- Same button downloads the necessary external libraries (internet) automatically.
- another button to "Run" the program
- another button to run our code through a battery of tests

Main build tools

Apache Ant:

Apache Maven:

Gradle: we'll use this on

gradle wrapper at the start of the project (provided)

```
plugins {
    id 'java'
    id 'application'
}

sourceCompatibility = '1.11'

application {
    mainClass = 'Main'
}
```

'java' → what language

build.gradle

'Main' → run class

Words

build — compile everything + download external dependencies (if needed)

clean — delete compiled files (clean up the folder)

run — run the application (if a main class is specified)

hello-gradle > src > main > java

src

build.gradle

main

unit test-demo

build.gradle

src

main

java ← src files

test

java ← tests

Test-Driven Development(TDD)

- write test for code before coding

1. Identify a feature
2. Write tests
3. Run tests. Should fail
4. Write code that passes the tests Unit.
5. Refactor & rerun the tests to make sure things go well, clean

build.gradle

```
plugins {
    id 'java'
    id 'application'
}

application {
    mainClass = 'Main'
}
repositories {
    mavenCentral()
}

dependencies {
    testImplementation('org.junit.jupiter:junit-jupiter:5.8.1') // junit version
}

test {
    useJUnitPlatform()
}
```

src/main/java/PrefixSum

```
public class PrefixSum {
    // sums
    static int sumInts(int a, int b) {
        return a+b;
    }

    // range
    static int[] prefixSum(int[] numbers) {
        return numbers;
    }
}
```

create the test folder in src/test/java

→ src/test/java/PrefixSumTest.java

```
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.*;

class PrefixSumTest {
    @Test
    void sumInts() {
        assertEquals(5, PrefixSum.sumInts(2, 3));
    }

    @Test
    void prefixSum() {
        assertArrayEquals(new int[]{1, 3, 6}, PrefixSum.prefixSum(new int[]{1, 2, 3}));
    }
}
```

expected actual

1 2 3 ✓

√ Test Results

@Override

- method name existed somewhere previously.
- We wish to override it
- prevents us from misspelling names.

Exceptions

try { // code may cause an exception

when exception is thrown, the other code below in here won't be executed

}

catch (ArithmeticException exception type e variable) { // what happens when that type occurs }

finally { // always execute whether or not caused an exception or return anything above }

Throw create or pass on exception

try { throw new exception type (); } ← "can show the text here"

catch (exceptionType e) { ... }

```
int offerMod (int x, int y){
  if (y==0)
    throw new Arithm...();
  return x/y;
```

```
void foo() {
    int numers[] = {4, 2, 7, 9};  // Length: 4
    int denoms[] = {2, 5, 0};     // Length: 3

    for (int i=0;i<numers.length;i++) {
      try { System.out.printf("%d/%d = %d\n",
            numers[i], denoms[i], numers[i]/denoms[i]);
      } catch (ArithmeticException e) {
        System.out.println("Can't divide by zero");
      } catch (ArrayIndexOutOfBoundsException e) {
        System.out.println("No matching element");
        throw e;
      }
    }
}
```

When An exception can be caught, handled, and passed on (known as re-thrown) to the outer caller.

rethrown if catch block can't handle it.

checked Exception

throws indicate that this method can throw this type of exception other than provided in Java.

```
public void setAge(int age) throws DataOutOfBoundsException {

    if (age <0 || age >120){
      throw new DataOutOfBoundsException(" "+age);
    }
    this.age = age;
}
```

```
public void makePerson() {
        Person clone = new Person();
        int random = (int)(Math.random() * 1000);
        try {
            clone.setAge(random);
        }
        catch (DataOutOfBoundsException e) {
            System.out.println(e.getMessage());
            // gets the message describing the problem
        }
    }
```

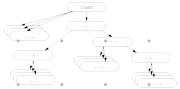
Custom Exception

```
public class DataOutOfBoundsException extends Exception {
   public DataOutOfBoundsException(String dataName){
     super("Data value " + dataName + " is out of bounds.");
   }
}
```

Show on the screen

JS: The Object

The origin of all classes.



Method in class Object :

```
String toString()
boolean equals(Object obj)
```

} frequently used

```
int hashCode()
Class<?> getClass()
protected Object clone()
protected void finalize()
void notify()
void notifyAll()
void wait()
void wait(long timeout)
void wait(long timeout, int nanos)
```

toString() Method

normally if you create an array list, Object class provides you .toString().

it's automatically return weird sign whether you call sout(a) or sout(a.toString())

```
public String toString(){
    return  getClass().getName() + '@' + Integer.toHexString(hashCode());
}
```

to fix that you can override toString() method and call a or a.toString()

now it's made from your own class

.equals() vs ==

behave differently

```
List<Integer> l1 = new ArrayList<>(Arrays.asList(2, 8, 4));
List<Integer> l2 = new ArrayList<>(Arrays.asList(2, 8, 4));
System.out.println(l1 == l2); // ==> false
System.out.println(l1.equals(l2)); // ==> true
```

== points to the same ref?

.equals() compare content

but without override .equals() in the class, it will work as == in default.

```
class ArraySet<T>
  List<T> items;

  public ArraySet(){items = new ArrayList<>();}
  public void add( value){items.add(value);}
  public boolean contains(T value){return items.contains(value);}
  public int size(){return items.size();}
}

ArraySet<Integer> set = new ArraySet<>();
ArraySet<Integer> set = new ArraySet<>();

System.out.println(set.equals(set));
```

```
@Override
public boolean equals(Object o) {
  ArraySet<T> other = (ArraySet<T>) o;
  if (this.size() != other.size()) { return false; }
  for (T item : this.items) {
    if (!other.contains(item)) { return false; }
  }
  return true;
}
```

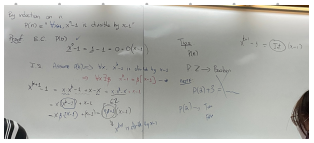
```
@Override
public boolean equals(Object o) {
    if (o==null || this.getClass()!=o.getClass()) { return false; } → prevent null and other type
    if (this == o) { return true; } //Speed up the case → ref

    ArraySet<T> other = (ArraySet<T>) o;
    if (this.size() != other.size()) { return false; }

    for (T item : this.items) {
        if (!other.contains(item)) { return false; }
    }
    return true;
}
```

L9: Inductive Thinking

Induction
- Predicate P(·) → Boolean
- base case
- inductive case

Ex: Lemma For all $n \geq 1$

$$1+2+3+\dots+n = \frac{n(n+1)}{2}$$



Pf By induction $P(n) \rightarrow 1+2+3+\dots+n == \frac{(n)(n+1)}{2}$

Basecase $P(1) \rightarrow 1 == 1$ true

Inductive case Aim: $P(k) \rightarrow P(k+1)$

WTS $P(k+1)$: $\underbrace{1+2+3+\dots+k}_{P(k)} + k+1 = \frac{(k+1)(k+2)}{2}$ RHS:

LHS: $\frac{k(k+1)}{2} + k+1 = (k+1)\left[\frac{k}{2}+1\right] = \frac{(k+1)[k+1+1]}{2}$ ✓

Inductive-inspired Algorithm

inductive proofs tend to translate to recursive algo

Ex: 

24: 5 5 7 7

25: 5 5 5 5 5

26: 5 7 7 7

27: 5 5 5 5 7

28: 7 7 7 7

29: 5+ 29

30: 5+ 25

↓ so on

```
List<Integer> change(int n){
    // key in the ans from 24-28
    if(n==24) return List.of(5,5,7,7);
    if(n==25) return List.of(5,5,5,5,5);
    if(n==26) return List.of(5,7,7,7);
    if(n==27) return List.of(5,5,5,5,7);
    if(n==28) return List.of(7,7,7,7);

    // a list of making n-5
    List<Integer> minus5 = new ArrayList<>(change(n-5));
    minus5.add(5);
    return minus5;
}
```

Ex: pow(b, w) → b^w

int int w≥0

Pf $P(n)$ = "$\forall b$, pow(b,n) → b^n"

B.C. $P(0) \rightarrow$ LHS pow(b,0) => 1 ✓ (by looking at the code)

RHS $b^0 = 1$

I.S. Assume $P(k-1) \rightarrow \forall b$, pow(b, k-1) → b^{k-1} (let k ≥ 1)

$\forall b$: pow(b,k) = pow(b, k-1) * b (code)

$= b^{k-1} * b$

$= b^k$

```
public int pow(int b,int w){
    if (w==0) return 1;
    return pow(b, w-1)*b;
}
```

if we do it the other way: faster

$b^{1024} = (b^{512})^2$ $\quad b^5 = (b^2)^2 \times b$

$b^{512} = (b^{256})^2$

1024 → 512 → 256 → ⋮ → 0

w → w/2 → w/4 → ⋮ → 1

Ex: FactHelper(n, 1) ↪ n!

Pf $P(n) \equiv$ "FactHelper(n,1) ↪ n!" (return)

B.C. $P(0) == 1$ ✓

I.S. Assume $P(n-1) \Rightarrow FH(n-1, 1) \hookrightarrow (n-1)!$ → know only when a = 1

$FH(n,1) \overset{\text{code}}{\Rightarrow} FH(n-1, n)$ → useless $a = n \neq 1$

the create new predicate

$P(n) = \forall a$, $FH(n,a) \hookrightarrow a \times n!$

FH(n,a)

→ FH(n-1, a×n)

→ FH(n-2, a×n×(n-1))

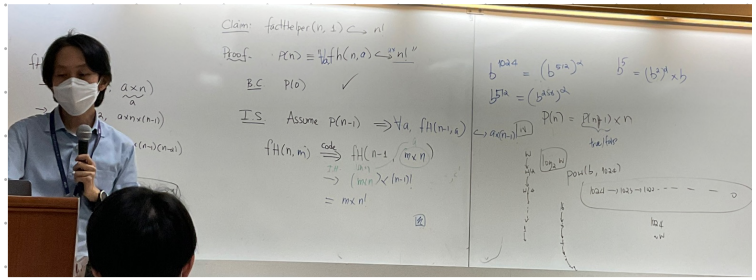
→ FH(n-3, a×n×(n-1)(n-2))

↓

→ FH(0, a×n×(n-1)×(n-2)×…×1)

I.S. Assume $P(n-1) \Rightarrow \forall a$, $FH(n-1, a) \hookrightarrow a \times (n-1)!$

$FH(n,m) \overset{\text{code}}{\Rightarrow} FH(n-1, m \times n)$ (going in) $\Rightarrow$ (unit test) $(m \times n) \times (n-1)! = m \times n!$



Strong induction

I.S.

$\forall k' < k$, $P(k') \Rightarrow pow(b,k') \rightarrow b^{k'}$

$\forall b$: pow(b,k)

Case I : even

t = pow(b, k/2)

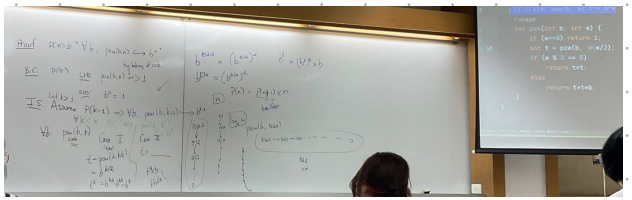
$= b^{k/2}$

$t^2 = b^{k/2} \cdot b^{k/2} = b^k$

Case II : odd

t = _____ HW

$t^2 * b$



Packages a group of classes, interfaces, other packages

→ **Reusable:** put logic/routines in a class inside a package & import the package and use the class

→ **Better organization:** if we have many classes, store to group similar types of classes in a meaningful package name.

→ **Name Conflicts:** Java won't allow us to have 2 classes same namespace.

Putting them in seperate packages will avoid name collision.

Types of packages Built-in packages, User-defined packages.

Built-in packages

Ex: `import java.util.ArrayList;`

`java` — subpackage of java; `util` — a class lives in subpackage util; `ArrayList` — top-level package

`import java.util.*;` → pour everything inside java.util

Instead of importing a package explicitly, we can refer to an entity (e.g. a class, an interface) in another package using its full name.

```
void demo() {
    java.util.List<Integer> list = new java.util.ArrayList<>();
    // proceed to use list as usual
}
```

interface — class

we use their full class/interface names, without importing them.
if class/interface only used once/twice, this is a good alternative to import it.

User-defined Packages top of the file: `package <<packageName>>;`

```
package hr;

public class Person {
    private int age;
    private String name;

    public Person(int age, String name) {
        this.age = age;
        this.name = name;
    }

    public int getAge() { return age; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
}
```

a class Person is inside the hr package

Directory Organization

Java expects classes inside a package X to be kept inside folder X

Gradle . base location .: src/main/java.

• Person belongs to the hr package, it will be kept inside a subdirectory called hr

src/main/java/hr/Person.java

Using Packaged Classes

use a class in another file

```
// Use full name
hr.Person p = new hr.Person();
```

to refer to the Person class inside hr package

or

```
import hr.Person; // top of file
Person p = new Person();
```

then we're free to refer to Person directly in this file

Package Hierarchy connect by point (.)

```
edu
└── muic
    └── hr
        ├── Department.java
        ├── Person.java
        ├── recruitment
        │   └── Recruit.java
        └── training
            └── Training.java
```

edu.muic.hr

edu.muic.hr.recruitment

	Private	Default	Protected	Public
Visible within same class	✓	✓	✓	✓
Visible within same package by subclass		✓	✓	✓
Visible within same package by non-subclass		✓	✓	✓
Visible within different package by subclass			✓	✓
Visible within different package by non-subclass				✓

Use StringBuilder to call a method that print lots of words

```
private String getLocationInfo() {
    StringBuilder info = new StringBuilder();
    info.append("north");
    info.append("\n");   // or '\n'  new line
    info.append("east");
    return info.toString();
}
```

To call and print `System.out.println(getLocationInfo());`

Iterable — iterator() required

Iterable is an interface being called by a method/class implements Iterable<String>

```
public interface Iterable<E> {
    Iterator<E> iterator();
}

public interface Iterator<T> {
    boolean hasNext();
    T next();
}
```

} state these some where b4 implement them

SLListSet → made from LinkedList

ArraySet → from ArrayList

```
import java.util.LinkedList;
import java.util.List;

7 usages
public class SLListSet<T> implements Iterable<T> {
    9 usages
    private List<T> items;
    2 usages
    public SLListSet(){items = new LinkedList<>();}
    public boolean contains(T value){return items.contains(value);}
    1 usage
    public int size(){return items.size();}
    5 usages
    public void add(T value){if(!items.contains(value)){items.add(value);}}

    2 usages
    private class SLListSetIterator implements Iterator<T> {
        4 usages
        private int itPos; //track
        2 usages
        public SLListSetIterator(){itPos = 0;}
        2 usages
        @Override
        public boolean hasNext() { return itPos < items.size(); }
        2 usages
        @Override
        public T next() {
            T toReturn = items.get(itPos);
            itPos++;
            return toReturn;
        }
    }
    1 usage
    public Iterator<T> iterator(){return new SLListSet.SLListSetIterator();}
    1 usage
    public boolean equals(SLListSet<T> that){
        Iterator<T> th = new SLListSet.SLListSetIterator();
        if(that.size()!=items.size()){return false;}
        while (th.hasNext()) {
            if(!items.contains(th.next())){return false;}
        }
        return true;
    }
    public static void main(String[] args) {
        SLListSet<Integer> a = new SLListSet<>();
        a.add(23);
        a.add(111);
        a.add(11);
        SLListSet<Integer> b = new SLListSet<>();
        b.add(23);
        b.add(11);
        System.out.println(a.equals(b));
    }
}
```

Call b4 iterating using while

```
import java.util.ArrayList;
import java.util.List;
2 usages
public class ArraySet<T> implements Iterable<T> {
    7 usages
    private List<T> items;
    1 usage
    public ArraySet(){items = new ArrayList<>();}
    public boolean contains(T value){return items.contains(value);}
    4 usages
    public void add(T value) { if(!items.contains(value)){items.add(value);} }
    public int size(){return items.size();}

    //inner class implementing Iterator
    1 usage
    private class ArraySetIterator implements Iterator<T> {
        4 usages
        private int itPos; // Track where the iterator currently is
        1 usage
        public ArraySetIterator(){itPos = 0;}
        2 usages
        @Override
        public boolean hasNext() { return itPos < items.size(); }

        2 usages
        public T next() {
            T toReturn = items.get(itPos);
            itPos++;
            return toReturn;
        }
    }
    //another inside class// call this to iterate
    1 usage
    public Iterator<T> iterator(){return  new ArraySetIterator();}

    public static void main(String[] args) {
        ArraySet<Integer> as = new ArraySet<>();
        as.add(1);
        as.add(5);
        as.add(1);
        as.add(3);
        Iterator<Integer> it = as.iterator();
        while (it.hasNext()){
            System.out.println(it.next());
        }
    }
}
```

Lecture 13: Perf Characterization I

Space & Time

7 ways (didn't write it down)

$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3)$

constant < $\log_2 n$ < n < $n \log n$ < n^2 < n^3

Empirical → Analytical

limitation
- Machine Dependent
- Cover only a few cases
- Need an implement? compare a vs. b

$F(n)$ = time taken to run on the hardest input of size n (worst case time) ... easiest input of size n (best case time)

$f(n) = 90n + 27n^2 + 17.215 n \log n + 2n^3$ worst case

linear $P(n) = 2n$

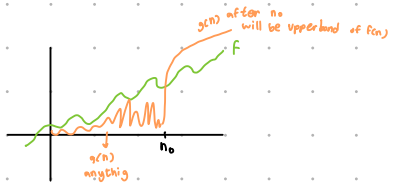
exponential $P(n) = 4^n$

cubic n^3

$x = \log_b n \iff b^x = n$

$\log_2 n$ how many times we can devide by 2 befor n goes to or below 1?

Asymptotic Notation: O Ω Θ



Big-O Notation

Def Let $f(n)$ and $g(n)$ — try to compare f and g

$f: \mathbb{R}^+ \to \mathbb{R}^+$ $g: \mathbb{R}^+ \to \mathbb{R}^+$

$f(n)$ is $O(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = \text{Constant} < \infty$

$f(n)$ is $O(g(n))$ if $\exists\ c > 0$ and $n_0 \geq 1$ s.t. $f(n) \leq c \cdot g(n)\ \forall n \geq n_0$

Ex: $f(n) = 10n^2 + 11 \cdot n^3 + 3$

a. $g_1(n) = n^2$ Is $f(n)$ $O(g_1(n))$? N

b. $g_2(n) = n^3$ " Y

c. $g_3(n) = n^4$ " Y

let plug in the def

$\lim_{n\to\infty} \frac{10n^2 + 11n^3 + 3}{n^2} = \infty \to \text{not} = c$

$\lim_{n\to\infty} \frac{10n^2 + 11n^3 + 3}{n^3} = 11$ ($\frac{10}{n} + \frac{n^3}{11} + \frac{3}{n^3}$)

$\lim_{n\to\infty} \frac{10n^2 + 11n^3 + 3}{n^4} = 0$

So $f(n) = O(g(n))$ when $f \lesssim g$

Big-Θ (delta)

Def Let $f(n)$ and $g(n)$

$f: \mathbb{R}^+ \to \mathbb{R}^+$ $g: \mathbb{R}^+ \to \mathbb{R}^+$

$f(n)$ is $\Theta(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = C > 0$

Ex: $f(n) = 10n^2 + 11 \cdot n^3 + 3$

a. $g_1(n) = n^2$ Is $f(n)$ $O(g_1(n))$? N

b. $g_2(n) = n^3$ " Y

c. $g_3(n) = n^4$ " N

let plug in the def

$\lim_{n\to\infty} \frac{10n^2 + 11n^3 + 3}{n^2} = \infty \to \text{not} = c$

$\lim_{n\to\infty} \frac{10n^2 + 11n^3 + 3}{n^3} = 11$ ($\frac{10}{n} + \frac{n^3}{11} + \frac{3}{n^3}$)

$\lim_{n\to\infty} \frac{10n^2 + 11n^3 + 3}{n^4} = 0$

So $f \overset{\Theta}{=} g$

f and g are equal when super large n

$d(n) = O(f(n))$, $e(n) = O(g(n))$

i) $d(n) + e(n) = O(f(n) + g(n))$

ii) $d(n) \cdot e(n) = O(f(n) \cdot g(n))$

```
int moo(int x) {return x + 2;}
```

moo runs in c unit of time (no matter what x is) — constant

moo(5) → c
moo(13) → c
$2c$

```
for (int k=0; k<n; i++) {
    moo(k);
}
```

more computation to run
c_2 c_1
$n c_2$

$f(n) = nc_2 + (n-1)c_1 + n + c_2 = (c + c_1 + c_2)n - c_1 = O(n)$

```
int maxInt(int[] lst){
1:  int m = lst[0];          → c
2:  for(int i=1; i<lst.length; i++){
3:      if (m< lst[i])       → c
4:          m= lst[i];       → was run by n-1 times.
    }
5:  return m;                → c
}
```

$\lim_{n\to\infty} n+1 = \infty$

$c + (n-1)c + c = nc + c$ → do the limit thing

$f(n) = O(n)$ it behaves like a linear function

let use c

$f(n) = nc + (n-1)c + n + c = (3c)n - c_1$

```
int sum(List<Integer> xs) {
1:  int total = 0, n= xs.size();   → c
2:  for (int i=0; i<n; i++)        → g(i) time that takes to run
3:      total += xs.get(i);        → take different time bcs array vs linked list
4:  return total;                  → c
}
```

interference

$Total = c + c + g(0) + g(1) + \ldots + g(n-1) = 2c + \sum_{i=0}^{n-1} g(i)$

Scenario I: ArrayList

$g(i) = k$ constant

$= 2c + \sum_{i=0}^{n-1} k = 2c + kn = O(n)$

take lim ↓ n times

recall: $1+2+3+\ldots+\square = \frac{\square(\square+1)}{2}$

Scenario II: Linked List

$g(i) = k \cdot i$

$= 2c + \sum_{i=0}^{n-1} k \cdot i = 2c + k\left[\sum_{i=0}^{n-1} i\right] = 2c + k\left[0+1+2+\ldots+n-1\right]$

$= 2c + \frac{(n-1)(n)}{2}$

$= O(n^2)$

```
def isUnique(lst):
    n= lst.length                  → c
    for i in range(n):
        for j in range(i+1,n):
            if lst[i]==lst[j]:     → c
                return false;
    return true                    → c
```

array

$g(i) = c + g(0) + g(1) + \ldots + g(n-1)$

$= c + \sum_{i=0}^{n-1} g(i)$

$g(i) = k(n-1-i)$

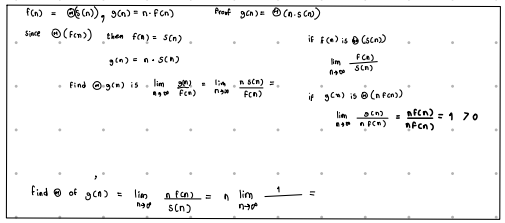
$= c + \sum_{i=0}^{n-1} k(n-1-i) = c + k\left[\sum_{i=0}^{n-1}(n-1-i)\right] = c + k\left[n-1+n-2+\ldots+1+0\right] = c + k\left[\frac{(n-1)(n)}{2}\right]$

$= O(n^2)$

if i=0 → j: 1, 2, ..., n-1 } n-1 ; if i=1 → j: 2, 3, ..., n-1 } n-2

B 14 !!

L93



Since if $\lim_{n\to\infty} \frac{f(n)}{g(n)} > 0$ $f(n) = \Theta(n)$ then $n = s(n)$

2 Tasks → $O(n^2)$ / $O(n \log n)$ → $O(n^2 + n \log n)$ → $O(n^2)$

L13

```
from copy import copy
def foobar(numbers: List[int]) -> List[int]:
    # make a copy of numbers
    numbers = copy(list(numbers))          → c
    for j in range(1, len(numbers)):       → n^2
        key = numbers[j]
        i = j-1
        while i >= 0 and numbers[i] > key:
            numbers[i+1] = numbers[i]
            i = i-1
        numbers[i+1] = key
    return numbers                         → c
```

Take $c + c + n^2 \to O(n^2)$

Chapter 6 : Performance Cha : Asymptotic Analysis

```
def is_unique(lst):
    n = len(lst) # the length of lst
    for i in 0..(n-1):
        for j in (i+1)..(n-1):
            if lst[i] == lst[j]:
                return False
    return True
```

take = $0, 1, 2, \dots, n-1$

$= f(0) + f(1) + \dots + f(n-1)$

$= \sum_{i=0}^{n-1} f(i)$

$i+1, i+2, \dots, n-1$

= for each $i = (n-1)-i$ value of j

inner loop each j

$f(i) = k \times (n-1-i)$

nested loop require a total of : $f(0) + f(1) + f(2) + \dots + f(n-1)$

$= \sum_{i=0}^{n-1} k \times (n-1-i)$

$= k \sum_{i=0}^{n-1} (n-1-i)$

So $(n-1)+(n-2)+\dots+(n-1-(n-1)) \simeq 1+2+3+\dots+(n-1) = \frac{n(n-1)}{2} = O(n^2)$

Ex:

```
void Fun(int n){
    int i, j, k, count = 0;          → O(1)
    for (i = n/2 ; i<= n; i++)       → O(n)
        for ( j = 1 ; j+n/2 <= n ; j++)   → O(n)
            for (k = 1 ; k <= n; k = k*2)  → O(log n)
                count ++;
}
```

Iter 1 $i = n/2 + 0$

Iter 2 $i = n/2 + 1$

Iter 3 $i = n/2 + 2$

Iter 4 $i = n/2 + 3$

Iter k $i = n/2 + (k-1) = n \rightarrow k = \frac{n}{2} + 1$ loop runs $\frac{n}{2}$ times

Iter 1 $j = 1$

Iter 2 $j = 2$

$j + \frac{n}{2} \le n$

$j \le n - \frac{n}{2} = \frac{n}{2}$

Iter k $j = k = \frac{n}{2}$ ← run $\frac{n}{2}$ times → $O(n)$

Iter 1 $k = 1 = 2^0$

Iter 2 $k = 2 = 2^1$

Iter 3 $k = 4 = 2^2$

Iter 4 $k = 8 = 2^3$

Iter p $k = 2^{p-1} = n$

$n = 2^{p-1}$

$\log n = \log 2^{p-1}$

$\log n = p-1 \ (\log 2^x)$

$p = \log n + 1 \rightarrow O(\log n)$

Let's conclude : nested loop

$O(1) + O(n) \cdot O(n) \cdot O(\log n)$

$= O(n^2 \log n)$

Ex:

```
void Fun(int n){
    if (n<= 1) return ;          → O(1)
    int i, j;                    → O(1)
    for (i = 1 ; i<= n ; i++ )   → O(n)
        for (j = 1 ; j <= n ; j++){   → O(1).
            printf("Hello \n");
            break;
        }
}
```

Iter 1 $i = 1$

I 2 $i = 2$

I 3 $i = 3$

I k $j = k$

$i = k = n$ → $O(n)$

wont let $j = 2$

take O(N) times

Ex:

```
void Fun(int n){
    int i = 1;                   → O(1)
    while (i<n) {                → I1 i = 1 = 2^0
                                   I2 i = 2 = 2^1
                                   I3 i = 4 = 2^2
                                   I4 i = 8 = 2^3
                                   Ik i = 2^(k-1)
        int j = n;               → O(1)
        while (j>0)
            j = j/2;
        i = 2*i;
    }
}
```

$i = 2^{k-1} = n-1$

$\log 2^{k-1} = \log n-1$

$k-1 = \log n-1$

$k = (\log n - 1) + 1$

$O(\log n)$?

I1 $j = n$

I2 $j = \frac{n}{2}$

I3 $j = \frac{n}{4}$

Ik $j = \frac{n}{2^{k-1}}$

$j = \frac{n}{2^{k-1}} = 1$

$n = 2^{k-1}$

$\log n = k-1$

$k = \log n + 1$

$O(\log n)$

total $O(\log n) \cdot O(\log n)$

$O((\log n)^2)$

Ex:

```
void Fun(int n){
    int i, j, k, count = 0;          → O(1)
    for (i = n/2 ; i <= n ; i++)
        for (j = 1 ; j <= n ; 2*j)
            for (k = 1 ; k <= n ; k++)
                count ++;
}
```

I1 $i = \frac{n}{2}$

I2 $i = \frac{n}{2} + 1$

I3 $i = \frac{n}{2} + 2$

Ik $i = \frac{n}{2} + k - 1 = n$

$k = \frac{n+1-\frac{n}{2}}{2} = \frac{2n+2-n}{2} = \frac{n+2}{2} \rightarrow O(n)$

I1 $j = 1$

I2 $j = 2$

I3 $j = 4$

Ik $j = 2^{k-1}$

$2^{k-1} = n$

$\log n = k-1$

$\log n + 1 = k$

$O(\log n)$

I1 $k = 1$

I2 $k = 2$

I3 $k = 3$

Ip $k = p$

$k = n$ → $O(n)$

all = $O(n) \cdot O(n) \cdot O(\log n) = n^2 \log n$

```
void programA(int n) {
    long prod = 1;                   → O(1)
    for (int c=n;c>0;c=c/2)          →
        prod = prod * c;
}

void programB(int n) {
    long prod = 1;                   → O(1)
    for (int c=1;c<n;c=c*3)
        prod = prod * c;
}
```

I1 $c = n$

I2 $c = \frac{n}{2}$

I3 $c = \frac{n}{4}$

Ia $c = \frac{n}{8}$

Ik $c = \frac{n}{2^{k-1}}$

$k = 1 = \frac{n}{2^{k-1}}$

$2^{k-1} = n$

$(k-1)\log 2 = \log n$

$k-1 = \log n$

$k = \log n + 1$

$k = O(\log n)$ #

I1 $c = 1$

I2 $c = 3$

I3 $c = 9$

Iq $c = 27$

Ik $c = 3^{k-1}$

$k = n-1 = 3^{k-1}$

$k-1 \log 3 = \log(n-1)$

$k-1 = \frac{\log(n-1)}{\log(3)}$

$k = \log(n-1-3)+1$

$k = \log(n-4)+1$

Youtube

Calculate Big-O

if, switch , +, -, >, <, = — constant , O(1)

for loop n times — O(n)

```
for (m) {
    for (n) {}
}
```
— O(m·n)

```
for (m){
    for ( int j=0 ; j<size; j*=2) {}
}
```
— O(n × $\log_2 n$)

run 2 for() at same time — O(2n) = O(n)

array a of len m , Arrays.sort(a) — O(m log m)

Arrays.copyOfRange (a, from, until) — O(until - from)

Linked list .get(i) — O(n²)

Ex: The code contains:

copyOfRange	O(i)
Arrays.sort	O(i log i)
ans[i] = copy[i]	O(1)

iteration i costs O(i) + O(i log i) + O(1) = O(i log i) = c · i log i

if i = 0 → 0

i = 1 → c 1 log 1

i = 2 → c 2 log 2

i = n-1 → c(n-1) log(n-1)

every log i is almost log n cos i ≤ n-1

$\sum_{i=0}^{n-1} c \cdot i \log n = (c \cdot \log n) \times (1+2+3 \ldots + n-1) = O(n^2 \log n)$

Comparing two functions

upperbound

f(n) = O(g(n))

If there exists constants C and n_0 (if not take limit)

f(n) ≤ c.g(n) for all n ≥ n_0

Meaning f(n) does not grow faster than g(n)

Ex: f(n) $\stackrel{?}{=}$ O(g(n))

f(n) ≤ c · g(n)

n ≤ c · 2n if c=1, n_0=1

1 ≤ 2 ✓

Recursive Code Analysis

each simple of operation costs 1 unit of time

One of the method where we try to reduce T(n) interms of its base conditions.

```
factorial(n) {
    if n==0 → 1
        return 1
    else
        return n × factorial(n-1)
}
```
(n: 1 unit, factorial(n-1): 1 subtract)

T(n) = time taken to calculate factorial(n)

T(n) = T(n-1) + 3 if n>0

T(0) = 1 ← try to make it to T(0)

T(n) = T(n-1) + 3

= T(n-2) + 6

= T(n-3) + 9

= T(n-k) + 3k

If we want T(0)

n-k = 0

k = n

⇒ T(n) = T(0) + 3n

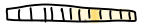
= 1 + 3n

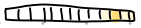
O(n) #

Iteration

- Find the middle of the list each time


Iteration 1 = $\frac{n}{2}$


Iteration 2 = $\frac{(\frac{n}{2})}{2} = n/2^2$


Iteration 3 = $\frac{(\frac{n}{2^2})}{2} = n/2^3$


Iteration k = $\frac{n}{2^k}$

let's think of worst case scenario

Iteration k = $\frac{n}{2^k}$

1 = $\frac{n}{2^k}$

n = 2^k

$\log_2 n = \log_2 2^k$

$\log_2 n = k \log_2 2$

$\log_2 n = k$

k = log n → O(log n)

Ex: we have 8 elements

k = O(log n) = $\log_2 8 = \log_2 2^3 = 3 \log_2 2$ = 3 iterations

Let d(n) = O(f(n)) , e(n) = O(g(n)) then

d(n) + e(n) = O(f(n) + g(n))

d(n) · e(n) = O(f(n) · g(n))

```java
int primSum(int[] xs) {
    if (xs.length == 1) return xs[0];
    if (xs.length == 2) return xs[0] + xs[1];
    else {
        int[] ys = Arrays.copyOfRange(xs, 1, xs.length);
        return xs[0]+xs[1]+primSum(ys);
    }
}

int whazIt(int[] ys) {
    if (ys.length == 0) return 0;
    if (ys.length == 1) return ys[0];
    int n = ys.length;
    int m = n/2;
    for (int i=0;i<n;i++) {
        int theSum = 0;
        for (int j=0;j<=i;j++) { theSum += ys[j]; }
        ys[i] = theSum;
    }
    int a = whazIt(Arrays.copyOfRange(ys, 0, m));
    int b = whazIt(Arrays.copyOfRange(ys, m, ys.length));
    return a + b;
}
```

O(1) O(1) O(n) T(n-1)

O(1) O(n²) T(n/2) O(n/2) O(n/2) T(n/2)

primSum

T(n) = T(n-1) + O(n) = O(n²)

i=0 → $\sum_{j=0}^{0}$ j-i = 0

i=1 → $\sum_{j=0}^{1}$ j-1 = (0-1)+(1-1) = -1

i=2 → $\sum_{j=0}^{2}$ j-2 = (0-2)+(1-2)+(2-2) = -2-1

i=3 → $\sum_{j=0}^{3}$ j-3 = (0-3)+(1-3)+(2-3)+(3-3) = -3-2-1

i=n → $\sum_{j=0}^{n}$ j-n = (0-n)+(1-n)+...+(n-n) = -n-(n-1)-...-1

= -[1+2+3+...+(n-1)+n]

= $-\frac{n(n+1)}{2}$

Total time to run = O(n²)

Total T(n) = O(1)+O(n²)+T(n/2)+T(n/2)+O(n/2)

= O(n²)+2T(n/2)

= O(n²) #

whazIt

```java
int mcss(int[] a) {
    int maxSum = Integer.MIN_VALUE; → O(1)
    for (int i=0;i<n;i++) {
        for (int j=i;j<n;j++) { → each j will run different
            thisSum = sumSubseq(a, i, j+1); → takes k(j-i)
            if (thisSum > maxSum)
                maxSum = thisSum;
        }
    }
    return maxSum
}
```

I1 i=0 → $\sum_{j=0}^{n} k(j-0)$ → 0+1+2+...+n = $k\frac{n(n+1)}{2} \le n^2$

I2 i=1 → $\sum_{j=1}^{n} k(j-1)$ → 1+2+3+...+n-1 = $k\frac{(n-1)(n)}{2} \le (n-1)^2$

I3 i=2 → $\sum_{j=2}^{n} k(j-2)$ → 2+3+...+n-2 (+1, -3) = $k\frac{(n-2)(n-1)}{2}$

In i=n-1 → $= k\frac{(n-i)(n-i+1)}{2} \le i^2$

$= k(n-i)^2$

total time f(n) = $\sum_{i=0}^{n-1} k(n-i)^2$

$= k(1^2+2^2+...+n^2)$

$= \frac{1}{6}n(n+1)(2n+1)$

$O(n^3)$ #

```java
// assume xs.length is a power of 2
int halvingSum(int[] xs) {
    if (xs.length == 1) return xs[0]; — O(1)
    else {
        int[] ys = new int[xs.length/2];
        for (int i=0;i<ys.length;i++) — O(n)
            ys[i] = xs[2*i]+xs[2*i+1];
        return halvingSum(ys); → T(n/2)
    }
}
```

T(n/2) + O(n) = O(n) #

```java
int anotherSum(int[] xs) {
    if (xs.length == 1) return xs[0]; → O(1)
    else {
        int[] ys = Arrays.copyOfRange(xs, 1, xs.length); → O(n-1)
        return xs[0]+anotherSum(ys); → T(n-1)
    }
}
```

T(n-1) + O(n) = O(n²)

```java
int[] prefixSum(int[] xs) {
    if (xs.length == 1) return xs; — O(1)
    else {
        int n = xs.length; — O(1)
        int[] left = Arrays.copyOfRange(xs, 0, n/2); — O(n/2)
        left = prefixSum(left); T(n/2)
        int[] right = Arrays.copyOfRange(xs, n/2, n); — O(n/2)
        right = prefixSum(right); T(n/2)

        int[] ps = new int[xs.length]; — O(1)
        int halfSum = left[left.length-1]; — O(n/2 - 1) — O(n/2)
        for (int i=0;i<n/2;i++) { ps[i] = left[i]; }
```

i=0 → O(n/2) → 4

k-1 = n/2 - 1

```
        for (int i=n/2;i<n;i++) { ps[i] = right[i - n/2] + halfSum; }
        return ps;
    }
}
```

O(n/2)

O(1)+O(1)+O(n/2)+O(n/2)+2T(n/2)+O(1)

+O(n/2)+O(n/2)+O(n/2)

2T(n/2)+O(n) → O(n log n)

```java
static boolean method2(int[] array, int key) {
    int n = array.length;
    for (int index=0;index<n;index++) {
        if (array[index] == key) return true;
    }
    return false;
}
```

I1 index = 0

I2 index = 1

Ik index = k-1

k = k-1 = n-1

k = n

B.C. first index is key O(n)

W.C. encounter every elements O(n)

```java
static void method1(int[] array) {
    int n = array.length;
    for (int index=0;index<n-1;index++) {
        int marker = helperMethod1(array, index, n - 1); — O(n)
        swap(array, marker, index); — O(1)
    }
}
static void swap(int[] array, int i, int j) {
    int temp=array[i];
    array[i]=array[j];
    array[j]=temp;
}
static int helperMethod1(int[] array, int first, int last) {
    int max = array[first];
    int indexOfMax = first;

    for (int i=last;i>first;i--) {
        if (array[i] > max) {
            max = array[i];
            indexOfMax = i;
        }
    }
    return indexOfMax;
}
```

O(1)

O(n)

I1 index = 0

I2 index = 1

I3 index = 2

Ik index = k-1

k-1 = n-2

k = n-1

O(n)

O(n)·O(n)

= O(n²) #

I1 i = last

I2 i = last-1

I3 i = last-2

Ik i = last-(k-1)

k = first+1 = last-(k-1)

first+1 - last = -k

last - first = k

O(n)

W.C = O(n²)

B.C = O(n²) → when

```java
static double method3(int[] array) {
    int n = array.length;

    3
```

work 5

```java
    double sum = 0;

    for (int pass=100; pass >= 4; pass--) {
        for (int index=0;index < 2^n;index++) {
            for (int count=4^n;count>0;count/=2)
                sum += 1.0*array[index/2]/count;
        }
    }
    return sum;
}
```

I1 c = 4n

I2 c = 2n

I3 c = n

I4 c = n/2

Ik c = $\frac{n}{2^{k-1}}$

$\frac{n}{2^{k-1}} = 1$

$n = 2^{k-1}$

log n + 1 = k

O(log n)

I1 i = 0

I2 i = 1

Ik i = k-1

k-1 = 2n+1

k = 2n+2

O(n)

I1 p = 100

I2 p = 99

Ik p = (100) - (k-1)

100 - k + 1 = 4

-k = 3 - 100

k = 97

O(1)

L 14 :

- ☐ More Running Time : mcss
- ☐ Recursive Code Analysis
  - O pow O sum
- ☐ Serching

mcss Maximum Contiguous Subseq. Sum

$a_0 \quad a_1 \quad a_2 \quad \dots \quad \dots \quad a_{n-1}$

ex 1: 1 2 5 7 9 12 — mcss
(it's all + the bigest subseq. is to include everyone)

ex 2: -2 11 -4 13 -5 -2 — mcss

• time calculation

How we pick the best mcss?

```
int BestSum=0
for i = 0,1,2,...,n-1
  for j= i+1...,n
    if sum(a[i:j]) > bestSum
      bestSum = sum(a[i:j])
```

(only a the biggest one)
python — in java:

```
static int sumSeq (int[] a, int start, int end){
  int sum =0;
  for (int i= start; i<end; i++)
    sum += a[i];
  return sum;
}
```

$k(j-i)$ — j-i how big of the length is how many time it loops

i=0 → j

$f(i=0) = \sum_{j=1}^{n} k(j-0) = k\frac{n(n+1)}{2}$

$f(i=1) = \sum_{j=1}^{n} k(j-1) = \sum_{j=1}^{n} k(1+2+3+\dots+n-1) = k\frac{(n-1)n}{2}$

$f(i=2) = k\frac{(n-2)(n-1)}{2}$

$f(i) = k\frac{(n-i)(n-i+1)}{2} \leq k(n-i)^2$ (simplify)

Total time: $\sum_{i=0}^{n-1} f(i) \leq \sum_{i=0}^{n-1} k(n-i)^2$

$= k(1^2+2^2+3^2+\dots+n^2)$

$= k\cdot\frac{1}{6}n(n+1)(2n+1)$ — Wolframalpha — calculation site

interm of big O $= O(n^3)$

FFT: 1. (H)?
2. can we do better than n^3? (sum) we can make it to n^2
1 2 5 — we know that we don't have to re sum all of sq.

write another func - parallel, simplier to the one that we want to analyzed to determine the running time

Recursive Code Analysis

Write a recurrence relation

ex: $T(w) :=$ time taken to run pow (x, w)

$T(0) = O(1)$ → const

For $w>0$, $T(w)$ — only change recursively in w

```
int pow (int b, int w){
  if (w ==0)        → const O(1)
    return 1;
  return b * pow(b, w-1);
}
```

(b * → const O(1) multi; pow(b, w-1) → T(w-1) my friend)

$w-k=0$
$k=w$

$T(0)=1$
$T(w) = T(w-1)+3$
$= T(w-2)+6$
$= T(w-k)+3k$
$= T(0)+3w$
$= 1+3w$
$= 3w$
$= O(w)$ as we know
$= O(n)$ — Same ans

if T(w) and w=0 O(1)

for w > 0
1) we call ourselves w/ w-1
2) we × b

$T(w) = T(w-1) + O(1)$

Common Recurrences

We list a couple common recurrences, assuming T(0) and T(1) are constant:

- $T(n) = T(n/2) + O(1)$ solves to $O(\log n)$.
- $T(n) = T(n/2) + O(n)$ solves to $O(n)$.
- $T(n) = 2T(n/2) + O(1)$ solves to $O(n)$.
- $T(n) = 2T(n/2) + O(\log n)$ solves to $O(n)$.
- $T(n) = 2T(n/2) + O(n)$ solves to $O(n \log n)$.
- $T(n) = 2T(n/2) + O(n^2)$ solves to $O(n^2)$.
- $T(n) = T(n-1) + O(1)$ solves to $O(n)$.
- $T(n) = T(n-1) + O(n)$ solves to $O(n^2)$.

So $T(w) = T(w-1) + O(1)$ — look at the table — $O(n)$

$T(w) = O(w)$ → linear

```
int pow2(int b, int w) {
  if(w==0)
    return 1;
  if (w%2 ==0)
    return pow2 (b, w/2)* pow2 (b, w/2)
  else
    return b* pow2(b, w/2)* pow2(b,w/2);
}
```

$T(w) :=$ time taken to run pow 2 (x, w)

$T(0) = O(1)$

For $w>0$, $T(w) = O(1) + T(w/2) + T(w/2) + O(1)$ (if; pow2, pow2 — recursive call twice; mult) ($2T(w/2) \neq T(w)$)

$= 2T(w/2) + O(1)$

in the table: $T(w) = O(w)$ → linear in the sift same as pow()

```
int pow3(int b, int w) {
  if(w==0)
    return 1;
  int t = pow3(b, w/2);
  if (w%2==0)
    return t*t;
  else
    return b*t*t;
}
```

(it keep some variables)

$T(w) :=$ time taken to run pow3 (x, w)

$T(0) = O(1)$

For $w>0$ $T(w) = O(1) + T(w/2) + O(1)$ (if; mult)

$= T(w/2) + O(1)$

$T(w) = O(\log w)$

☐ Sum

```
int sum(int[] a) {
    if (a.length==0) return 0;
    else if (a.length==1) return a[0];
    else {
        m = a.length/2;
        return sum(Arrays.copyOfRange(a,0,m)) +
            sum(Arrays.copyOfRange(a,m,a.length));
    }
}
```

(size n/2; 0,...,m → O(m))

$T(n) = O(1) + T(\frac{n}{2}) + T(\frac{n}{2}) + O(n)$ (if, else if)

$= 2T(\frac{n}{2}) + O(n)$

$T(n) = O(n \log n)$

Searching

a = [10, 19, 3, 11, 33, 42, 5] — uses for loop, return inside, 1 outside

Does a contain 42? — linear search - O(n)

sorted version of a: b = [3, 5, 10, 11, 19, 33, 42] — should be faster than a
(see a middle; 42 > 11 so we'll focus on only; middle; 42 > 33 so focus on)

```
boolean exists (int[] a, int key) {
  if (a.length ==0)
    return false;
  int mi = a.length/2;
  if (a[mi] == key)
    return true;
  else if (a[mi] < key)   // focus on the right
    return exists(a[mi+1:],key) → Jython
    return exists(Arrays.copyOfRange(a,mi+1,),key) → Java
  else
    return exists(a[:mi],key)
}
```

$T(n) =$ T taken to run exists on a of len = n

$T(0) = O(1)$

$T(n) = O(1) + n/2 + T(n/2)$ (if += : +<; half O(1) a[:] slicing; recurse)

a' — O(size copied) — we should fake it not actually copied

$T(n) = T(n/2) + O(n) = O(n)$ — no improvement from a we're wrong!

// view just take a look but do nothing

```
boolean existsHelper (int[] a, int key, int start, int stop) {
  int n = stop - start;
  int mi= (start+stop)/2;
  if (a[mi]==key)
    return true;
  else if ( a[mi] < key)      look on the right
    return existsHelper(a,key, mi+1, stop);
  else
    // to do
}
```

$T(n) = T(n/2) + O(1)$

$= O(\log n)$ → cus it devides each iteration time by 2

key: avoid copying a list as possible.

5.15:

Sorting

The simpler sorting algorithms have $O(n^2)$ running time, We can make it $O(n \log n)$

- comparison based algorithms $x < y$, $x == y$, $x > y$
- work w/ Comparable interface → x.compareTo(y) → return int → boolean less(Tx, Ty) return x.compareTo(y) < 0;
  - public class Sorting <T extends Comparable<T>>
    - passitive value (>0) → x > y
    - zero (=0) → x == y also x.equals(y) is true
    - negative value (<0) → x < y
- we define a utility function void swap(T[] a, int i, int j) inside this class (a[i] ↔ a[j])

```
boolean isSorted(T[] a){ check whether a is already sorted or not
    int n = a.length;
    for (int i=0; i < n-1; i++){
        if (a[i].compareTo(a[i+1])>0)   → a[i] ? a[i+1]   expect a[i].compareTo(a[i+1])<=0
            return false;
    }
    return true;
}
```

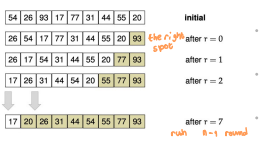
Worst case: O(n)

Best case: O(1) compare the first pair and it's wrong

Bubble Sort

not practical

- from previous one we check that the list is sorted or not, if not, we'll swap here.
- The sequence is sorted = no swaping
- The (r+1) rightmost spots store the (r+1) largest elements



most n-1 passes are needed

pass r takes O(n-r) time

and the code performance at most n-1 passes,

the running time is $O(n^2)$

$(n-1) + (n-2) + \dots + 1 = O(n^2)$

```
void bubbleSort(T[] a) {
    int n = a.length;
    for (int r=0;r<n-1;r++) {        → O(n-1) → O(n)
        for (int i=0;i<n-r-1;i++) {  → O(n)
            if (a[i+1].compareTo(a[i]) < 0)
                swap(a, i, i+1);
        }
    }
}
```

Rounds required ≤ n-1

Time per round: O(n)

Bubble Sort $O(n^2)$ w.c

O(n) best go 1 round

Insertion Sort

read notes, it's faster than bubble sort → Selection Sort: $O(n^2)$ (read in the note)

insert new element each time

s = [26, 54, 93, 17, 77, 31, 44, 55, 20]

N = [26, 54, 93] (find the right spot, push)

[17, 26, 54, 93]

[17, 26, 54, 77, 93]

normally go through each element

26

26, 54

26, 54, 93

(17, 26, 54, 54, 93) 93 ← do more work

- scane from the right end find i spot: O(1) best case (the right is there), O(i) worst case
- push i

Round # = 0, 1, 2, ..., n-1

Each round # =

Insertion Sort: sorted every element go so far right everytime

O(n) B.C.

$\sum_{i=0}^{n-1} i = \frac{n(n+1)}{2} = O(n^2)$ W.C.

Merge Sort

cut down the problem size in half

s = [26, 54, 93, 17, 77, 31, 44, 55, 20]

my friend's called msort → [26, 54, 93, 17] [77, 31, 44, 55, 20] → [17, 26, 54, 56, 93] [20, 31, 44, 56, 77]

merge (fully sorted!)

keep track on x, y and compare each one

[~~17~~, 26, 56, 93] [~~20~~, 31, 44, 55, 77]

[17, 20 .....]

mergeInto → O(n) w.c.

```
void mergeInto(int[] x, int[] y, int[] out){
    int i = 0; //index into x not yet been outputted
    int j = 0; //index into x not yet been outputted
    for (int k=0; k < out.length; k++){ → O(n)
        if (i >= x.length)          // run out of x
            out[k] = y[j++];
        else if (j >= y.length)     // run out of y
            out[k] = x[i++];
        else if (x[i] <= y[j])
            out[k] = x[i++];
        else
            out[k] = y[j++];
    }
```

```
void mSort(int[] a){
    int n = a.length;
    if (n <= 1) return;
    int[] left = Array.copyOfRange(a,0,n/2)   → O(n/2)
    int[] right = Array.copyOfRange(a, n/2, n) → O(n-n/2) → O(n/2)
    mSort(left);   → T(n/2)
    mSort(right);  → T(n/2)
    mergeInto(left, right, a); → O(n)
}
```

T(0) = O(1)

T(1) = O(1)

$T(n) = O(n/2) + O(n/2) + T(n/2) + T(n/2) + O(n)$

(copy, copy, msort(left), msort(right), merge Into)

$T(n) = 2T(n/2) + O(2n) = 2T(n/2) + O(n) = O(n \log n)$ #

no worst cas or best case here it'll split everytime

Quick Sort

not practical

let's call "pivot" or "p"

just random

s = [26, 54, 93, 17, 77, 31, 44, 55, 20]

call friend: [26, 17, 31, 20] < 44

sorted: [44] = 44

call friend: [54, 93, 77, 55] > 44

qsort → [17, 20, 26, 31] [44] [55, 54, 77, 93] ← qsort

[17, 20, 26, 31, 44, 55, 56, 77, 93] done!

concatination

T(0) = T(1) = Q(1)

T(n) =

```
def qsort(xs: List[int]) -> List[int]:
    if len(xs) <= 1: return xs
    p = xs[0] — random.choice(xs)
    lt = [x for x in xs if x < p]   → O(n)
    eq = [x for x in xs if x == p]  → O(n)
    gt = [x for x in xs if x > p]   → O(n)
    return qsort(lt) + eq + qsort(gt)
           T(n_lt)        T(n_gt)
```

O(1) ↑ we don't know exact "number" 0 → n-1

$T(n) = \underbrace{O(1)}_{p} + \underbrace{O(n)}_{lt} + \underbrace{O(n)}_{eq} + \underbrace{O(n)}_{gt} + \underbrace{T(n_{lt})}_{qs(lt)} + \underbrace{T(n_{gt})}_{qs(gt)} + \underbrace{O(n)}_{++}$

$2T(n/2)$ (++: concatination happen n times)

$= T(n-1) + O(n) = O(n \log n)$

$T(n) = O(n^2)$ # when xs = [1, 2, 3, 4, 5]

bc we pick xs[0]

↓ median takes time

↓ use random u might be lucky

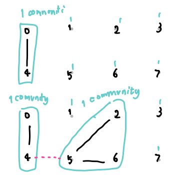
Read 8.9

L16: Disjoint Sets

Fiction: Connected Worlds

No one know any one — 0 1 2 3
Everyone lives in their own community since they only know themselves — 4 5 6 7

0 & 4 become friend



link (4,5)
- link (x, y) - unite the community of x with comm of y
- Connected (x, y) - ask if x knows y? (same community)
  - we all know ourselves

each community

They are disjoint

Storing Data Sets

set of set is weird in java
- List < Set < Integer > >   [{0,4}, {1}, {2,5,6}, {3,7}]

```
class DisjointSets {
    List< Set < Integer > > allTheSets;
    Map < Integer, Integer > backRef     # index {0: 0, 1:1, 2:2, 3:3, 4:0, 5:2, 6:2, 7:3}
    int[] backRef ;  int[] size ;
    DisjointSets (int n) {
        backRef = new int [n];
        size = new int [n];
        for (int k=0; k<n; k++) backRef[k] = k;  size[k] = 1;     // point themselves
    }

    // return index of the set where x appears     → find someone who points on itself
    int find (int x) {
        while (backRef[x] != x) {       // if (backRef[x] == x) return x; return backRef[x] = find(backRef[x]);
            x = backRef[x];             → log* n
        }
        return x;
    }

    void link (int x, int y) {
        int xIndex = find(x), int yIndex = find(y);
        if (size[xIndex] <= size[yIndex]) {
            backRef[xIndex] = yIndex;
            size[yIndex] += size[xIndex];
        } else {
            backRef[yIndex] = xIndex;
            size[xIndex] += size[yIndex];
        }
    }

    boolean isConnected (int x, int y) {     // O(1)
        int xIndex = find(x), int yIndex = find(y);
        return xIndex == yIndex;
    }
}
```

Lazy
Find: O(n)
link: O(n)
is Conn: O(n)

link (4,5)
[{0,4}, {1}, {2,5,6}, {3,7}]

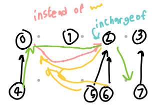
$\log_2 n$ → how many time $\frac{n}{2 \cdots 2} \le 1$

$\log^* n$ → how many log $\log((\log)\dots) \le 1$

Ex: $\log^*(2^{2^{16}})$ → 2^{16} → 16 → 4 → 2 → 1   (5)

Idea:
- → large = deep
- → join small → large

instead of ... in charge of ... 2 lives



backRef = parent of , point to

backRef[i] i points to backRef[i]

0	1	2	3	0	2	2	3
0	1	2	3	4	5	6	7

if link (4,7) — does not point to anyone yet

Lemma If r is a root, then $h(r) \le \log_2(size[r])$

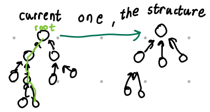
Proof (invariant) true forever, initially all height 0

$size(r) \le n$
$h(r) \le \log_2(n)$

Before:
link(x,y):
$h(r_x) \le \log_2(size[r_x])$
$h(r_y) \le \log_2(size[r_y])$
$size[r_x] \le size(r_y)$

After:
$size'[r_y] = size[r_y] + size[r_x]$
$h' = \max(h(r_y), 1 + h(r_x))$
WTS $h' \le \log_2(size[r_y] + size[r_x])$
$1 + h(r_x) \le 1 + \log_2(size[r_x]) \to \log_2(2 size[r_x])$
claim: $2 size(r_x) \le size(r_y) + size(r_x)$
bc $size[r_x] \le size(r_y)$

current one, the structure so deep

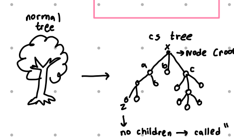


L17: Priority Queues (PQs) → VIP gets serve first

PQ: Max-oriented
n ← # of items

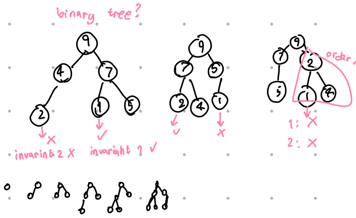
	Unordered	sm→large Sorted	Binary Heap tree (goal)
- find Max	O(n)	O(1)	O(1)
- remove Max	O(n)	O(1)	O(log n)
- add	O(1)	O(n)	O(log n)
- size	O(1)		
- isEmpty	O(1)		

PQ: Sort [Heap Sort]



x is the parent of a
a is a child of x
→ not Binary trees: every node has ≤ 2 children
no children → called "a leaf"

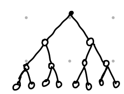
binary tree?



invariant 2 ✗   invariant 1 ✓   order ✗   1: ✗ 2: ✗

Perfect binary tree:

In real life we can't make it perfect but we'll make it close



→ all leaves same level

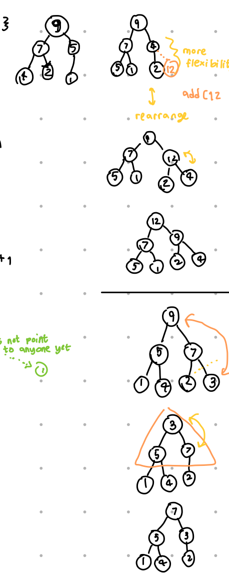
Binary Heap Tree

- a binary tree with two invariants:

ordering ① $x \ge y$ && $x \ge z$ — The root has the max numb

shape ② Full at all levels except potentially at the bottom level, where the node or 2 left-justified

S = {1, 2, 4, 5, 7, 9}



more flexibility — add (12) — rearrange

- find Max   O(1) → root
- add
  - Step 1: Add to bottom right
  - Step 2: "Swim" until no more violations

time ~ height of tree

- remove Max
  - Step 1: Swap root & bottom right
  - Step 2: Sink the root val

time ~ height of tree

Ex: Input: a long list of numbers   k << n
output: lowest k numbers

if k = 3
Store the lowest k number
cont. in 20, 4, 9, 11
add 10
20, 4, 9, 11, 10

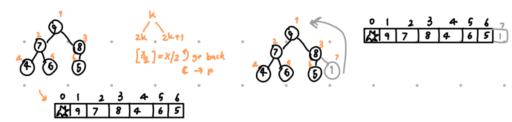
store lowest   call recv O(n)   the total time = O(nk)

```
class SmallestK {
    List<Integer> bottom;
    int k;
    SmallestK (int k) { this.k = k; bottom = new ArrayList<>(); }
    void recv (int num) {            → O(k)
        bottom.add(num);
        if (bottom.size() > k) {
            int largest = Collections.max(bottom);   ← O(k)
            bottom.remove(largest);                  ← O(k)
        }
    }
}
```

size ≤ k+1

How to write a tree in Java?

k: 2k, 2k+1; [k/2] = x/2 go back, c → p



0	1	2	3	4	5	6
	9	7	8	4	6	5

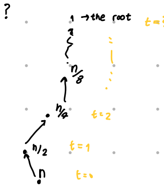
What is the height?

→ The bottom

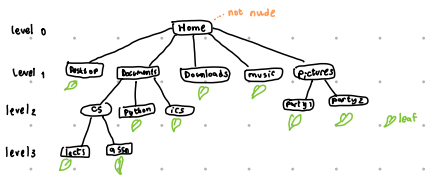
$\frac{n}{2^t} = 1$

$n = 2^t$

$t = \log n$



Lec 13 : Tree, Binary Trees & Leisure walk?.



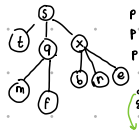
Def^n A tree $T = (N, P)$ where $p: N \to N \cup \{\perp\}$ (nodes, "parent" function; no parent)

T tells u who is the parent

satisfying : ① Unique root r s.t. $p(r) = \perp$ root has no parent

② Every node $v \neq r$ has a parent in N

③ for every node v, $p(\cdots p(p(p(v))))) \rightsquigarrow = r$
keep asking v's parent will lead to root



$p(s) = \perp$
$p(t) = s$
$p(b) = x$

dict map → finite function ≈ map
m : parent

{ s: null, t: s, q: s, x: s, m: q, f: q, b: x, r: x, e: x }

TreeMap
HashMap

Binary tree: every node has 0, 1, 2

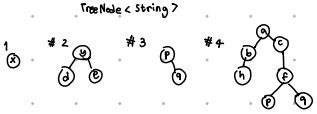
all valid : no children | 2 children (left, right) | (right, left)

want this capability

some data, name, ##

Special Tree: Binary Heap
↳ keep in array

TreeNode < String >

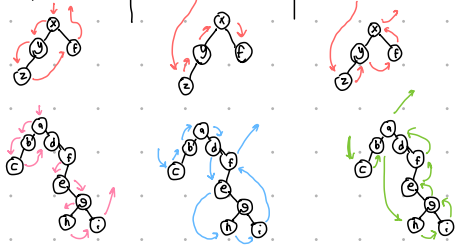



```
class TreeNode<K>{
    K key;  // pay load
    TreeNode<K> left, right;

    TreeNode(K key, TreeNode<K> left, TreeNode<K> right){
        this.key = key;  this.left = left;  this.right = right;
    }
}
main{
    TreeNode<String> t1 = new TreeNode<String>("x", null, null);
    TreeNode<String> t2 = new TreeNode<String>("y",
                                  new TreeNode<>("d", null, null),
                                  new TreeNode<>("e", null, null));
    TreeNode<String> t3 = new TreeNode<String>("y", null, new TreeNode<>("q", null, null));
```

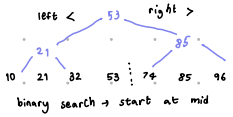
```
static<K> int numLevels(TreeNode<K> root){
    if (root == null)
        return 0;
    return 1 + Math.max(numLevels(root.left), numLevels(root.right));
}
```

now

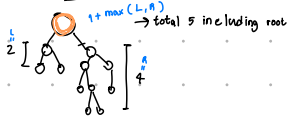
pre-order	in-order	post-order
def pre(r):	def inorder(r)	def post(r):
visit(r)	inorder(r.left)	post(r.left)
pre(r.left)	visit(r)	post(r.right)
pre(r.right)	inorder(r.right)	visit(r)



Ex.



binary search → start at mid



L19 : Maps & BST

Map (dict, associative array)

$f : D \rightarrow R$ (D: keys, R: value)

finite func

key	value
"Thai"	"Baht"
"Japan"	"Yen"

Java: `Map<K,V>`

`Map<String, Integer>`

Types: HashMap

TreeMap

There are two classes of map:

Operation	Unsorted (HashMap)	Sorted (TreeMap)
• get(k) → return value of key k	O(1)	O(log n)
• put(k,v) → store val v into key k	O(1)	O(log n)
• containsKey(k) → does it has k?	O(1)	O(log n)
• remove(k) → delete key k	O(1)	O(log n)

lowerKey(k) → the greatest key $< k$

floorKey(k) → the greatest key $\leq k$

ceilingKey(k) → the least key $\geq k$

higherKey(k) → the least key $> k$

} O(n)

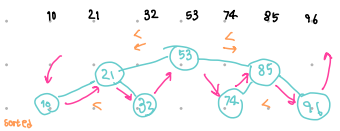
O(log n) binary search

ex: K: 1 5 7 9

lowerKey(7) → 5

floorKey(7) → 7

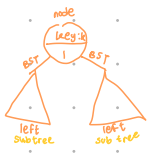
Tree: 10 21 32 53 74 85 96



sorted

in-order {10, 21, 32, 53, 74, 85, 96} sorted!

if 21 > k then go right



keys in left $< k <$ keys in right

• Binary Search Tree [BST]

Def: A binary search tree is defined recursively as

a) an empty tree; or

b) a node storing a key k & two BST – left & right – where entire left $< k <$ entire right

Observation: want nice & "bushy"

```
class TreeNode {
    int key;
    String value;
    TreeNod left, right;
}

public class Lecture19 {
    // if empty, return null
    public static Integer lastKey(TreeNode root){
        if (root == null) return null;
        if (root.right == null) return root.key
        return lastKey(root.right);
    }

    public static String get(TreeNode root, int k){
        if (root == null) return null;
        if (root.key == k) return root.value;
        else if (root.key < k) return get(root.right, k);
        else return get(root.left, k);
    }

    public static Integer floorKey(TreeNode root, int k){
        if (root == null) return null;
        if (root.key == k) return k;
        else if (k < root.key && root.left != null) return floorKey(root.left, k);
        else if (k > root.key) {
            Integer rightFloor = floorKey(root.right, k);
            if (rightFloor == null) return root.key;
            else return rightFloor;
            return (rightFloor == null)? root.key : rightFloor;
        }
        else return null;
    }
}
```

read only, won't modify

running time: height

TreeMap

Red-Black Tree

$h \leq 2\log_2(n+1)$

The running time: ~ h how tall the tree

worst:

put(k,v) do the get and fail → insert

height

O(h)

if 80? move tree → height control takes place

$h \geq \log_2(n+1)$

Introduction to Graph Theory & Representation

① They call me graphs

good for represent asymmetric relationship

directed graph (digraph)

4 vertices
4 arcs

Ex: This graph is $G = (V, A)$, where

$V = \{a, b, c, d\}$ set of vertices (nodes)

$A = \{(a,b), (c,b), (b,d), (c,d)\}$ set of directed edges (arcs)

undirected graph

edge $\{u,v\}$ is same as edge $\{v,u\}$

4 vertices
5 edges

Ex: this undirected graph example is $G = (V,E)$, where

$V = \{a, b, c, d\}$

$E = \{\{a,b\}, \{a,c\}, \{a,d\}, \{b,c\}, \{b,d\}\}$

Graph Terminology

Neighbors & Neighborhood: vertex u —neighbor / adjacent→ vertex v if edges btw them

for directed graph: in- and out- neighbors of a vertex is distinguishable

for undirected graph: $G = (V,E)$ neighborhood of v = $N_G(v)$ or $N(v)$ → {neighbors of v}

$N_G(v) = \{u \in V \mid \{u,v\} \in E\}$

when $G = (V,A)$ is directed: $N^+(v)$ set of out-neighbors of v N^+

$N^-(v)$ set of in-neighbors of v N^-

degree: $\deg(v) = |N(v)|$ (to directed: $\deg^+$ and $\deg^-$)

neighborhood

brah ~

② Graph Representation

Operations: find degree (# neighbors) of a vertex

Find if u and v are adjacent

Iterate over the neighbors of a given vertex

Iterate over all the edges of the graph

vertex are number 0 to n-1

Representations:

Edge List list of path work with both directed & undirected graphs

$\{(0,2), (0,3), (1,3), (2,0), (2,3), (3,0), (3,1), (3,2)\}$

Adjacency Matrix for undirected graph, the matrix is symmetric ($A^T = A$), diagonal is 0s

$A(G) = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$ ← matrix of binary values

$A_{i,j}$ is 1 iff $(i,j) \in E$

Adjacency Array keeps a array, the i-th array stores the neighbors of vertex i

```
int[][] a = {
    {2, 3},     // node 0's neighbor(s)
    {3},        // node 1's neighbor(s)
    {0, 3},     // node 2's neighbor(s)
    {0, 1, 2},  // node 3's neighbor(s)
}
```

for undirected, store each $\{v,w\}$ in both directions

Adjacency Map

```
Map<Vertex, Set<Vertex>> graph;

graph = {
    0: {2,3},
    1: {3},
    2: {0,3},
    3: {0,1,2}
}
```

generic type

we can use both HashMap and TreeMap — running time different

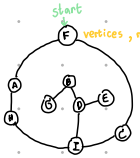
Code 13.1: An interface to represent an undirected graph.

```
1  public interface UndirectedGraph<Vertex> {
2      int numEdges(); /** How many edges? */
3      int numVertices(); /** How many vertices? */
4      int deg(Vertex v); /** Return the degree of v */
5
6      /** Return an iterable of vertices adjacent to v */
7      Iterable<Vertex> adj(Vertex v); who is the neighbors of v?
8
9      /** Is there an edge between u and v? */
10     boolean isEdge(Vertex u, Vertex v);
11
12     /** Add a new vertex */
13     void addVertex(Vertex v);
14
15     /** Add an edge between u and v */
16     void addEdge(Vertex u, Vertex v);
17
18     /** Remove an edge */
19     void removeEdge(Vertex u, Vertex v);
20 }
```

A.removeAll(B)

$\Theta(\min(|A|,|B|))$

L20: Graph Traversals



vertices, nodes

Breadth-First Search (BFS)

What if we want to know

- Shortest path from $s \to t$
- What can be reached from s?

⋮ and more

i	Vtxes that can be reached using exactly i edges (F_i)	frontier
0	F (self)	$F_0 = \{F\}$
1	A, B, C (neighbor of F)	$F_1 = N(F_0) \setminus F_0$ (subtract set of F_0)
2	~~F~~, H, G, D, I (neighbor of A, B, C; ~~F~~ have seen b4 above)	$F_2 = N(F_1) \setminus (F_0 \cup F_1)$
3	~~I~~, ~~A~~, ~~B~~, E, ~~H~~, ~~C~~ (treat like a set we won't duplicate)	$F_3 = N(F_2) \setminus (F_0 \cup F_1 \cup F_2)$
4	~~D~~ ∅	$F_4 =$ ⋮

Start: $F_0 = \{s\}$

X_i = "seen" = crossed out" = reached $\le i$ edges

$F_{i+1} = N(F_i) \setminus (F_0 \cup F_1 \cup \dots \cup F_i)$

$X_{i+1} = X_i \cup F_{i+1}$

```
def bfs(G,s):                          # starting vtx
    F0 = {s}, X0 = {s}, i=0
    while Fi ≠ ∅:
        Fi+1 = N(Fi) \ Xi
        Xi+1 = Xi ∪ Fi+1
        i++
    return Xi                          # everything I've seen so far
```

vertice can appear at most once: $F_0, F_1, F_2, \dots, F_d \to \emptyset$

```
interface UndirectedGraph<Vertex> {...}

public class Lecture20 {

    public Set<Integer> bfs(UndirectedGraph<Integer> G, int s) {
        Set<Integer> frontier = new HashSet<>(List.of(s));
        Set<Integer> visited = new HashSet<>(List.of(s));
        while (!frontier.isEmpty()) {
            // frontier = N(frontier) - visited        Fi → Fi+1
            frontier = nbrs(G, frontier);   → Σ_{v∈Fi} (1+deg(v))
            frontier.removeAll(visited);    → ≤ bounded by nbrs
            // visited = visited ∪ frontier
            visited.addAll(frontier);       → ≤ "
        }return visited;
    }
}
```

$3 \sum_{v \in F_i} (1+\deg(v))$

time $= \sum_{i=0}^{d} 3 \sum_{v \in F_i} (1+\deg(v)) \le \sum_{v \in V} (1+\deg(v)) = \sum_{v \in V} (1) + \sum_{v \in V} \deg(v)$ (upper bound)

$|V| = n$, $2m$

$= n + 2m$

$= O(n+m)$ # vertices, edges #

Ex: 

$\sum \deg(v) = 8 = 2(4)$

// $N(\{A,B,C\}) \subseteq N(A) \cup N(B) \cup N(C)$

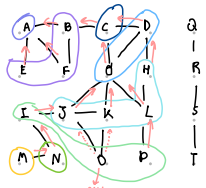
```
Set<Integer> nbrs(UndirectedGraph<Integer> G, Set<Integer> F){
    Set<Integer> nbrs = new HashSet<>();
    for (Integer src : F) {
        for (Integer dest : G.adj(src)){
            nbrSet.add(dest);          // O(1)
        }
    }
    return nbrSet;
}
```

find adjacents of everything in the F

neighbor → $\deg(v)+1$ (deg(v) adj, 1 loop)

Lemma: nbrs(G,F) runs in time $O\left(\sum_{v \in F} (1+\deg(v))\right)$

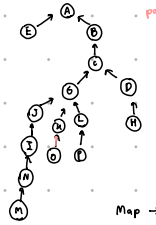
frontier



either from J or K for this one let's say K

remember where I came from

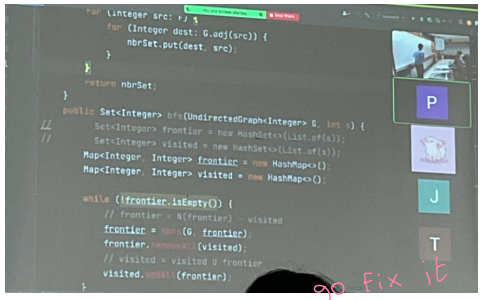
A bfs tree parent, child



Map → revers to tell others from top to bottom

fix!

```
Map<Integer, Integer> gbrsWithSource(Undir...<Integer> G, Set<Integer> F){
    Map<Integ, Integ> nbrset = new HashMap<>();
    for (Integer sc
```



go fix it (see go look @ the note)

S 12 :

Weighted Graphs and Representation

denoted by $G = (V, E, w)$ ← graph G

tgt with a weight function $w: E \to L$ ← set of possible annotations

every edge, $w(e)$ gives the label (i.e., weight) on that edge.

Representation in code :

- store on the side a Map from every edge to its label/weight
- but we wanna access the label/weight on an edge at the same time
- instead of Map < Vertex, set < vertex > > unweighted case

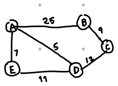
  we'll Map < Vertex, Map < Vertex, Label > >

  From each vertex to map storing for each neighbor the label/weight associated with that edge

- if G = map , G.get(v) ↳ Map < vertex, Label >,

  keys = nbrs of v

  value for the key v = labels/weights corresponding to the edge uv



```
G = {
    "A": {"B": 25, "D": 5, "E": 7},
    "B": {"A": 25, "C": 9},
    "C": {"B": 9, "D": 12},
    "D": {"A": 5, "C": 12, "E": 11},
    "E": {"A": 7, "D": 11}
}
```

in form of Map < String, Map < String, Integer > >

nbrs of E = A D   the edge DC weigh 12

An Interface for Weighted Undirected Graph

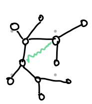
Similar to undirected Graph but

- make adj return a Map < Vertex, Label > instead of Iterable < vertex >
- add a method label getWeight (Vertex u, Vertex v)

Shorted Path Exercise :

From → if same then link! ← End

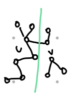
Extra : Prove the Light-Edge Rule



T + e ⇒ cycle

T − f → disconn. → 2 Trees

A FSOC the min-weighted edge (e = {u, v} smallest edge) across cut is not in the MST



T' = T + e − f (f: Tree edge across the cut)

Claim: T' is a spanning tree

$w(T') = w(T) - w(f) + w(e)$ (largest, small)

$w(T') < w(T)$

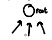
Lect 21 : Minimum Spanning Trees (MST)

Spanning Tree

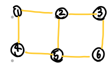
how many minimum # of edges inorder for everyone is able to reach everyone.
= 5 edges

Tree no cycles, called "spanning tree", n-1 edges

rooted tree

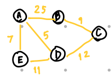


Claim: On n nodes, u need n-1 edges



There r many ways to draw spanning tree

Weighted Graph



Math : $G = (V, E) + w: E \to \text{Lable}$ (set; weight function; Lable: ℝ)

W(AE) ↳ 7

HOW TO STORE IT ?

Map < V, Set<V> > :

G = { A: {B, D, E},
B : {A, C},
C : {B, D},
D: {A, C, E},
E: {A, D}
}

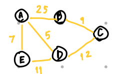
Map < V, Map < V, L > >

G = { A: {B: 25, D: 5, E: 7},
B: {A: 25, C: 9},
⋮
}

What's new :

- adj(v) ↳ Iterable → Map<V,L>
- getWeight (u,v)

MST



weights:

w ( ) = 25 + 9 + 12 + 11 = 57

w ( ) = 39

w ( ) = 33 cheapest!

What we've learnt :

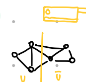
① $w(T) = \sum_{e \in E(T)} w(e)$

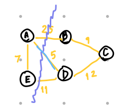
② not all spanning tree is equal

Lemma : There's a unique MST of G provided G is connected & has distinct weights. (no duplicate weights)

Cut

$(u, \bar{u})$ → $V \setminus u$: everything but u





$u = \{A, E\}$

$\bar{u} = V \setminus u = \{B, D, C\}$

$E(u, \bar{u}) = \{AB, AD, DE\}$ one pt in u, one pt in $\bar{u}$

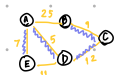
edges across the cut → Edges that got cut

w ( ) = 33

Light-Edge Rule

Thm For any $(u, \bar{u})$, the min-weighted edge across the cut is in the MST

Prim's Algoritm



apply light-Edge Rule

$u_0 = \{C\}$ − BC (light-weight in MST)

$u_1 = \{B, C\}$ − DC

$u_2 = \{B, C, D\}$ − AD

$u_3 = \{A, B, C, D\}$ − AE

MST = E → A → D → C → B

Steps:

① Pick an arbitrary starting vtx s.

② $u_0 = \{s\}$, $T = \emptyset$ no edges

③ For i = 1, 2, 3 …, n-1 :

- Apply L·E·R (lighted-edge rule)

  $(u_{i-1}, \bar{u}_{i-1})$ ↳ $e = \{x, y\}$

- $u_i = u_{i-1} \cup \{x, y\}$ & $T = T \cup \{e\}$

  add but just 1 bc one of them is already in u_{i-1}

if u implement Prim's Priority Queue → m log n (edge)

Kruskal's → prefered in class

why it fits light edge Rule

0. $T = \emptyset$, D = DisjointSet ( )

1. Sort the edges, so $w(e_1) < w(e_2) < \dots < w(e_m)$ (small weight) ← fastest sort $O(m \log m)$

2. For $i = 1, 2, 3, \dots, m$: → m times
   - let $\{x, y\} = e_i$
   - if x can't reach y in T : if (!D.isConnected(x,y)) } log n
     (all vertices reachable from x using T alone; consider $(U_x, \bar{U})$)
     - Add e_i to T → is L·E·R

       D.link(x,y)

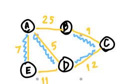
The time = $O(m \log m + m \log n)$

$O(m \log m) = O(m \log n)$ ($m \to n^2$)

$m \leq \binom{n}{2} \leq n^2$

nothing

AD → AE → BC → DE → DC → AB



Is there a way go to E → D that already in blue
if yes then skip, if not it's a cycle

L22 : Randomized Algorithms

How can we analyze the running time of the code that incoperates random?

As math :

Y = # of tosses til H

$$E[Y] = \sum_{k=1}^{\infty} k \cdot \Pr[Y=k] \quad (1-p)^{k-1} p$$

$$= \sum_{k=1}^{\infty} k(1-p)^{k-1} \cdot p$$

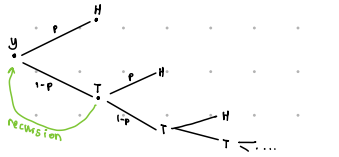
$$= p \sum_{k=1}^{\infty} k(1-p)^{k-1}$$

$$= p\left(\frac{1}{p^2}\right) = \frac{1}{p}$$

Y	
H	1 → p
TH	3 → $(1-p)^2 p$
TTH	6 → $(1-p)^3 p$
⋮	
$T^{n^6}H$	10^6+1 → $(1-p)^{10^6} p$
$T^k H$	$(1-p)^k p$

← geometric Random Variable

As CS:



2 cases: H (1 toss) T (1 toss + starts over)

$$E[Y] = p \cdot 1 + (1-p)[1 + E[Y]]$$

let $x = E[Y]$:

$$x = p + (1-p)[1+x]$$

let's solve for x:

$$x = p + 1 + x - p - px$$

$$x = 1 + (1-p)x$$

$$1 = x - (1-p)x$$

$$1 = px$$

$$x = \frac{1}{p}$$

$$E[Y] = \frac{1}{p} \#$$

Randomized Quick Sort (randomly pick the pivot)

```
def qs(a):
    if |a| ≤ 1 : return a
    p ← random choice from a        O(1)
    lt, eq, gt ← <p, ==p, >p        O(n)
    return qs(lt) + eq + qs(gt)
```

$T(n) = O(n) + T(\text{left}) + T(\text{rgt})$ (don't know "how big" it's — it's random; left: k, rgt: $n-k-1$)

$$E[T(n)] = n + \sum_{k=0}^{n-1} \Pr[|lt| = k]\left(E[T(k)] + E[T(n-k-1)]\right)$$

$\frac{1}{n}$ by the lemma below

$E[f(x)] = \sum \Pr[x=k] f(k)$

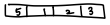
$$= n + \frac{1}{n} \sum_{k=0}^{n-1} \left[E[T(k)] + E[T(n-k-1)]\right]$$

$0,1,2,\ldots,n-1$ super $n-1, n-2, n-3, \ldots, 0$

$$= n + \frac{2}{n} \sum_{k=0}^{n-1} E[T(k)]$$

if $f(n) = E[T(n)]$

$$f(n) = n + \frac{2}{n} \sum_{k=0}^{n-1} f(k) \xrightarrow{A_s} f(n) = O(n \log_2 n)$$



$\Pr[|lt| = k] = ?$

	pivot	lt	eq	gt
$p = 1/n$	1	∅	1	2,3,5
$1/n$	2	1	2	3,5
$1/n$	3	1,2	3	5
$1/n$	5	1,2,3	5	∅

Lemma: For $k = 0,1,2,\ldots,n-1$, $|lt| = k$ w.p. $1/n$

```
input:   int a[]   → unique
         k : int
output:  k-th smallest # in a
```

```
def qs(a, k):
    if |a| ≤ 1 : return a[0]
    p ← random choice from a        O(1)
    lt, eq, gt → <p, ==p, >p        O(n)
    if |lt| ≥ k : return qs(lt, k)
    elif len(lt)+1 == k : return p   same to eq
    else: return qs(gt, k-|lt|-1)
```


Sorted(a)[k] — $n \log n$

PQ — $n \log k$

Qselect — n

if lucky $X_n \le \frac{3n}{4}$

$$T(n) = O(n) + \max\left(T(|lt|), T(|gt|)\right)$$

$$E[T(n)] = E[O(n) + \max^{k}\left(T(|lt|), T(|gt|)^{n-k-1}\right)]$$ (X_n)

$$= \text{lucky case} + \text{o/w}^{\text{pessimistic}}$$

$$= \frac{1}{2}\left(n + E[T(\tfrac{3n}{4})]\right) + \frac{1}{2}\left(n + E[T(n)]\right)$$

let $g(n) = E[T(n)]$

$$g(n) = \left[n + \frac{1}{2} g\left(\frac{3n}{4}\right)\right] + \left[\frac{1}{2} g(n)\right]$$

$$g(n) = g\left(\frac{3n}{4}\right) + 2n$$

$$= 2n + 2\left(\frac{3n}{4}\right) + 2 \cdot \frac{3}{4} \cdot \frac{3}{4} \cdot n + 2 \cdot \left(\frac{3}{4}\right)^3 \cdot n + 2 \cdot \left(\frac{3}{4}\right)^4 n + \ldots$$

$$= 2n\left[1 + \frac{3}{4} + \left(\frac{3}{4}\right)^2 + \left(\frac{3}{4}\right)^4 + \ldots\right]$$

$$= 2n\left[\frac{1}{1 - 3/4}\right]$$

$$= 2n[4]$$

$$= 8n$$

$O(n)$ ✗

recall: $\frac{1}{1-x} = 1 + x + x^2 + x^3 + \ldots$

lemma: $\Pr[|lt| \le \frac{3n}{4}] \approx \frac{1}{2}$ *

- $X_n = \max(|lt|, |gt|)$
- $X_n \le \frac{3n}{4} \iff \frac{n}{4} \le |lt| \le \frac{3n}{4}$ *

$$\Pr[X_n \le \tfrac{3n}{4}] = \Pr[\tfrac{n}{4} \le |lt| \le \tfrac{3n}{4}]$$

$$= \sum_{k=\frac{n}{4}}^{\frac{3n}{4}} \Pr[|lt| = k] \quad \text{by lemma: } k = 0,1,2,\ldots,n-1,\ |lt|=k \text{ w.p. } \tfrac{1}{n}$$

$$= \frac{1}{n} \cdot \left(\tfrac{3n}{4} - \tfrac{n}{4}\right)$$

$$= \frac{1}{2}$$

Lab: HashMap

Map<K, V> — HashMap / TreeMap (balance → BBST)

cousin

Set<K> just keyset

What if:

keys are small ints 0 .... H-1 (10,000)

Map<SI, String>

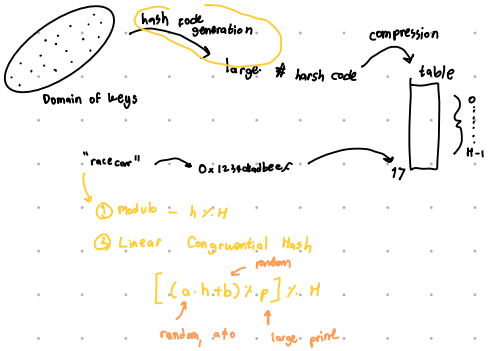
```
class SmallIntToStringMap {
    final const int H = 10,000;
    String[] table = new String[H];
    void put(int k, String v){ table[k] = v; }   O(1)
    String get(int k){ return table[k]; }        O(1)
}
```

Challenges:

① K is not always int

② Not always small int

HashMap (HashTable)



① modulo — h % H

② Linear Congruential Hash

[(a·h+b) % p] % H (a: random, ato; p: large prime)

Hash code generation

goal: unique 2 diff keys → diff. codes

"evil" = 69+118+105+108 = 400

"live" = 400

Polynomial hash code a = 42

"evil" = $69a^3 + 118a^2 + 105a^1 + 108a^0$

.hashCode is the default ↺ the memory address of the object

If a.equals(b), expect a.hashCode() == b.hashCode()

We want to overwrite .hashMap

Map<K, V>

init table = new V[2] → we'll resize it

array doubling trick. len(table) ≈ # of entries → .H ∼ Θ(n)

get(k) ① h = k.hashCode()

② i = compress(h)

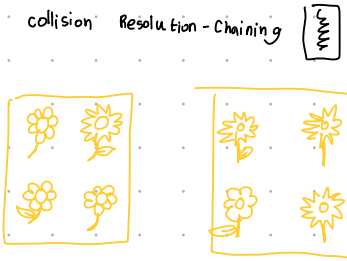
③ ~~linear search~~ return table[i]

put(k, v) ① h = k.hashCode() ② i = compress(h) } O(1)

③ table[i] = ~~v~~; linear search & update O(n)

O(# of entries in table i)

collision Resolution - Chaining



Simple Uniform Hashing Assumption (SUHA)

A hash function h uniformly distributes keys among integers 0... H-1

$Pr[h(k) = i] = \frac{1}{H}$

X_i = # of keys hashed to table[i]

o = no, i = yes

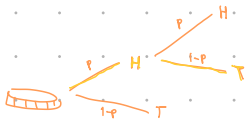
keys = {$k_1^0, k_2^1, \ldots, k_n^1$}

$X_i = X_{i,1} + X_{i,2} + X_{i,3} + \ldots + X_{i,n}$

$E[X_i] = E[\sum_{k=1}^{n} X_{i,k}]$

$= \sum_{k=1}^{n} E[X_{i,k}] = \frac{n}{H}$ ($E[X_{i,k}]$: $\frac{1}{H}$; $E[X_{i,j}] = 1 \cdot \frac{1}{H} + 0$)

$= 2$



Task 1: Breadth-First Search Running Time (2 points)

Our breadth-first (BFS) implementation from class uses `HashSets` and `HashMaps` to keep track of progress. In this task, you'll reanalyze the running time of BFS if we use `TreeSets` and `TreeMaps` instead. Show your work carefully. To keep things simple, we'll work with the BFS code shown above. It only reports vertices reachable from a source s. It has been rewritten from the version presented in class to make the operations more explicit.

```
Set<Integer> nbrsExcluding(
    UndirectedGraph<Integer> G,
    Set<Integer> vtxes,
    Set<Integer> excl
  ) {
    Set<Integer> union = new TreeSet<>(); // not HashMap
    for (Integer src : vtxes) {
        for (Integer dst : G.adj(src))
          if (!excl.contains(dst)) { union.add(dst); }
    }
    return union;
}
Set<Integer> bfs(UndirectedGraph<Integer> G, int s) {
    Set<Integer> frontier = new TreeSet<>(Arrays.asList(s));
    Set<Integer> visited = new TreeSet<>(Arrays.asList(s));

    while (!frontier.isEmpty()) {
        frontier = nbrsExcluding(G, frontier, visited);
        visited.addAll(frontier); // the i-th position is what's reached at i hops
    }

    return visited;
}
```

You should know that the `TreeSet` implementation uses a balanced binary search tree (BST), so `add`, `contains`, and `remove`, unlike in a `HashSet`, take $O(\log S)$ per operation, where S is the size of the collection. Though `addAll` may perform other optimizations, for the purpose of this task, assume that `addAll(X)` simply repeatedly calls `add` on each element of X.

You should also assume that `UndirectedGraph G` is implemented as an adjacency table using `HashMaps`, so `G.adj` takes $O(1)$.

Task 2: Mathematical Facts (2 points)

Read me! Prove only one of the following:

(i) Remember that $[n]$ is the set $\{1, 2, \ldots, n\}$. Let $p : [n] \to [n]$ be a random permutation on $[n]$ chosen uniformly among the $n!$ permutations. Consider the following code:

```
int minSoFar = Integer.MAX_VALUE;
int numUpdate = 0;
for (int i=1;i<=n;i++) {
    if (p(i) < minSoFar) {
        minSoFar = p(i);
        numUpdate++;
    }
}
```

Prove that at the end of the **for**-loop,

$$\ln(n+1) \le \mathbf{E}[\texttt{numUpdate}] \le 1 + \ln n.$$

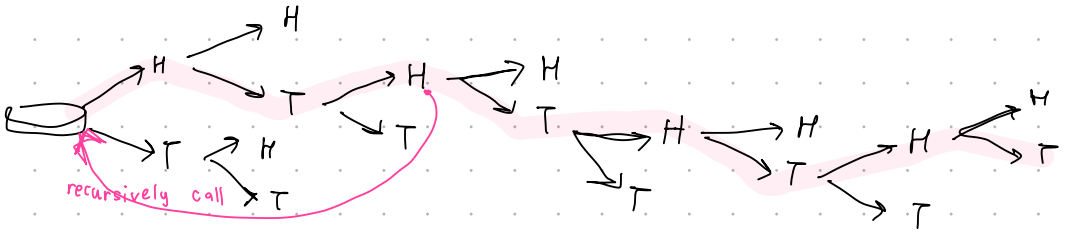
(ii) Let $G = (V, E, w)$ be an undirected connected weighted graph with distinct edge weights. Show that G has a unique minimum spanning tree.

1. You have a coin that turns up heads with probability p. How many tosses do you need in expectation before you encounter the pattern `HT` consecutively (including the tosses `HT` themselves)?

2. A fancy search algorithm is shown below. Suppose a key `k` is present in the array `a`, which contains unique numbers. What is the running time of this algorithm in expectation?

```
def fancy_search(a: List[int], k: int) -> int:
    random_loc = pick a number between 0 and len(a) - 1 uniformly at random
    while a[random_loc] != k:
        random_loc = pick a number between 0 and len(a) - 1 uniformly at random
    return random_loc
```

O(n)



$E[x] = 2p + 2(1-p) + p[2 + E[x]]$

let x be $E[x]$

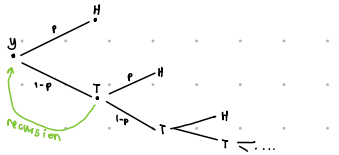
$x = 2p + 2 - 2p + 2p + xp$

$x = 2 + 2p + xp$

$x - xp = 2 + 2p$

$x[1-p] = 2 + 2p$

$x = \frac{2+2p}{1-p}$



2 cases : H T

$E[Y] = p \cdot 1 + (1-p)[1 + E[Y]]$

1 toss ; 1 toss + start over

$E[x] = p + (1-p) +$

เปิดโหมดน้ำ
สูตรได้ใบ

ใบ ชมวง + กระเพรา + ห่วยหอม + พริก + ปลาซ่องเกร็ด นึ่ง
ตัดใบชมวง ถึงเปล่าๆ ก็ได้
กลิ่นยน้ำร้า