香港中文大學（深圳）
The Chinese University of Hong Kong

CSC3100 Data Structures
Course information

Li Jiang
School of Data Science (SDS)
The Chinese University of Hong Kong, Shenzhen

About instructor (Li Jiang)

- Background
  - PhD: 2017-2021, CUHK
  - Postdoc: 2021-2023, MPI-INF
  - Assistant Professor: 2024-present, CUHK-SZ

Prof. Li Jiang

- Contact
  - Email: jiangli@cuhk.edu.cn
  - Office: Teaching Complex C610

- Research
  - General area: computer vision, artificial intelligence
  - Topics: 3D scene understanding, autonomous driving, etc.

Lecture location and time

- Two sessions in total
  - Leading instructor: Prof. Yixiang Fang

CSC 3100 - Data Structures

Section	L01-LEC(3504)	**Enrollment Total**	98	**Status**	
Session	Regular	**Enrollment Capacity**	100		

Days & Times	Room	Instructor	Meeting Dates
MoWe 13:30 - 14:50	Teaching D Bldg 103	Li JIANG	2024/01/08 - 2024/05/10

Section	L02-LEC(3505)	**Enrollment Total**	80	**Status**	
Session	Regular	**Enrollment Capacity**	100		

Days & Times	Room	Instructor	Meeting Dates
MoWe 15:30 - 16:50	Teaching Complex C302	Yixiang Fang	2024/01/08 - 2024/05/10

Prof. Yixiang Fang

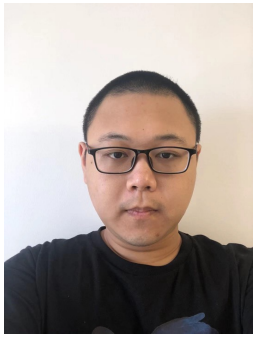
Lecture location and time

- Session by Li Jiang
  - Lecture location: Room 103, Teaching D Building
  - Lecture time (3 hours/week)
    - 13:30 pm ~ 14:50 pm on Monday
    - 13:30 pm ~ 14:50 pm on Wednesday
  - Office hour
    - Time: 4:00 pm ~ 5:00 pm on every Monday
    - Location: Room 610, Teaching Complex C

TA information

- 7 TAs
  - Leading TA: Mr. Shu Wang
  - Host tutorials; assignments; exams; office hour


Shu Wang


Chunxu Lin


Yingli Zhou


Ziyi Zhao


Yuyang Xia


Yige Jiang


Xuanzhuo Liu

Office hours of TAs

- Office hours of TAs
  - Thursday 14:00-16:00, pending
  - Friday 14:00-16:00, pending

ID	Name	PGTA/USTF	Email address
1	Shu Wang (leading TA)	PGTA	shuwang3@link.cuhk.edu.cn
2	Chunxu Lin	PGTA	chunxulin1@link.cuhk.edu.cn
3	Yingli Zhou	PGTA	yinglizhou@link.cuhk.edu.cn
4	Yige Jiang	USTF	yigejiang@link.cuhk.edu.cn
5	Ziyi Zhao	USTF	ziyizhao2@link.cuhk.edu.cn
6	Yuyang Xia	USTF	yuyangxia@link.cuhk.edu.cn
7	Xuanzhuo Liu	USTF	xuanzhuoliu@link.cuhk.edu.cn

Tutorials

- Tutorials
  - Check announcements in your e-mail box or Blackboard
  - Check Wechat group messages

Group: Spring-CSC3100

Valid until 1/14 and will update upon joining group

Tutorial time	Monday	Tuesday	Wednesday	Thursday	Friday
18:00 - 18:50			~~TA305~~		
19:00 - 19:50		TA305	TA305		
20:00 - 20:50		TA305			

Tutorials

- Tutorial arrangement

Id	Content	PGTF/USTF	Instructor
1	JDK environment	Shu Wang	Yixiang Fang
2	Sorting	Yige Jiang	Yixiang Fang
3	Insertion Sort	Ziyi Zhao	Yixiang Fang
4	Linked List	Shu Wang	Yixiang Fang
5	Stack	Xuanzhu Liu	Yixiang Fang
6	Queue	Ziyi Zhao	Yixiang Fang
7	Binary Search Trees	Chunxu Lin	Li Jiang
8	AVL Trees	Xuanzhu Liu	Li Jiang
9	Heap	Yuyang Xia	Li Jiang
10	Hash	Chunxu Lin	Li Jiang
11	BFS & DFS of Graphs	Yige Jiang	Li Jiang
12	Dijkstra & Floyd	Yuyang Xia	Li Jiang

Assessment

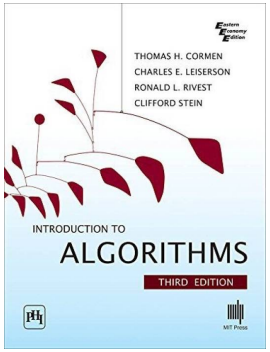
Homework 39% + Midterm exam 20% + Final exam 40% + CTE 1%

	Release date	Due date	Weight	Instructor	TAs
Programming assignment 1	Jan 24	Feb 7	9%	Yixiang Fang	Yingli Zhou Yuyang Xia
Programming assignment 2	Mar 6	Mar 20	10%	Li Jiang	Yingli Zhou Xuanzhuo Liu
Midterm exam	Mar 23 ?	--	20%	Yixiang Fang	Shu Wang Chunxu Lin
Programming assignment 3	Mar 27	Apr 10	10%	Li Jiang	Yingli Zhou Yige Jiang
Programming assignment 4	Apr 17	May 1	10%	Yixiang Fang	Yingli Zhou Ziyi Zhao
Final exam	May 11 – 18 ?		40%	Li Jiang	Shu Wang Chunxu Lin

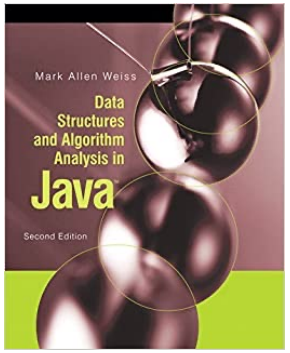
* For assignments, students can use Python/C/C++/Java to answer the questions in Online Judge (OJ) system
* OJ system url: http://oj.cuhk.edu.cn/

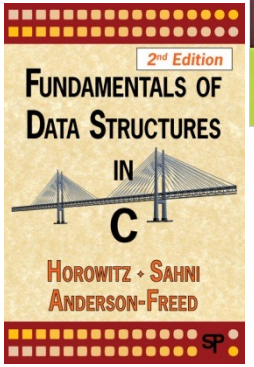
Textbook and references

- Textbook
  - T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, 3rd Edition, The MIT Press, 2009.
  - https://edutechlearners.com/download/Introduction_to_algorithms-3rd%20Edition.pdf



- References
  - M.A. Weiss, Data Structures and Algorithm Analysis in Java, 2nd Edition, Addison-Wesley, 2007.
  - Ellis Horowitz, Sartaj Sahni, Susan Anderson-Freed, Fundamentals of Data Structures in C.





Teaching methods

- Lectures (3 hours/week)
  - Important materials from the textbook will be covered
  - Please ensure you stay up with the materials
  - Java will be used for illustrating the data structures
  - Feel free to interrupt to ask questions

- Tutorials (1 hour/week)
  - Illustrate more on the difficult parts
  - Show students more examples
  - Discuss assignment/exercise

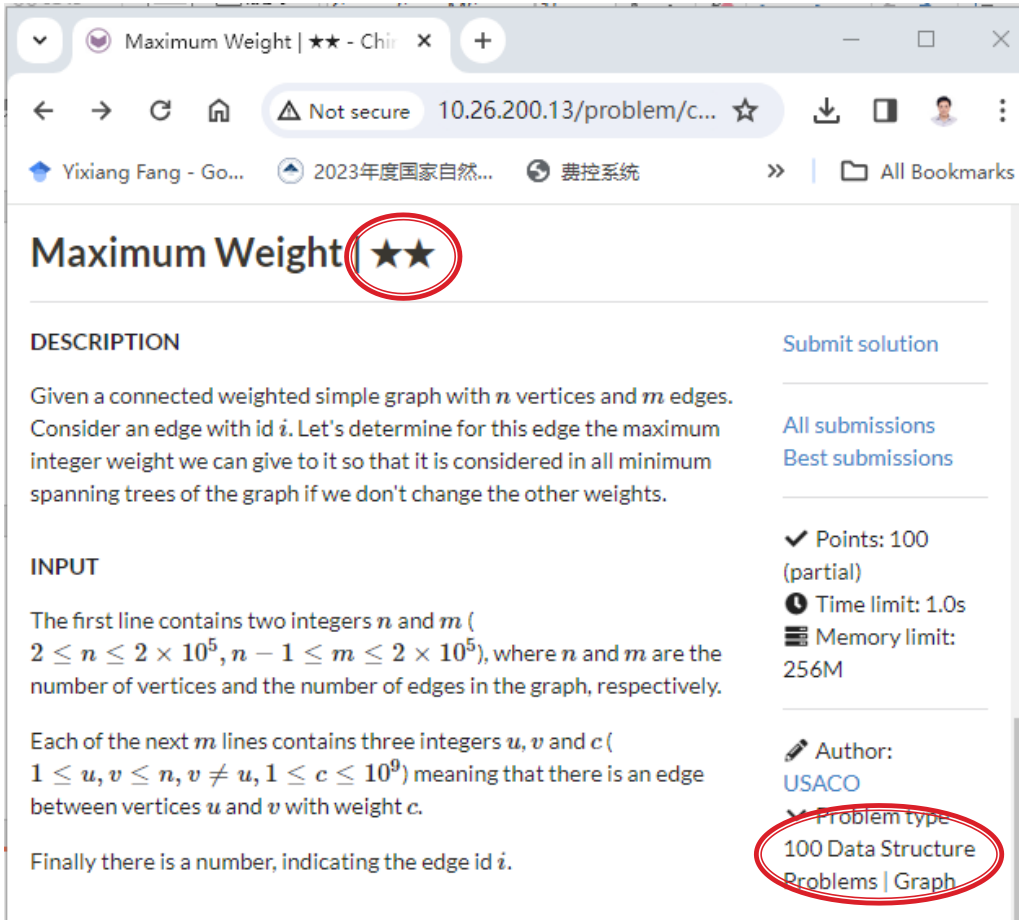
How to do well in this course?

- Common suggestions
  - Attend the lectures physically
  - Slides of lectures/tutorials will be uploaded to Blackboard before lectures/tutorials; learn them in advance
  - Use examples to facilitate learning
  - Data structures are not just reading materials; you need to write more codes!!!
- Special suggestions
  - If you feel difficult,
    - Try to focus on the content of slides and seek for help (TAs and instructors are ready to answer your questions!)
  - If you feel easy,
    - Read more details (e.g., theoretical analysis) in the textbook
    - Use the learned techniques to solve ACM-ICPC problems

How to do well in this course?

- 100 Data Structure Problems in OJ
  - */**/*** refer to easy/normal/difficult
  - 8 categories in total

Category	Number
Array/Linked List	12
Stack/Queue	12
Tree (BST, Heap)	16
Graph (BFS, DFS, MST, SP)	16
Sorting	10
Hashing	8
Recursion/Divide-and-conquer	16
Strings	10



Acknowledgement:

Tianci Hou, Yohandi, Ziyi Zhao, Letian Cheng, Yuzhou Lin, Yutian Wang Yiyu Ren, Frederick Khasanto

Learn more and practice more

- ACM-ICPC
  - ACM: Association for Computing Machinery
  - ICPC: International Collegiate Programming Contest

- Online Judge (OJ) systems
  - Many programming problems
  - CUHKSZ OJ (campus only)
    - http://oj.cuhk.edu.cn/
  - SJTU OJ
    - https://acm.sjtu.edu.cn/OnlineJudge/
  - PKU OJ
    - http://poj.org/

CUHK-SZ programming team

- Coaches: Yixiang Fang, Chenhao Ma
- Student coach: Yuyang Xia

- The first CUHK-SZ programming contest
  - Time: April 2, 2023
  - Duration (hours): 4
  - Questions: 12
  - Participates: 235
  - Awards: 45
  - Bonus (RMB): 50+K
  - Sponsor: Bopu

News of this contest: https://mp.weixin.qq.com/s/EbRwux0HupSyN8FfJfY6Bg

CUHK-SZ programming contest

Bonus of programming contest

- Students who got awards (e.g., First/Second/Third Prizes) in CUHK-SZ programming contest, 广东省省赛, ICPC/CCPC, NOI/ NOIP
  - Bonus: exemption from 4 programming assignments
  - Duty: be the tutorial helpers; presentation in tutorials and answer students' questions
  - If you are interested in the bonus above, please send an email to the leading TA with supporting materials (e.g., award certificates) before Jan 14, 2024
    - Mr. Shu Wang, shuwang3@link.cuhk.edu.cn

Course policy

- Your work MUST be your own
  - Cheating is against "fair-play" and will not be tolerated under any circumstances

- Assignments
  - Penalty will be imposed on copying assignments; minimum penalty is zero mark for the assignments
  - There will be penalty for late submission:
    - 0~24 hours after deadline: final score = your score x 0.8
    - 24~72 hours after deadline: final score = your score x 0.5
    - 72+ hours after deadline: final score = your score x 0

Examples of code plagiarism

← 返回 | 代码对比

[redacted].cpp 与 1[redacted].cpp 相似度为 100%

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,a[100100],r[100100];
long long ans=0;
void mergesort(int s,int t)
{
 if(s==t) return ;
 int mid=(s+t)/2;
 mergesort(s,mid);
 mergesort(mid+1,t);
 int i=s,j=mid+1,k=s;
 while(i<=mid&&j<=t)
 {
  if(a[i]<a[j]) r[k++]=a[i++];
  else r[k++]=a[j++],ans+=mid-i+1;
 }
 while(i<=mid) r[k++]=a[i++];
 while(j<=t) r[k++]=a[j++];
 for(int i=s;i<=t;++i)
 a[i]=r[i];
}

int main()
{
 scanf("%d",&n);
 for(int i=1;i<=n;++i)
 scanf("%d",&a[i]);
 mergesort(1,n);
 printf("%lld\n",ans);
```

```cpp
#include<bits/stdc++.h>
using namespace std;
int n,a[500500],r[500500];
long long ans=0;
void msort(int s,int t)
{
        if(s==t) return ;
        int mid=(s+t)/2;
        msort(s,mid);msort(mid+1,t);
        int i=s,j=mid+1,k=s;
        while(i<=mid&&j<=t)
        {
                if(a[i]<a[j]) r[k++]=a[i++];
                else r[k++]=a[j++],ans+=mid-i+1;
        }
        while(i<=mid) r[k++]=a[i++];
        while(j<=t) r[k++]=a[j++];
        for(int i=s;i<=t;++i)
        a[i]=r[i];
}
int main()
{
        scanf("%d",&n);
        for(int i=1;i<=n;++i)
        scanf("%d",&a[i]);
        msort(1,n);
        printf("%lld\n",ans);
}
```

Examples of code plagiarism

Sample1.cpp 与 Sample2.cpp 相似度为 100%

```
#include<iostream>
using namespace std;

void foo1(){
// an easy function
  int a = 10;
  cout<<a;
}

void foo2(){
/* I
   am
   comments*/
  int b = 20;
  cout<<b;
}

int main(){
  foo1();
  foo2();
}
```

```
#include<iostream>
using namespace std;

void fun2(){
  int c=20; cout<<c;
}

void fun1(){
  int d=10; cout<<d;
}

int main(){
  fun1();
  fun2();
}
```

Changing the variable and function names, and re-organizing function orders are typical kinds of code plagiarism… We will use a software tool to detect code plagiarism automatically!

Don't send your source codes to others, which may make you be the one who is copied from! (抄和被炒都会有处罚，默认就是0分)

Students with diverse backgrounds

Year	Percentage
1st year	0%
2nd year	57.3%
3rd year	24.2%
4th year	18.5%
Overall	100%

- Student distribution (178)
  - SME (63): 35.4 %
    - Financial engineering: 56
    - Finance: 4
    - Economics: 1
    - Global Business Studies: 2
  - SSE (25): 14.0%
    - Electrical and Computer Engine: 4
    - Mathematics and Applied Mathem: 16
    - Electronic Info Engineering: 5
  - SDS (87): 48.9%
    - Data Science and Big Data Tech: 14
    - Computer Science and Engineering: 61
    - Statistics: 12
  - Others (3): 1.7%
    - Bioinformatics: 1
    - English Studies: 2

Please try to consider your classmates before complaining

Feedback & acknowledgements

- Feedback is important and also welcome!
  - Talk to course instructors and TAs, or send us emails
  - Please talk to us before complaining to others
  - Questionnaire

- Acknowledgements
  - Lecture slides
    - Prof. Tianshu Yu (CUHK-SZ), Prof. Xiang Wan (CUHK-SZ), Prof. Kaiming Shen (CUHK-SZ), Prof. Wenye Li (CUHK-SZ), Prof. Minming Li (CityU, HK), Prof. Zengfeng Huang (FDU), Prof. Jane You (PolyU, HK), Prof. Sibo Wang (CUHK)
  - Tutorial slides
    - Mr. Xingchao Wang (CUHK-SZ), Mr. Panwen Hu (CUHK-SZ), Mr. Ziteng Weng (CUHK-SZ)

香港中文大學（深圳）
The Chinese University of Hong Kong

CSC3100 Data Structures
Lecture 1: Introduction

Li Jiang
School of Data Science (SDS)
The Chinese University of Hong Kong, Shenzhen

Outline

- Introduction
  - Why take this course?
  - Basic concepts, e.g., abstract data type (ADT)
  - Relationship of ADT, data structures, and algorithms
  - Topics in this course
  - Tentative teaching plan

Why take this course?

- Required course
  - Also very important and very useful
  - A fundamental course in computer science
- Learn to save the data (data structure) and manipulate the data (algorithms) effectively and efficiently
- No single data structure fits all scenarios
  - Array: friendly to search (if sorted), not friendly to updates
  - List: friendly to updates, not friendly to search
- How useful in practice?
  - E.g., validate one Chinese ID in 1.4 billion people
  - E.g., find the best driving route
  - E.g., find webpages in Google

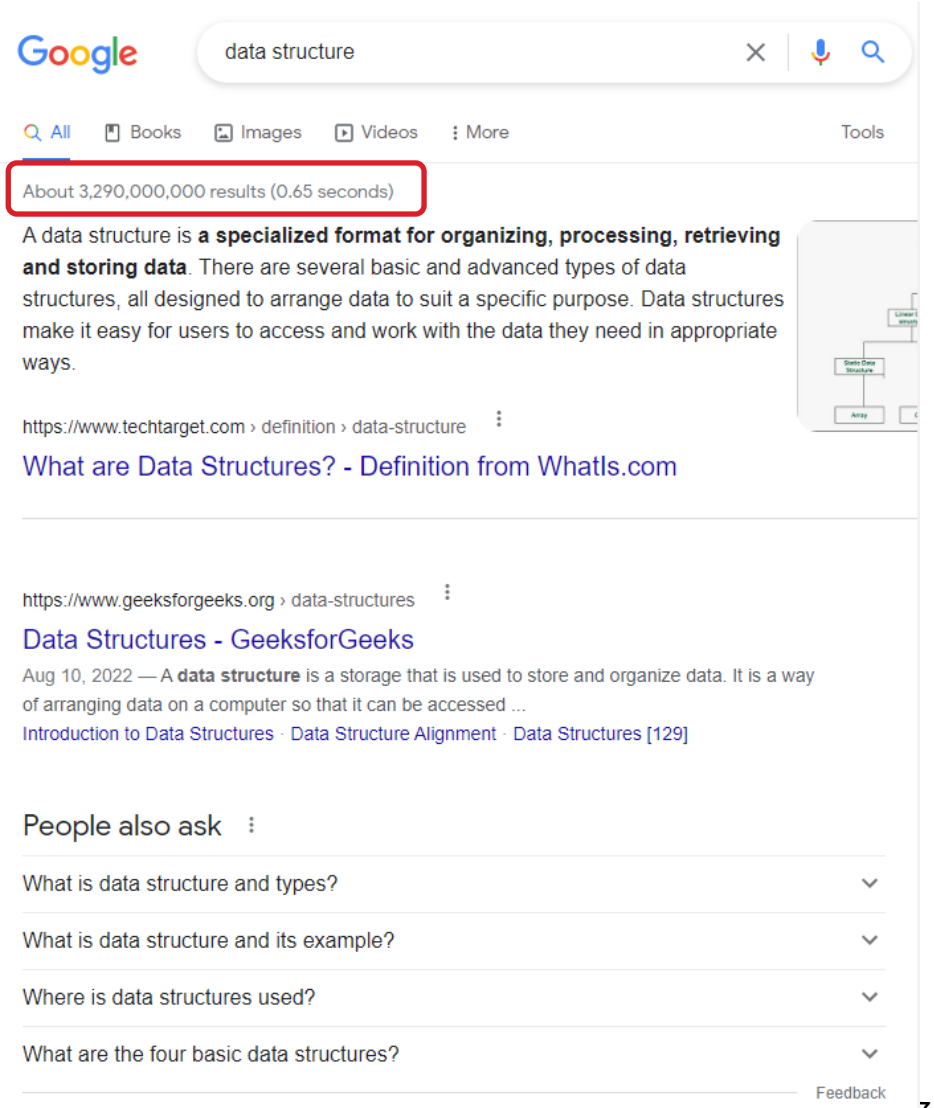
Real examples

- Route planning
  - Find the shortest path between two specific locations
  - Input:
    - A road network
    - A source node (location)
    - A destination node (location)
  - Output
    - A path, or a sequence of edges, with the shortest total distance

Real examples

- Google search
  - Find the documents matching your query keywords
  - Use sophisticated algorithms to create an index structure which is just a data structure

30 trillion (30×10^{12}) webpages



One sentence about this course

This course is about how to use computing resources and tools to solve practical problems **correctly** and **efficiently**

Logical thinking and math

Good at data structures, algorithms, and programming

Data abstraction

- A clear separation between
  - the abstract properties of a data type and
  - the concrete details of its implementation

- Example: smartphone
  - Users do not know
    - How calls are made (e.g., CPU, memory, battery, and electric wire)
    - How the phone accesses the Internet
    - How the data is stored in the phone
  - Users do know
    - To make a call: input a person's phone number
    - To access the Internet: open the browser

Abstract Data Type (ADT)

- An ADT is for encapsulation (information hiding)
  - The implementation of an ADT and its operations can be localized to one section of the program
  - Procedures that make use of the ADT can safely ignore its implementation details

- Benefit of ADT
  - User-friendly: users do not need to know the mechanisms of how to connect to the Internet
  - Designer-friendly: designer can change mechanisms without affecting users
  - Protection: others cannot know your secrets!

How to separate?

- An ADT only provides the definition of operations
  - It consists of names of every operation (function), the type of its arguments, and the type of its result

```
1  ADT IntegerSet
2   IntegerSet createEmptySet();
3   IntegerSet addElementToSet(integer, SetA);
4   Boolean search(integer, SetA);
5   IntegerSet intersection(setA, setB);
6   IntegerSet union(setA, setB);
7   IntegerSet difference(setA, setB);
```

  - Does not reveal the internal implementation details
    - How the set is represented? Array, List, Tree, etc.?
    - How the operations are implemented? Many different ways

Relationships: ADT, DS, algorithms

- A data structure (DS): implementation of an ADT
  - List ADT
    - Implementation: ArrayList or LinkedList
  - Set ADT
    - Implementation: Hash Table or Red-Black Tree

- An algorithm: implementation of operations in ADT

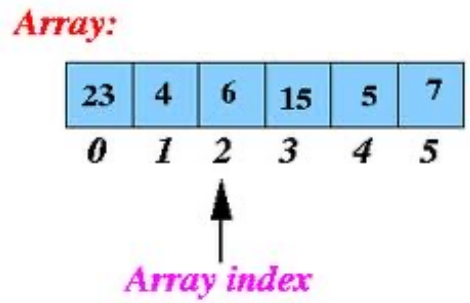
Input
A sequence of numbers: $a_1, a_2, \ldots a_n$

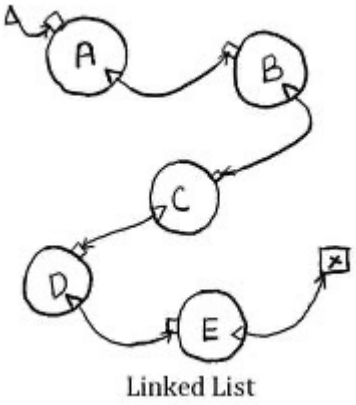
$\rightarrow$ Sorting Algorithm $\rightarrow$

Output
A permutation (reordering) $\langle a_1', a_2', \ldots a_n' \rangle$ of the input sequence, $a_1' \leq a_2' \leq \ldots \leq a_n'$

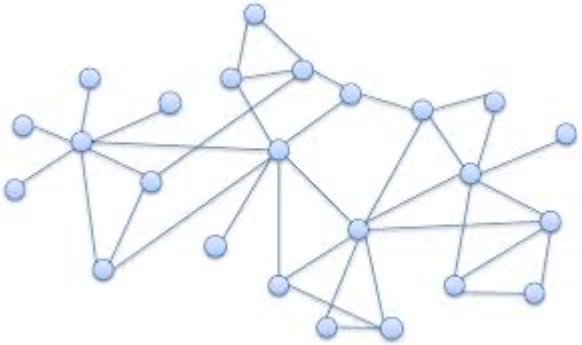
What are the common ADTs?

- Arrays, lists, stacks, queues, trees, graphs, etc.

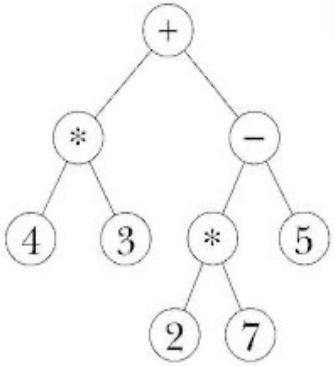




Linked List




Tree



Topics covered

- How to define "efficient"?
  - Time complexity analysis
- No single data structure fits all scenarios
  - Array: Efficient for sorting and search, not for updates
  - List: Efficient for updates, but not in searching/sorting
  - Stack/queues: Fast for certain types of updates, not in searching/sorting
  - Trees: Heap & binary search trees
  - Hash tables: Hash functions
  - Graphs: Graph algorithms
- Efficient operations
  - Sorting
  - Searching/Updates on different data structures (Discussed when learning the specific data structure)

Tentative teaching plan

*This tentative teaching plan is subject to change as the instructor sees fit.

Week	Content
1	Course overview, Java basics
2	Arrays, insertion/merge sort
3	Time complexity
4	List
5	Stack, queues
6	More sorting algorithms
7	Trees

Week	Content
8	Midterm exam
9	Trees
10	Hashing
11	Graphs
12	Graphs
13	Graphs, other data structures
14	Course review

Throughout this course, we will see:
- How to estimate the time cost of a program
- How to select proper data structure(s) to solve real problems
- Techniques to improve the speed of a program

Example: selection problem

- Problem
  - Given N numbers, determine the k^{th} largest, where N > k

- Solution 1:
  1) read N number into an array
  2) sort the array in descending order
  3) return the element in position k

- Solution 2:
  1) read the first k elements into an array and sort them in descending order
  2) each remaining element is read one by one,
     - 2.1) it is ignored if it is smaller than or equal to the k^{th} element in the array
     - 2.2) otherwise, it is placed in its correct spot in the array, bumping one element out of the array
  3) the element in the k^{th} position is returned as the answer

Example: selection problem

- Two natural questions:
  - Which solution is better ?
    - By simulation
    - By theoretical analysis
  - Is either algorithm good enough (particularly when N is very large)?
    - A simulation using 1 million elements and k = 500,000 will show that NEITHER algorithm finishes in a reasonable amount of time
    - Is there a better algorithm?
- What is the conclusion?
  - Writing a working program is not good enough!
  - The optimum solution: a correct algorithm with minimum resources and minimum running time

Exercise

- Is it possible to find a better algorithm to select the k^{th} largest number?

Mathematics review

- Exponents

$$X^A X^B = X^{A+B}$$

$$\frac{X^A}{X^B} = X^{A-B}$$

$$(X^A)^B = X^{AB}$$

$$X^N + X^N = 2X^N \neq X^{2N}$$

$$2^N + 2^N = 2^{N+1}$$

Mathematics review

- Logarithm definition:

$$X^A = B \text{ if and only if } \log_x B = A$$

- All log are to be base 2 unless specified otherwise

– *Useful equalities*

$$\log_A B = \frac{\log_c B}{\log_c A}; C > 0$$

$$\log AB = \log A + \log B$$

$$\log\left(A/B\right) = \log A - \log B$$

$$\log x < x, \quad \forall x > 0$$

$$\log 1 = 0, \log 2 = 1, \log 1024 = 10, \log 65536 = 16$$

Mathematics review

- Series: arithmetic series

$$\sum_{i=1}^{N} i = \frac{N(N+1)}{2}$$

$$\sum_{i=1}^{N} i^2 = \frac{N(N+1)(2N+1)}{6}$$

Example: To find the sum

2+5+8+….+ (3k-1)

= 3(1+2+3+…+k)

- (1+1+1+….+1)

$$= \frac{3k(k+1)}{2} - k$$

Mathematics review

- Series: geometric series

$$\sum_{i=0}^{N} A^i = \frac{1 - A^{N+1}}{1 - A}$$

$$If\ 0 < A < 1, then$$

$$\sum_{i=0}^{N} A^i \leq \frac{1}{1 - A}$$

- Derivation

Let S = $1+A+A^2+\ldots\ldots$ (1)

where, $0<A<1$

then $AS = A+A^2+A^3+\ldots$(2)

Subtracting (1) and (2), we get $S-AS <= 1$, i.e.

$$S \leq \frac{1}{1 - A}$$

Mathematics review

- To prove a false statement:
  - proof by counter example

e.g., Fibonacci number:

$F_0 = 1, F_1 = 1, F_{k+1} = F_k + F_{k-1}$

To show the statement $F_k \leq k^2$ is false, we can compute a concrete counter example, e.g.,

$F_{11} = 144 > 11^2$.

- To prove a correct statement
  - proof by induction
    - (1) proving a base case
    - (2) inductive hypothesis
  - proof by contradiction
  - (1) assume it is false
  - (2) show that this assumption is false