CS2010: ALGORITHMS AND DATA STRUCTURES

Lecture 1: Module Overview & Introduction

Vasileios Koutavas

School of Computer Science and Statistics
Objective: learn to solve computational problems efficiently

“Algorithms + Data Structures = Programs”
— Niklaus Wirth
→ **Algorithm**: The steps to **correctly** perform a task that answers a **general**¹ computational problem
→ **Algorithm:** The steps to **correctly** perform a task that answers a **general**\(^1\) computational problem

→ What is the median age of all people in Ireland?
→ **Algorithm**: The steps to *correctly* perform a task that answers a general\(^1\) computational problem

→ What is the median age of all people in Ireland?
→ What is the quickest path from here to Marrakech?
→ **Algorithm**: The steps to ***correctly*** perform a task that answers a **general**\(^1\) computational problem
→ What is the median age of all people in Ireland?
→ What is the quickest path from here to Marrakech?

\(^1\)Algorithms answering the above questions had better work when the population of Ireland changes & for other holiday destinations!
→ **Algorithm:** The steps to **correctly** perform a task that answers a general\(^1\) computational problem

- What is the median age of all people in Ireland?
- What is the quickest path from here to Marrakech?

→ **Data Structures:** The methods to store the information needed for the algorithm.

\(^{1}\)Algorithms answering the above questions had better work when the population of Ireland changes & for other holiday destinations!
Are these good measures of a program’s efficiency?

- How long it takes the program to run on my laptop
- How long it takes the program to run on the fastest computer
- How long it takes the program to run on the slowest computer
- How long it takes the program to run with the largest input
- How long it takes the program to run with the smallest input
Established measure: how well the program scales to larger inputs

→ When I double the input size my program takes the same time to run on the same computer (constant running time).
→ When I double the input size my program takes twice the time to run on the same computer (linear running time).
→ ...
**Established measure:** how well the program *scales* to larger inputs

→ When I *double* the input size my program takes the same time to run on the same computer (*constant running time*).

→ When I *double* the input size my program takes twice the time to run on the same computer (*linear running time*).

→ ...

We also care about **memory needed:**

→ When I *double* the input size my program needs the same amount of memory to run (*constant memory space*).

→ When I *double* the input size my program takes twice the amount of memory to run (*linear memory space*).

→ ...

4
→ Learn the most common A&DS that every CS graduate must know.
   → Example algorithms: Merge Sort, Union-Find, Dijkstra’s Shortest Path Tree, ...
→ Learn the most common A&DS that every CS graduate must know.
   → Example algorithms: Merge Sort, Union-Find, Dijkstra’s Shortest Path Tree, ...

→ Identify which known algorithms/data structures best fit specific problems
   → Example: What is the best algorithm for finding the median?
     – It depends:\(^2\) QuickSelect, MedianOfMedians, IntroSelect, using SoftHeaps

\(^2\)https://www.quora.com/What-is-the-most-efficient-algorithm-to-find-the-kth-smallest-element
Learn to **evaluate** new algorithms

- **Efficiency**: *calculate* the running time and memory usage
  - We are interested mostly on **how well they scale**
  - Worst-case, average-case, best-case, **amortised** performance
  - Experimentally measure performance
  - Calculate performance using mathematical methods (big-O notation, tilde notation, cost models, ...)

- **Correctness**: **rigorous testing** and some informal correctness arguments
→ Learn to **evaluate** new algorithms
  → Efficiency: **calculate** the running time and memory usage
    → We are interested mostly on **how well they scale**
    → **Worst-case**, average-case, best-case, **amortised** performance
    → Experimentally measure performance
    → Calculate performance using mathematical methods (big-O notation, tilde notation, cost models, ...)
  
  → Correctness: **rigorous testing** and some informal correctness arguments

→ Learn to use **Abstract Data Types**: the interfaces of Data Structures
Learn to evaluate new algorithms

- Efficiency: calculate the running time and memory usage
  - We are interested mostly on how well they scale
  - Worst-case, average-case, best-case, amortised performance
  - Experimentally measure performance
  - Calculate performance using mathematical methods (big-O notation, tilde notation, cost models, ...)

- Correctness: rigorous testing and some informal correctness arguments

Learn to use Abstract Data Types: the interfaces of Data Structures

Practice implementing A&DS in Java
CS2010 Logistics
Example Problem
A software engineer was asked to design an algorithm which will input two unsorted arrays of integers, A (of size N) and B (also of size N), and will output true when all integers in A are present in B. The engineer came up with two alternatives:

```java
boolean isContained1(int[] A, int[] B) {
    boolean AInB = true;
    for (int i = 0; i < A.length; i++) {
        boolean iInB = linearSearch(B, A[i]);
        AInB = AInB && iInB;
    }
    return AInB;
}

boolean isContained2(int[] A, int[] B) {
    int[] C = new int[B.length];
    for (int i = 0; i < B.length; i++) { C[i] = B[i] }
    sort(C); // heapsort
    boolean AInC = true;
    for (int i = 0; i < A.length; i++) {
        boolean iInC = binarySearch(C, A[i]);
        AInC = AInC && iInC;
    }
    return AInC;
}
```
(a) Calculate the worst-case running time of each of the two implementations using the asymptotic $\Theta$ notation.

(b) For each implementation, how much extra memory space is it required to store copies of the elements in A and B? You should take into account any copies made within the methods sort, linearSearch, and binarySearch.

(c) Find an implementation which is more efficient than both of the engineer’s implementation.
Why?
→ To get a technology job
http://www.careercup.com
→ To create the “New Google”
http://en.wikipedia.org/wiki/PageRank
→ To make science

→ To play the stock market

[Link](http://www.theguardian.com/business/2012/oct/21/superstar-traders-lost-magic)

One theory for the decline of the superstar trader is the rise of the analytical nerd and computerised algorithmic trading. Schmidt says: "The superstars are confronted with a changing market. The punting around is not working. You now need to be either a traditional long-term stock picker, a very short-term person working on algorithms, or a combination of both. There is no future for guys like Coffey."
To rule the world!

http://www.theguardian.com/science/2013/jul/01/how-algorithms-rule-world-nsa
→ For fun!