CS 180
1/5/16

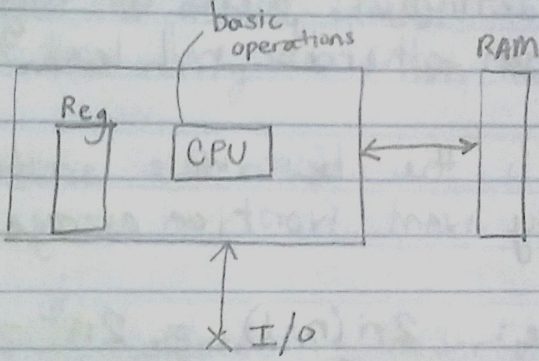
syllabus online
homework assigned every thursday
due every thursday before 8am

| | | |
|---|---|---|
| 7-8 | homework | 30% |
| | midterm | 30% |
| | final | 40% |

OH T 7-8am } BH 3532C
R 11-12pm

Model of Computation
a model that is universal (UM)
Serial model of computation

limited # of registers



the CPU computes a simple operation

all analysis is function of n

example: add n numbers
$x_1, x_2, \ldots x_n$

- first read all numbers into RAM: takes about 2n unit of time
  ↳ n for reading it, n for writing into RAM
- Into CPU: ~n
- adding: ~(n-1)
- output: ~1

runs in ~ 4n (time complexity)

want to change the order of the runtime

lower bound argument: must read all the numbers so the time complexity is at least ~ n most times we will be using VM not parallel

formal definitions are not approximate

example: Famous

- everyone knows them
- he/she doesn't know anyone

who is famous in the class?

model of computation: pick 2 people

ask if $A \xrightarrow{\text{knows}} B \rightsquigarrow 1$ unit

needs to ask $2(n-1)$ questions to see if one person is famous

by definition, there can only be one famous person otherwise will lead to contradiction

$2(n-1)$ is the best case

usually want worst or average case

worst case: $2n(n-1) = 2n^2 - 2n$

$\sim 2n^2$

$\sim n^2$

no proof

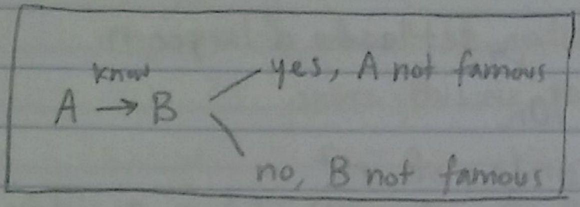
informally the lower bound <u>seems</u> to be n since we must ask everyone a question otherwise the output may change

can we improve the algorithm? yes

the proposed algorithm is an exhaustive search

technique: problem reduction
reduce the number of possibilitees

arbitrary ≠ random
random requires/implies effort



one person is eliminated as a famous candidate

after n-1 questions, we are left with one candidate but we still don't know if that candidate is actually famous
must verify → if not, there is no famous person

runtime $3(n-1)$
$\sim n$

optimal algorithm: lower and upper bound match

## Asymptotic Analysis

polynomial ↑
exponential ↓

| | 10 |
|---|---|
| $n$ | 10 |
| $2n$ | 20 |
| $n^2$ | 100 |
| $2^n$ | 1024 |
| $n!$ | 4mil |

formal: Asympotic Analysis

$f(n) = O(g(n))$

y $\exists$ $n_0, c$ s.t.

$f(n) \le c g(n)$ $\quad n \ge n_0$

only care about large n

example: $f(n) = O(n^2)$

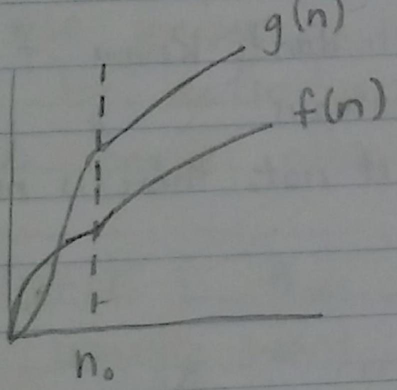
$f(n) = \frac{n^2}{4} < n^2$

set c to a larger n

$f(n) = 5n^2 < 10n^2$

log n is order of $n^2$

$2^n$ is not order of $n^2$ but $n^2$ is the order of $2^n$



this is why we only care about large n values hence $n \ge n_0$

c <u>must</u> be a constant; cannot be a function of n