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 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?

---

\(^1\) Algorithms answering the above questions should work when the population of Ireland changes & for other 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 route from here to the airport?

\(^1\) Algorithms answering the above questions should work when the population of Ireland changes & for other 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 route from here to the airport?

\(^1\)Algorithms answering the above questions should work when the population of Ireland changes & for other 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 route from here to the airport?

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

→ Array, Linked List, Hash Table, Binary Tree, etc.

\(^1\)Algorithms answering the above questions should work when the population of Ireland changes & for other destinations!
Programs

→ must be **correct**
Programs

→ must be correct
  → Mathematical guarantee of correctness only though Formal Verification (see CS4004)
Programs

→ must be **correct**
  
  → Mathematical **guarantee** of correctness only though **Formal Verification** (see CS4004)

→ **Confidence** of correctness though **Testing**

  → in CS2010 (and in the SW industry): **Unit Testing**
Programs

→ must be **correct**
  → Mathematical **guarantee** of correctness only though **Formal Verification** (see CS4004)

→ **Confidence** of correctness though **Testing**
  → in CS2010 (and in the SW industry): **Unit Testing**

→ Other kinds of testing: integration testing, validation testing, user testing, etc.
ALGORITHMS + DATA STRUCTURES = PROGRAMS

Programs

→ must be correct
  → Mathematical guarantee of correctness only though Formal Verification (see CS4004)

→ Confidence of correctness though Testing
  → in CS2010 (and in the SW industry): Unit Testing

  → Other kinds of testing: integration testing, validation testing, user testing, etc.

→ must be efficient
WHAT ABOUT EFFICIENCY?

Is this a good measure of a program’s efficiency?

→ How long it takes the program to run on my laptop
WHAT ABOUT EFFICIENCY?

Is this a good measure 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
What about efficiency?

Is this a good measure 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
Is this a good measure 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
WHAT ABOUT EFFICIENCY?

Is this a good measure 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 \textbf{scales} to larger inputs

$\rightarrow$ When I \textbf{double} the input size my program takes \textbf{the same} time to run on the same computer (\textit{constant running time}).

$\rightarrow$ When I \textbf{double} the input size my program takes \textbf{twice} the time to run on the same computer (\textit{linear running time}).

$\rightarrow$ ...
**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*).
- ...

Scalability may sometimes seem crude but at least allows us to compare algorithms.
**WHAT ABOUT EFFICIENCY?**

**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*).
→ ...

Scalability may some times seems crude but at least allows us to compare algorithms.
→ Learn the most common A&DS that every CS graduate must know.

→ Example algorithms: Merge Sort, Union-Find, Dijkstra’s Shortest Path Tree, ...
The Software Engineer’s toolbox

→ 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: QuickSelect, MedianOfMedians, IntroSelect, using SoftHeaps
The Software Engineer’s toolbox

→ 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:\textsuperscript{a} QuickSelect, MedianOfMedians, IntroSelect, using SoftHeaps
→ Learn Abstract Data Types: interfaces of A&DS
The Software Engineer’s toolbox

→ 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: a QuickSelect, MedianOfMedians, IntroSelect, using SoftHeaps

→ Learn Abstract Data Types: interfaces of A&DS

→ Practice implementing A&DS

\[ a \text{https://www.quora.com/What-is-the-most-efficient-algorithm-to-find-the-largest-}
\text{element-in-an-array-of-n-elements} \]
The Computer Scientist’s toolbox

→ Learn to **evaluate** new algorithms
→ **Efficiency:** calculate the running time and memory usage
  → how well they scale
  → **Measuring aspects:** Worst-case, average-case, amortised, experimental performance
  → **Measuring systems:** big-O notation, tilde notation, cost models
→ **Correctness:** **rigorous testing** and some informal correctness arguments (see Unit Testing, test coverage)
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.
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. The first is:

```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;
}
```
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:

The second is:

```java
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.

(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?
http://www.youtube.com/embed/vSi6YoTPWLw?rel=0&start=8&end=165
WHY?

→ To get a technology job
http://www.careercup.com
→ To create the “New Google”
   http://en.wikipedia.org/wiki/PageRank
WHY?

→ To make science

→ To win big on the stock market

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."
WHY?

→ To rule the world!

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

How algorithms rule the world
The NSA revelations highlight the role sophisticated algorithms play in sifting through masses of data. But more surprising is their widespread use in our everyday lives. So should we be more wary of their power?
→ For fun!