CSCE 411: Design and Analysis of Algorithms

Lecture 1: Intro, Asymptotic Runtimes, Divide and Conquer

Date: January 14, 2025 *Nate Veldt, updated by Samson Zhou*

Course Logistics

- Read section 2.3, and chapter 4 for first week of classes.
- Read (or skim) chapters 1-3 to ensure familiarity with prerequisites
- Syllabus quiz is due Sat, Jan 18. HW 1 and intro video due Fri, Jan 24

1 Computational Problems and Algorithms

Definition 1. A ______________________________ is a general task defined by a specific type of __________ and an explanation for a desired ______________

A specific case of the problem is called an ______________

Example 1. Sorting

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

Output: A permutation σ of the input sequence so that

$$a_{\sigma(1)} \leq a_{\sigma(2)} \leq \cdots \leq a_{\sigma(n)}$$

An instance of this problem is the sequence

Example 2. Min element

Input: An array of n numbers: $[a_1, a_2, \dots, a_n]$

Output: The smallest element in the array and its index.

Definition 2. An **algorithm** is a computational procedure that

- ______________________________
- ______________________________

An **algorithm** is said to be **correct** if it ______________________.

2 Asymptotic Runtime Analysis (Chapter 3)

2.1 Rules for runtime analysis

- n denotes the size of the input
- Each basic operation takes constant time
- We focus on the ____________________ runtime
- We only care about the ____________ of the runtime

2.2 Some initial examples

Question 1. Given an array of n items, find whether the array contains a negative number using the following steps:

for $i = 1$ to n **do**
  if $a_i < 0$ **then**
    Return (true, i)
  end if
end for

What is the runtime of this method?

A $O(1)$

B $O(n)$

C $O(n^2)$

D It depends

E Other

F Don't know, I need a reminder for how this works.

Question 2. Given an $n \times n$ matrix A, what is the runtime of summing the upper triangular portion using the following algorithm? (same answers).

$sum = 0$
for $i = 1$ to n **do**
  for $j = i$ to n **do**
    $sum = sum + a_{ij}$
  end for
end for
Return sum

2.3 Formal Definitions

Let n be input size, and let f and g be functions over $\mathbb{N}$.

Definition 3. Big O notation.

A function $g(n) = O(f(n))$ (we say, "g is big-O of $f(n)$") means:

Definition 4. Big Ω notation.

$g(n) = \Omega(f(n))$ means:

Definition 5. Θ notation.

$g(n) = \Theta(f(n))$ means:

Equivalently, this means

Additional runtime examples

1. $4n^4 + n^3 \log n + 100n \log n$

2. $n + 2(\log n)^2$

3. $2^n + 10^{100}n^45$

Logarithms in Runtimes Which of the following runtimes are the same asymptotically? Which are not?

- $O(n \log n)$ and $O(n \lg n)$

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

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

- $O(n^{\log 100})$ and $O(n^{\lg 100})$

3 How to Present an Algorithm

Presenting and analyzing can be broken up into four steps.

1. **Explain**: the approach in basic English
2. **Pseudocode**: for formally presenting the algorithmic steps
3. Prove: the **correctness**
4. Analyze: the **runtime complexity**

As a rule it's a good idea to go through all steps when presenting an algorithm. Sometimes we will focus more on just a subset of these (e.g., you may be asked to prove a runtime complexity of an algorithm on a homework but not a correctness proof).

We will go through all four steps when we present the *merge sort* algorithm.

4 The Divide and Conquer Paradigm

The divide and conquer paradigm has three components:

- **Divide**:
- **Conquer**:
- **Combine**:

Example: Mergesort (Textbook, Chapter 2.3, 4) Given n numbers to sort, apply the following steps:

- Divide the sequence of length n into
- Recursively
- Combine

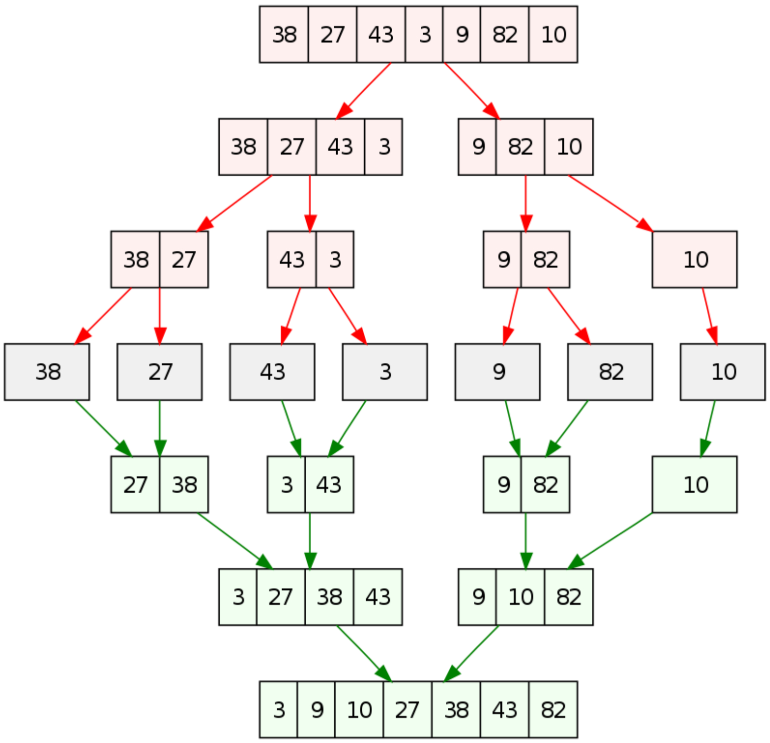

Image courtesy of Wikipedia: https://en.wikipedia.org/wiki/Merge_sort.

MERGESORT(A)
  $n =$ **length**(A)
  if $n == 1$ **then**

  else
    $m = \lfloor n/2 \rfloor$

  end if

4.1 Analyzing The Merge Procedure

Correctness: To merge two sorted subarrays into a master array

- Maintain a pointer to the ________________________________
- At each step, ____
- Since subarrays are sorted, one of these numbers is ________________________________

  ____
- so place ____
- At each step, we guarantee that: ______________________________________
- So continuing until both subarrays are empty, ______________________________