CS660: Algorithms - Lecture 1

Instructor: Hoa Vu

San Diego State University

Course administrivia

- Instructor: Hoa Vu
  Email: hvu2@sdsu.edu
  Office hours: Tuesday-Thursday 2-3pm, GMCS 542
- Teaching Assistant: TBA
  Email: TBA
  Office hours: TBA

Course administrivia

- Textbook: Algorithms by Jeff Erickson. The book is available online for free at `http://jeffe.cs.illinois.edu/teaching/algorithms/`.
- You can also buy a hard copy on Amazon for about $30 (some additional chapters are not in the hard copy).
- Please do the reading assignment before class. The reading assignment for each lecture can be found on the course's blackboard.

Course administrivia

- Topics:
  - Divide and conquer
  - Dynamic programming
  - Greedy algorithms
  - Graph algorithms (shortest paths, bipartite matching, maximum flow, etc.)
  - NP-Completeness and approximation algorithms
  - Randomized algorithms.

Course administrivia

- An introduction course in data structures and algorithms is strongly advised. You should be familiar with basic data structures and algorithms.
- A good background in algebra and discrete math is also necessary. In particular, you should be familiar with how to write and read proofs.

Course administrivia

- There will be 2 midterm exams, 1 final exam, and homework. The course grade is broken down as follows:
  - Midterms (25 % each)
  - Final exam (35 %)
  - Homework (40%)
- See the university's website for the final exam schedule.

Course admnistrivia

- The lowest score homework will be dropped. No late homework will be accepted. If you need to do make-up exams, please notify me in advance.
- Do not search for solutions to the homework online.
- You may discuss the homework with other students. However, you must write up the solutions on your own.
- Cheating and plagiarism will be dealt with according to the university policy.

Course admnistrivia

- The lowest score homework will be dropped. No late homework will be accepted. If you need to do make-up exams, please notify me in advance.
- Do not search for solutions to the homework online.
- You may discuss the homework with other students. However, you must write up the solutions on your own.
- Cheating and plagiarism will be dealt with according to the university policy.

What is this course about?

- Design efficient algorithms
  - Efficiency is measured by running time, memory use, communication & number of rounds (in distributed algorithms).
  - In this course, we will mostly focus on the running time.
- What are the steps to design an algorithm?
  1. Specify the algorithm (the best way to describe an algorithm is via structured English & pseudo-code). Read page 13 in the textbook and the examples we cover in class. You should describe algorithms in this fashion.
  2. Show that the algorithm is correct.
  3. Analyze the running time.

A bit about myself

- I do research in theoretical computer science, mostly algorithms.
- If you are interested in doing research or an MS project with me, please talk to me (I do have funding).
- Some topics: graph algorithms, data science algorithms, or applying algorithms to suitable applications.

Abstraction: Makes it easier to study efficiency

- Making an algorithm fast involves many considerations that are architecture specific.
- For the sake of simplicity and generality, we will assume that: any location in memory can be accessed a unit cost and measure running time in basic steps such as pairwise arithmetic operation and memory accesses.

Example: Selection sort

- Suppose we want to sort a list of integers $A[1], A[2], \ldots, A[n]$.
- Selection sort algorithms:
  1. For $i = 1, \ldots, n$:
     1. Scan through $A[i], A[i+1], \ldots, A[n]$ to find the smallest number.
     2. Let $A[j]$ be the smallest among $A[i], A[i+1], \ldots, A[n]$.
     3. Swap $A[j]$ and $A[i]$.
- Proof of correctness: After the t-th iteration, $A[i]$ is the i-th smallest number for $1 \leq i \leq t$. Therefore, after the n-th iteration, the list is sorted.
- Running time: in the i-th iteration, we need to access $A[i], A[i+1], \ldots, A[n]$ to find the next smallest element. This takes $n - i + 1$ unit time cost. The swapping step can also be done in 3 unit time cost. Therefore, the running time is

$$\sum_{i=1}^{n}((n-i+1)+3) = 3n + \sum_{i=1}^{n} i = 3n + \frac{n(n+1)}{2} = O(n^2)\ .$$

Big-Oh

- To simplify the running time analysis, we do not count the steps exactly.
- The running time is measured "asymptotically".
- Definition: $f(n) = O(g(n))$ if there exists constants c and n_0 such that

$$f(n) \le c \cdot g(n) \text{ for all } n \ge n_0.$$

- Selection sort running time:

$$3 + \frac{n(n+1)}{2} = O(n^2)\ .$$

- Exercise: show the above by definition.

Big-Oh

- Definition: $f(n) = o(g(n))$ if there exists constants c and n_0 such that for all $k > 0$

$$f(n) \leq ck \cdot g(n) \text{ for all } n \geq n_0.$$

- Exercise: Show that $n^2 = o(n^{2.5})$ by definition.

Slightly easier characterization

- $f(n) = O(g(n))$ if
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} < \infty.$$
- $f(n) = o(g(n))$ if
$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

Big-Oh exercises

- Show that $100 = O(1)$.
- Show that $3n^3 + n = O(n^3)$.
- Show that $n \log_2 n = O(n^{0.01})$.
- Show that $\log_b n = O(\log_a n)$ for constants a, b.
- Show that $f(n) = O(g(n))$ and $\ell(n) = O(k(n))$ then $f(n) + \ell(n) = O(g(n) + k(n))$.

Slightly easier characterization

- $f(n) = O(g(n))$ if
$$\lim_{n \to \infty} \frac{f(n)}{g(n)} < \infty.$$
- $f(n) = o(g(n))$ if
$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0.$$

L'Hospital's rule

- If f' and g' exist, then

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}.$$

- Use L'Hospital rule to show that $10n^{10} - 1000n^9 = O(n^{10})$.
- Exercise (will be in homework 1) Show that $n^3 = O(e^{\sqrt{n}})$.

Other notations

- $f(n) = \Theta(g(n))$ if $f(n) = O(g(n))$ and $g(n) = O(f(n))$.
- $f(n) = \Omega(g(n))$ if $g(n) = O(f(n))$.
- $f(n) = \omega(g(n))$ if $g(n) = o(f(n))$.

Divide and Conquer (Recursion) - Exponential

- Assuming n is a power of 2. We want to compute $A(n) = 3^n$.
- Algorithm 1: Set x to 1. For $i = 1$ to n: $x \leftarrow 3x$.
- Algorithm 2:
  1. If $n = 1$, return 3.
  2. Else $x = A(n/2)$. Return $x \cdot x$.
- Assuming multiplication takes $O(1)$ time, the second algorithm runs in $O(\log n)$ time. Why?

Divide and Conquer (Recursion) - Tower of Hanoi

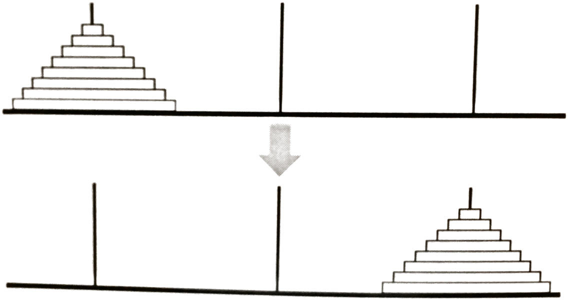

Figure 1.1. The (8-disk) Tower of Hanoi puzzle

- Move the discs from the original needle to the target needle.
- During the process, no bigger disc can be on top of a smaller disc.
- Design an algorithm that output a solution to the problem.

Divide and Conquer (Recursion) - Tower of Hanoi

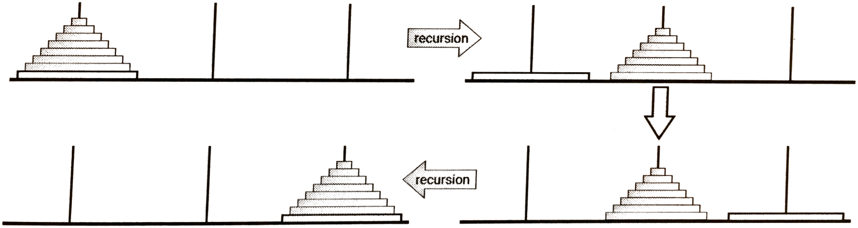

Figure 1.2. The Tower of Hanoi algorithm; ignore everything but the bottom disk.

- First, move the $n-1$ smallest discs to the non-target needle.
- Move the largest disc to the target needle.
- Move the $n-1$ smallest discs to the target needle.

Divide and Conquer (Recursion) - Tower of Hanoi

```
HANOI(n, src, dst, tmp):
  if n > 0
        HANOI(n − 1, src, tmp, dst)    ⟨⟨Recurse!⟩⟩
        move disk n from src to dst
        HANOI(n − 1, tmp, dst, src)    ⟨⟨Recurse!⟩⟩
```

Figure 1.4. A recursive algorithm to solve the Tower of Hanoi

Divide and Conquer (Recursion) - Tower of Hanoi

- The running time is given by the recurrence

$$T(n) = 2T(n-1) + 1$$

- Claim: $T(n) = 2^n - 1$. Prove by induction.
- $T(1) = 2^1 - 1 = 1$ which is true since we only need to move one disc.
- Suppose $T(i) = 2^i - 1$. Then
  $T(i+1) = 2T(i-1) + 1 = 2(2^i - 1) + 1 = 2^{i+1} - 1$.
- Therefore, the running time is $O(2^n + 1) = O(2^n)$.