Monday (January 4th)

Algorithms:

URP program, let Yener know.

wear a mask around Yener

get on the piazza → homework posting

handwritten notes scanned & uploaded.

all announcements on piazza

TA's & I will use submitty for HW

recordings online, "as the need arises" → * get access to the recordings.

Yener will give permission to the recordings

Webex recordings → email him.

The reason he isn't giving permission to everyone for the recordings is because no one shows up to class when they have recordings.

This is supposed to be fun & interactive. Yener wants people to come & talk to him.

What we will learn: ~~how do you~~ given a problem how do you find a solution that is correct.

Monday January 9th 2023

Algorithms 12

goals: minimize space, time complexity
which is really minimizing energy.

1. how many steps it will take to run a given algorithm

eg: given input size N, when you are solving a problem
how many steps will it take are what
is the cost?
Cost → the operation
(addition vs multiplication vs exponentiation)

given
64 bit word
if a number doesn't fit in the word then it can't be stored.
So the size of the number is important.

be clear on:

number of steps on input N is a gross approximation
need to determine the amount of bits to operate on N.

N is a countable number of things.

Big-O notation gives an upper bound on time complexity.
"The Program is not going to exceed this bound

Ω → Lower bound. Omega
"No Matter which algorithm you use, it takes atleast
this amount of resources"

Algorithms Monday January 4th 2023

An Algorithm can give an upper bound

to get the lower bound a theorem must be developed

One goal is to optimize, make the upper bound match

the lower bound

A gap emerges between the upper and lower bound

there is a constant approximation for NP problems.

if you cannot close the gap → this problem does not exist a solution in polynomial time

it takes exponential or super exponential time at

which point you approximate based on the biggest factor.

"you stay up all night at the discotech then fail the final"

"in greedy algorithms there is no regret"

heisenberg uncertainty principles

if you observe a system you will change

the value

lab assignment every week - "simple". show your work, get a check

Will give homework with modifications from popular questions online

Will give exams. closed book.

yener doesn't trick students.

he will tell you what is in the exam during lecture.

yener does late hours for zoom meetings

Algorithms Monday January 9th

Today: using Fibonacci sequence
- time complexity
- Bit complexity
- Big-O notation

$Fib = [0, 1, 1, 2, 3, 5, 8, 13 \ldots]$

$F_0 = 0$, $F_1 = 1$ } Base case

n^{th} element of the sequence $F_n = F_{n-1} + F_{n-2}$ } recursive definition.

if you have a formula like this

you can turn this into an algorithm.

```
FIB 1(n)    n is an integer > 0
    if      n = 0    return 0
    if      n = 1    return 1
    return  FIB1(n-1) + FIB1(n-2)
```

① correctness → is this algo. given n going to return some string & is that some thing the answer we are looking for?

② efficiency → time / space

call stack: F(n)

depth $\lceil \frac{n}{2} \rceil$ ceiling

F(n-1) F(n-2)

depth: n-1

F(n-2) F(n-3)

F(2)

F(1) F(0)

The leaves of the execution graph will be recursive calls to the base case.

Algorithms. Monday January 9th 2023

continues...

# of recursive calls < # of nodes in a complete binary tree of depth $n-1$.

time complexity of this recursive algorithm can be bounded by calls, which is bounded by the number of nodes in a complete binary tree.

↓

$\# = 2^{n-1} = 2^n$

because it is a geometric series

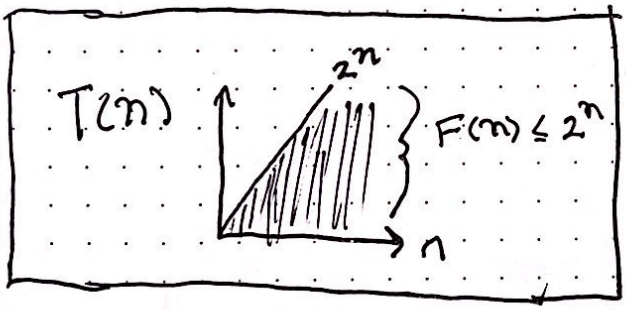
$$\sum_{i=0}^{k} r^i = \frac{r^{k+1}-1}{r-1}$$

in Fib case:
$r = 2$
$k = n-1$

$F(10) \leq 2^{10}$

Time complexity of FIB1(n)

in general → $F(n) \leq 2^n$: exponential in n.



Improvement! Stop recomputing and eliminate redundancy in the call stack.

Use memorization to cache the results of $F(n)$ storing it in a linear array, size of n.

FIB2(n)

$i = 0, 1, 2, 3 \dots n$

$F = 0\ 1\ 1\ 2\ 3\ 5 \dots$

F[0] = 0

F[1] = 1

for i = 2...n

F[i] = F[i-1] + F[i-2] → $n+1$ steps → linear time algorithm.

We assume that access is constant time so it's 1 is cost.

Monday, January 9th 2023

Algorithms

Continued...

$F(n-1) + F(n-2)$

$n-1$ bits, $n-2$ bits

n operations $\Rightarrow (n-2+1) * n = n^2 - n \approx \boxed{n^2}$

$n = 2^{(k)}$ bits to represent. | k is a linear function $\Rightarrow$ deal with larger numbers

Claim: $F_n \geq 2^{0.5n}$ for $n \geq 6$ $\quad F(n) \leq 2^n$

Proof: (by Induction)

F_n

Base case: $F_6 = 8 \geq 2^{(0.5) \times 6} = 2^{6/2} = 8$ ✓

Inductive Step: For $n > 6$

$F_{n+1} = F_n + F_{n-1} \geq 2^{n/2} + 2^{(n-1)/2} = 2^{(n-1)/2} \cdot (2^{1/2} + 1)$

$\geq 2^{(n-1)/2} \cdot 2 \geq 2^{(n+1)/2}$ ✓

$F(n) \approx 2^{0.5n} \approx 2^n$

Next time ~~type~~ Big O notation.