CS251 Midterm 1 - Fall 2022

schari

September 2022

1 Summations and Logarithm Rules

- Summations
  - Given c is a constant, $\sum_{i=m}^{n} c = c(n - m + 1)$
  - $\sum_{i=1}^{n} i = \frac{1}{2}n(n+1)$
  - $\sum_{i=1}^{n} i^2 = \frac{1}{6}n(n+1)(2n+1)$
  - Given a function $f(i)$, $\sum_{i=m}^{n} f(i) = \sum_{i=1}^{n} f(i) - \sum_{i=1}^{m-1} f(i)$
- Log Rules
  - In CS 251, if you are just given a $\log(n)$ without a base, they probably mean $\log_2(n)$
  - $\log(ab) = \log(a) + \log(b)$
  - $\log(\frac{a}{b}) = \log(a) - \log(b)$
  - Given 2 numbers a and b, $\log_a(n) = \frac{\log_b(n)}{\log_b(a)}$
  - $\log(n^a) = a\log(n)$
  - $a^{\log_a(n)} = n$
  - $a^{c\log_a(n)} =$

2 Experimental Analysis

- Limitations
  - Different machines can vary the run time
  - other processes/noise
  - May not be precise all the time

3 Recursive Functions

- Functions that call themselves in order to solve simpler problems
- Recursive functions don't call themselves infinitely; eventually stop when they reach a base case
-

4 Runtime Analysis

5 Arrays and LinkedLists

6 Stacks

- Data structure to store and remove data
- Last data pushed into the stack would be the first data popped off (LIFO)
  - Think of it like a stack of plates; the last plate placed on top is the first plate taken from the stack
- Standard methods for stacks:
  - `push()` - Add an element to the top of the stack
  - `pop()` - Remove the element from the top of the stack
  - `isEmpty()` - Whether or not there are elements on the stock
  - `size()` - Number of elements on the stack
  - `peek()` - View the element at the top of the stack without removing it
- Implementation using Arrays vs LinkedLists
  - Arrays: Lower memory overhead; unable to resize to accomodate more elements
  - LinkedLists: Pointers require more memory; can expand to increase number of elements in the stack

7 Queues

-

8 Trees