CS261 Notes 4/3/19

Intro to C

- header ("interface", .h) files can have
  - variable declarations: int x
  - type declarations: #define TYPE yeet
  - Preprocessor commands: #define XY 47
  - Func. Proto's: int yeet(int a, int b);
- A note about printf, scanf and comments
  - Printf is used printf("...%t...", var) where t is the correct character for the type to be printed, and var is that same type
  - comments ... might have to be /* ... */ but not 100% sure on that...
- Allocating Memory
  - use malloc(how_many_bytes)
    - ↳ malloc(15 * sizeof(struct name)); will allocate memory for an array of 15 name structs.
    - ↳ Malloc returns pointer.
- Freeing Memory
  - ↳ use free(ptr_to_free)
  - ↳ check with valgrind
  - ↳ ensure no null pointer before free

CS 261 Notes 4/5/19

Big O

- a way of measuring execution time, platform-independently.
- Algorithm A processes n elements in time t.
- we will consider the likely worst case

Big O for Simple Loops

- consider the number of times the loop will execute. check the comparation (x<n, for example) and the action (n++ for example).
- Ex: for (i=0, ix i<n, i++){...} /* O(n) */

Big O for Recursive Problems

- Multiply the number of times the function will be executed with the time per execution...

Common Big-O complexities

$n!$	factorial
2^n or C^n	Exponential
$n^d \mid d>3$	Polynomial
n^3	Cubic
n^2	Quadratic
$n\sqrt{n}$	"n-root-n"
$n \log n$	"n-log-n"
n	Linear
$\sqrt{n}$	Root n
$\log n$	Logarithmic
1	Constant

NEXT PAGE

Big O for calling and called functions

- Ex:

```
while (n-- < 0){
    if (condition){
        do_this_function();
    }
}
```

`do_this_function();` → $O(n)$

`if (condition){ do_this_function(); }` → $O(n) \cdot O(n) = O(n^2)$

`while (n-- < 0){ ... }` → $O(n^2) \cdot O(1) = O(n^2)$

Logarithmic Big O

- reasonably easily identified because:
  - the number of times you can split an array of length n is log(n)!
- Most "divide and conquer" algs are log
- Algs that use an ordering relationship within the data are likely log(n)...

Big O Domination

- If you have a program that executes many things with different big O complexities, then the largest big O complexity dominates
- Ex: $O(8n + 4n^2 + 1) \Rightarrow \boxed{O(n^2)}$
  - $8n$ → single loop
  - $4n^2$ → nested loop
  - 1 → print statement...