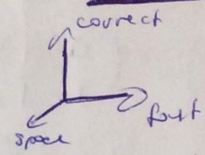
CS 124 — lecture 2 — (1)



(1) Multiply Alg.

74	19
148	9
~~296~~	~~4~~
~~592~~	~~2~~
1184	1
1406	

Algorithms

1. Does it stop?
2. Is it correct?
3. How fast is it?
4. How much space it takes?

3 D trade off space

(2) 74
100 1 1 (19)
multiply with the sum of powers of 2

Problem: Calculating Fibonacci #s

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 ....
at least doubling every 2 steps → exponential growth

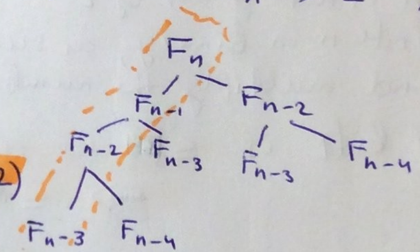
$F_n = F_{n-1} + F_{n-2}$

$F_0 = 0$, $F_1 = 1$

$F_n \geq 2^{n/2}$, $n \geq 6$

1 exponential $0 \leq n$

Fib (n)
  if n = 0 return 0
  else if n = 1 return 1
  else return F(n-1) + F(n-2)



time
Let T(n) = # additions
$T(n) = T(n-1) + T(n-2) + 1$
$T(0) = 0$, $T(1) = 0$
$T(2) = 1$

- potentially exponential

space: - depends on recursion implementation
- can put [ ] on the stack etc..

2 iterative $0 \leq n$

Fib2 (n)
  F[0] = 0
  F[1] = 1
  for i = 2 to n do
    F[i] = F[i-1] + F[i-2]
  return F[n]

time: n - 1 additions
↑
initially small then big #s.
≫ $O(n^2)$ considering growth in bits

space
- linear
- or constant, 3 cell array

3 matrix reduce fibonacci problem into matrix powering pr

$$\begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

$$\begin{pmatrix} F_2 \\ F_3 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^2 \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

$\vdots$

$$\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$

Consider raising a number to a power.

$3^n \rightarrow$ n multiplications, does not help, already have n-1 addition

case let n be a power of 2

$n = 2^k$

$\log n = k$ multiplications

$$\left.\begin{aligned} 3 &= 3 \\ 3^2 &= 9 \\ 3^4 &= 81 \\ 3^8 &= 7209 \end{aligned}\right\} \text{repeated squaring}$$

$\vdots$

case n is not power of 2

write n in binary, as sum of powers of 2

and multiply the numbers taken to these powers

# of bits is $\le \log n$ for a number n

$\Longrightarrow \le \log n$ multiplications in the final step

$\le \log n$ multiplications for each primary step

$\rightarrow \le 2 \log n$ in total

Consider matrices is a fixed # of mult and additions

each matrix is a

$$\begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} F_0 \\ F_1 \end{pmatrix}$$ 4 mult, 2 add

$$\rightarrow \le \underbrace{(4 \text{ mult} + 2 \text{ add})}_{C}\, 2 \log n$$

$\le$ log n

But representing nth fibonacci # takes

$F_n \ge 2^{n/2}$ for $n \ge 6$

$\text{bits}(F_n) \ge \frac{n}{2}$ addition of 2

then not faster than iter. $F\, n^2$?

$$c \cdot (\log n) \cdot (n + M(n))$$

time to multiply

n Formula:

still repeated squaring

$$F_n = \frac{1}{\sqrt{5}}\left[\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n\right]$$

real numbers
whereas matrix is integers
↳ latter may have a better implementation

Induction

if a statement $P(n)$ holds for $n=1$, and if for every $n \geq 1$ $P(n) \Rightarrow P(n+1)$ ← base case ← IH

then $P(n)$ holds $\forall\, n \geq 1$

ex 1

$$S(n) = \sum_{i=1}^{n} i$$

Prove $S(n) = \dfrac{n(n+1)}{2}$

Base case: true for $n=1$

$$1 = \frac{1 \cdot 2}{2}$$

IH: assume true for n $\quad S(n) = \dfrac{n(n+1)}{2}$

reduction to IH: show true for $n+1$

$$\underset{\text{by def.}}{S(n+1) = S(n) + (n+1)} \underset{\text{by IH}}{=} \frac{n(n+1)}{2} + (n+1) \text{ by IH}$$

$$= \frac{(n+1)(n+2)}{2} \quad \checkmark$$

Philosophy of O notation
- not care about constant factors
- care about asymptotics as the problem size gets big

Defn.

we say for [redacted] positive (integer) functions $f(n)$ and $g(n)$ that $f(n)$ is $O(g(n))$

if $\exists$ constants c, N

s.t. $\forall\, n \geq N$

$$f(n) \leq c\,g(n)$$

ex $2n^3 + 4n^2$ is $O(n^3)$. ✓

for $n \geq 1 \leftarrow N$

$$2n^3 + 4n^2 \leq \underset{\nwarrow c}{6}\, n^3$$

ex $2^{0.001n}$ is $O(n^{1000})$ NO

$$\lim_{n \to \infty} \frac{n^{1000}}{2^{0.001n}} = 0$$

ex $10 \log_2 n$ is $O(\ln n)$ ✓

$$\underbrace{\log_2 e}_{c} \cdot \log_e n = \log_2 n$$

ex

$n^{1/1000}$ is $O([\log_2 n]^{1000})$ NO

$$m = \log_2 n$$

$2^{0.0001m}$ is $O(m^{1000})$