Jan 4

Monday, January 4, 2016 12:54 PM

Github

m.s — / rbwalden/alg

Algorithms.

$F_0 = 0$

$F_1 = 1$

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

Store 2 last result.

A system to measure without depending of platform.

Algorithm

def A Finite sequence of steps which

solves a problem

Solves a problem.

The size of the algorithm = # of steps

↓

Speed

measure # clock cycle

function based on size of input / output.

Big O notation

$O(n)$

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

domain : int

range : pos int

if there exist const C and N

$f(n) \leq C\,g(n)$ for all $n \geq N$

↓

essentially

actually one huge C, N

Function matters.

$1 \to O(1)$ bounded function

$f(n) \leq C \quad \forall\, n \geq N$

$f(n) \leq C \quad \forall n \geq N$

$\forall$: For all

$\log_b n$

$\log_b n = \frac{\ln n}{\ln b}$

$u = \log_b n$

Jan 6

Wednesday, January 6, 2016 12:56 PM

Prop: in the following list, any func $f(n)$ to the left of any func $g(n)$ satisfies

$$f(n) = O(g(n)) \quad \text{but} \quad g(n) \neq O(f(n))$$

1, $\log_b n$, n^a $(a>0)$, c^n $(c>1)$, $n!$

bounded, logarithm, polynomial, exponential, factorial

if $\lim_{x\to\infty} \frac{f(x)}{g(x)} = L$

Case ① $0 < L < \infty$

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

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

notation

$f(n) = \Theta(g(n))$

or

$g(n) = \Omega(f(n))$

$f(n) = \Omega(g(n))$

Case ② $L = 0$

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

$g(n) \neq O(f(n))$

Case ③ $L = \infty$

then $f(n) \neq O(g(n))$

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

if $f(n) \neq O(g(n))$, for every choice of c, N, there is $n \geq N$

$$f(n) > Cg(n)$$

happen repeatedly.

Want to show

$$g(n) > c\, f(n) \quad \text{inf} - \text{often.}$$

$$\frac{1}{c} > \frac{f(n)}{g(n)}$$

$$\lim_{x\to\infty} \frac{\log_b x}{x^a} = \lim_{x\to\infty} \frac{\ln x / \ln b}{x^a} = \lim_{x\to\infty} \frac{\frac{1}{x}\frac{1}{\ln b}}{a x^{a-1}}$$

$$= \lim_{x\to\infty} \frac{1}{a \ln b\, x^a}$$

$$= 0$$

$c^x = e^{x \ln c}$

$$\lim_{x\to\infty} \frac{x^a}{c^x} = \lim_{x\to\infty} \frac{a x^{a-1}}{c^x \ln c} = \lim_{x\to\infty} \frac{a(a-1)x^{a-2}}{c^x (\ln c)^2} \cdots = 0$$

$n! = 1 \cdot 2 \cdot 3 \cdots n.$

$$e^c = \sum_{n=0}^{\infty} \frac{c^n}{n!} = 1 + c + \frac{c^2}{2!} + \cdots$$

$$\lim_{x\to\infty} \frac{c^n}{n!} = 0$$

$$n! \geq \left(\frac{n}{2}\right)^{\frac{n}{2}} = \left(\left(\frac{n}{2}\right)^{\frac{1}{2}}\right)^n > 2c^n = 2^n c^n$$

$$\left(\frac{n}{2}\right)^{\frac{1}{2}} > 2c$$

$n! \geq \left(\frac{n}{2}\right)^{\frac{n}{2}} = \left(\left(\frac{n}{2}\right)^{\frac{1}{2}}\right)^{n} > 2c^{n} = 2^{n} c^{n}$

$\left(\frac{n}{2}\right)^{\frac{1}{2}} > 2c$

Ex: $\ln(n!) = \Theta(n \ln n)$

$\ln(n!) \leq \ln(n^{n}) = n \ln n$

$\lim_{x \to \infty} \frac{\ln x!}{x \ln x} \to 1.$

$1 + 2 + \cdots + n = \frac{n(n+1)}{2} \Rightarrow O(n^2)$

In general. if $p(n)$ a $n^{d} + a_{d-1} n^{d-1} + \cdots$

↳ first degree (highest power)

$O(n^{d})$

Jan 8

Friday, January 8, 2016 12:56 PM

Sorting an array.

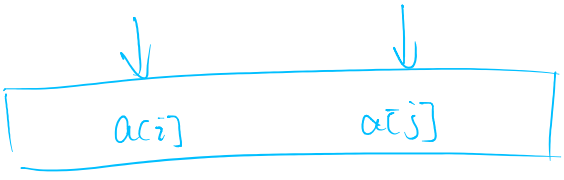
given an array with n entries as input

a[n] output a[n] in sorted orders

Worst Case (usually) steps.

Sometimes Avg Case → Expectation

Assume all of our sorting algorithms will be Comparison based

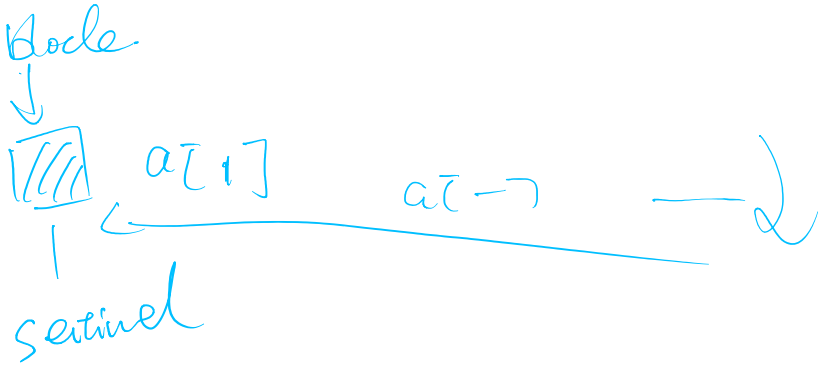


given $i < j$

check $a[i] \leq a[j]$ | Comparison

swaps a[i] a[j]

Insertion Sort



Worst Case: Reverse Order

a[1] — a[n]

no sorted	sorted
1	+1 Comp
2	+1
3	⋮
⋮	(n-1) Comp.
n-1	
$\frac{(n-1)n}{2}$	
$O(n^2)$	

Calculate Avg Case

Probability distributions

D → probability of every

Probability distributions

P_1 —— P_n probability of every possible event.

$P_i \geq 0$

$\sum_{i=0}^{n} P_i = 1$

X = random variable

$P_i = Prob(X = x_i)$

$E(X)$ = expected value of X

$= P_1 x_1 + P_2 x_2 + \cdots + P_n x_n$

$= \sum_{i=0}^{n} P_i x_i$

Ex: X = rolling a single die

$X = 1, 2, 3, 4, 5, 6$

$P = \frac{1}{6}$ ——

$E(X) = \frac{1}{6} + \frac{2}{6} + \frac{3}{6} + \frac{4}{6} + \frac{5}{6} + \frac{6}{6}$

$= \frac{21}{6}$ ← avg.

$= 3.5$

---

back to sorting

n! cases.

$n!$ Cases.

$N = n!$

$P_i = \frac{1}{n!}$

$E(X) = \frac{1}{n!}(\quad) + \frac{1}{n!}(\quad) + \frac{1}{n!}(\quad) + \frac{1}{n!}(\quad) + \frac{1}{n!}(\quad)$

too much

Count Swap

X = # of swap $\rightarrow$ = # of transposition

RANDOM

swap: AR
NR
DRN
OR
URON

transposition = pairs of entries $a[i], a[j]$ when $i < j$ but $a[i] > a[j]$

Jan 11

Monday, January 11, 2016 12:58 PM

Look for $n!$ possible orderings.

Prob of any order $= \frac{1}{n!}$

$E(\text{swaps}) = \sum_{\text{all } n!} \frac{1}{n!}$ (# of transpositions in some order)

$= \frac{1}{n!} \sum_{\text{all } n! \text{ orderings}}$ (# of transpositions for one ordering)

$+ E(\text{swaps}) = \sum_{\text{all } n!} \frac{1}{n!}$ (# of — reverse order)

$= \frac{1}{n!} \sum$ (# of ——)

$= 2E(x) = \frac{1}{n!} \sum \frac{n(n-1)}{2}$

$= \frac{n(n-1)}{2} = \frac{n(n-1)}{4}$

$= \Theta(n^2)$

Selection Sort.

① Find the smallest entry ($n-1$ comparisons)

Swap the smallest element and first one (if necessary)

② —— $n-2$

# of comparisons.

① Find the smallest thing (n-1 comparisons)

Swap the smallest element and first one (if necessary)

② —— n-2

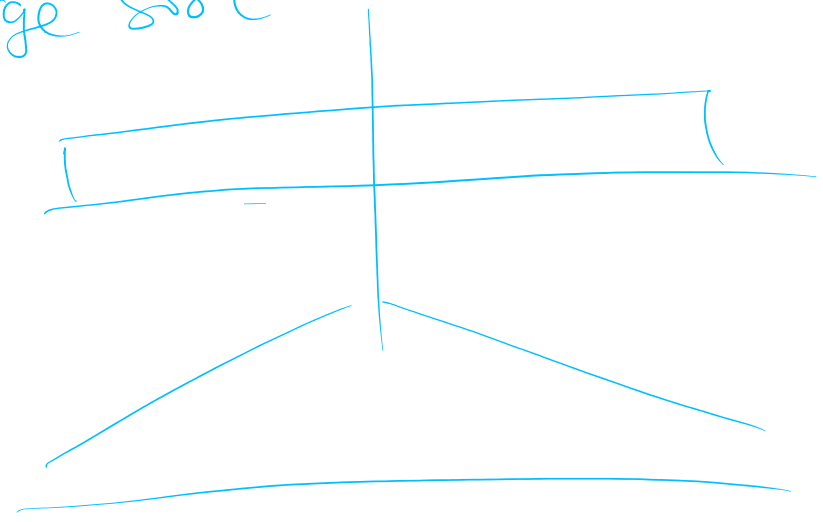
# of comparisons.

n-1
n-2
n-3
|
1

+

$\frac{(n-1)n}{2} = \Theta(n^2)$

merge sort



Space $\Theta(n^2)$

mergeSort (a, i, $\frac{i+j}{2}$) LHS

mergeSort (a, $\frac{i+j}{2}$, j) RHS.


website

Jan 13

Wednesday, January 13, 2016 12:55 PM

let $T(n)$ = # of comparison done by merge sort on an array of N entries.

$T(N)$ = # of comp on LH + # of comp on RH + # of comp to merge

$= T(\lceil \frac{N}{2} \rceil) + T(\lfloor \frac{N}{2} \rfloor) + N$

$T(1) = 0 \quad T(2) = T(1) + 2 = 2$

$T(N) = T(\frac{N}{2}) + T(\frac{N}{2}) + N = 2T(\frac{N}{2}) + N$

$\hookrightarrow T(N) = aT(\frac{N}{b}) + f(N)$

recursive part — pre-and recursive part.

$T(N) = aT(\frac{N}{b})$ | only sub problem

assume we know $T(1), T(2) \dots T(b)$

$T(\frac{N}{b}) = aT(\frac{N/b}{b}) \rightarrow aT(\frac{N}{b^2})$

$$= a^2 T\left(\frac{N}{b^2}\right)$$

$$= a^3 T\left(\frac{N}{b^3}\right)$$

$$\vdots$$

$$= a^k T\left(\frac{N}{b^k}\right)$$

Stop when

$$1 \le \frac{N}{b^k} < b$$

$$b^k \le N < b^{k+1}$$

$$k \le \log_b N < k+1$$

$k = \lfloor \log_b N \rfloor \rightarrow \Theta(\log_b N)$

$= O(\ln N)$

$a^k = ??$

$$a^{(\log_b N)} = N^{\log_b a}$$

back to

$$T(N) = a\left(T\left(\frac{N}{b}\right)\right) + f(N)$$

$N^{\log_b a}$

if $f(N) = \Theta(N^{\log_b a - \varepsilon})$

then for $T(N)$, $f(N)$ Doesn't matter.

$T(N) = \Theta(N^{\log_b a})$

then $T(N) = \Theta(N^{\log_b a})$

if $f(N) \neq \Omega(N^{\log_b a})$

then $T(N) = \Theta(N^{\log_b a} \log N)$

if $f(N) = \Theta(N^{\log_b a + \varepsilon})$

then $T(N) = \Theta(N \log^{a+\varepsilon}) = \Theta(f(N))$

For merge sort

$T(N) = \Theta(N \log N)$

$T(N) = aT(\frac{N}{b}) + f(N)$

$T(\frac{N}{b}) = aT(\frac{N}{b^2}) + f(\frac{N}{b})$

$T(N) = a^2 T(\frac{N}{b^2}) + f(N) + af(\frac{N}{b})$

$T(N) = a^3 T(\frac{N}{b^3}) + f(N) + af(\frac{N}{b}) + a^2 f(\frac{N}{b^2})$

$\vdots$

$T(N) = a^k T(\frac{N}{b^k}) + f(N) + af(\frac{N}{b}) + \cdots + a^{k-1} f(\frac{N}{b^{k-1}})$

Stop when $1 \leq \frac{N}{b^k} < k$

$b^k \leq N < b^{k+1}$

$\Theta(N^{\log_b a})$

$b^k \le N < b^{k+1}$

if $f(N) = O(N^{\log_b a - \varepsilon})$

then extra is

$\Theta(N^{\log_b a - a})$.

$1 + \left(\frac{N}{b}\right)^a + \left(\frac{N}{b^2}\right)^\varepsilon + \cdots$

$\Theta(N^{\log_b a - \varepsilon})[\, b^\varepsilon + b^{2\varepsilon} + b^{3\varepsilon} + \cdots \,]$

Jan 20

Wednesday, January 20, 2016 1:00 PM

$T(n) = \Theta(n^{\log_b a})$

Case 1 : if $f(n) = O(n^{\log_b a - \epsilon})$ $\epsilon > 0$

$$f\left(\frac{n}{b}\right) = O\left(\frac{n}{b}^{\log_b a - \epsilon}\right)$$

$$= O\left(\frac{n^{\log_b a - \epsilon}}{b^{\log_b a - \epsilon}}\right)$$

$$= O\left(\frac{n^{\log_b a - \epsilon}}{a \cdot b^{-\epsilon}}\right)$$

$$af\left(\frac{n}{b}\right) = O\left(n^{\frac{\log_b a - \epsilon}{b^{-\epsilon}}}\right)$$

$$= O\left(n^{\log_b a - \epsilon} \cdot b^{\epsilon}\right)$$

$$S = \frac{(b^{\epsilon})^k - 1}{b^{\epsilon} - 1} = \Theta(n^{\epsilon})$$

Case 2

$$f(n) \Rightarrow \Theta(n^{\log_b a})$$

$$\therefore T(n) = \Theta(n^{\log_b a} \cdot \log n)$$

Case 3 $f(n) = \Omega(n^{\log_b a + \epsilon})$

$$T(n) = \Omega(f(n))$$

in order to conclude that

$$T(n) = \Theta(f(n))$$

Use (regularity condition)

Use regularity condition

if there is some $c < 1$

such that

$$af\left(\frac{n}{b}\right) < cf(n)$$

$$a^2 f\left(\frac{n}{b^2}\right) < caf\left(\frac{n}{b}\right) < c^2 f(n)$$

$$\frac{1-c^k}{1-c} \longrightarrow f(n)$$

Ex: Use the master theorem to solve the following recurrence

a) $T(n) = 8T\left(\frac{n}{2}\right) + 5n^2 \log n$

b) $T(n) = 8T\left(\frac{n}{2}\right) + 5n^3$

c) $T(n) = 8T\left(\frac{n}{2}\right) + 5n^4$.

$\log_b a = \log_2 8 = 3.$

a) $f(n) = O(n^{3-\varepsilon})$

$n^2 \log n = O(n^{2.9})$

$n^2 \log n = O(n^{2.9})$

$\epsilon = .9$

$\therefore T(n) = \Theta(n^3)$

b) case 2

$T(n) = \Theta(n^3 \log n)$

c) case 3:

check regularity:

$a f(\frac{n}{b})$ — $a = 8$, $b = 2$

$f(n) = 5n^4$

$8 \cdot 5(\frac{n}{2})^4 = 40\frac{n^4}{16} = 2.5n^4$

$af(\frac{n}{b}) \leq cf(n)$

$c = 0.5$

Success

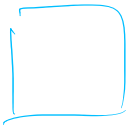
$T(n) = \Theta(n^4)$

Otherwise

$\Omega(n^4)$

Quick Sort

Smaller



"middle"

bigger.

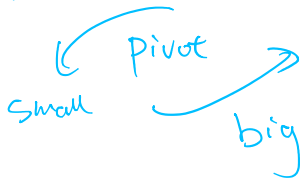
Worst $O(n^2)$

Best $O(n \log n)$

$1 + \sqrt{2} + \sqrt{3} + \cdots + \sqrt{n} \leq \sqrt{n} + \sqrt{n} + \sqrt{n} + \cdots + \sqrt{n} \leq n\sqrt{n}$

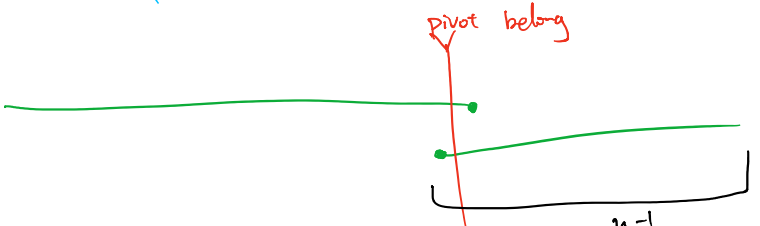
$n^{\frac{3}{2}}$

Quick sort



X = the number of comparison to QS of an array of n entries.

$E(X_n)$ = expected # of comparison to place pivot.
+ expected # of — to QS left.
+ — — — to QS right



Assume any every can be in the right most slot with $P = \frac{1}{n}$

$$= (n+1) + \frac{2}{n}\left[E(X_1) + E(X_2) - + E(X_{n-1})\right]$$ Sum $\sum_{m=1}^{n-1} \frac{1}{n} E(X_m)$

$\frac{1}{n}(E(X_1) + E(X_{n-2}))$

$\vdots$

$\frac{1}{n}(E(X_{n-1}) + 0)$

$=$

$$nE(X_n) = n(n+1) + 2(E(X_1) + - + E(X_{n-1}))$$

$$(n-1)E(X_{n-1}) = (n-1)(n) + 2(E(X_1) + — + E(X_{n-2}))$$

$$\downarrow$$

$$nE(X_n) - (n+1)E(X_{n-1}) = 2n + 2E(X_{n-1})$$

$$nE(X_n) - (n+1)E(X_{n-1}) = 2n$$

$$\downarrow$$

$$nE(X_n) = (n+1)E(X_{n-1}) + 2n$$

For comparison

$$nE(X_n) = (n+1)E(X_{n-1})$$

$$n E(X_n) = (n+1) E(X_{n-1})$$

$$E(X_n) = \frac{n+1}{n} E(X_{n-1}) = \frac{n+1}{n} \cdot \frac{n}{n-1} E(X_{n-2})$$

$$E(X_{n-1}) = \frac{n}{n-1} E(X_{n-2})$$

$$\vdots$$

$$E(X_n) = \frac{n+1}{n} \cdot \frac{n}{n-1} \cdot \frac{n-1}{n-2} \cdots \frac{2}{3} E(X_1)$$

$$= \frac{(n+1)}{2} E(X_1)$$

$$\frac{E(X_n)}{n+1} = \frac{E(X_{n-1})}{n} + \frac{2}{n+1}$$

$$\frac{E(X_{n-1})}{n} = \frac{2}{n} + \frac{E(X_{n-2})}{n-1}$$

back.

$$\frac{E(X_n)}{n+1} = \frac{2}{n+1} + \left[\frac{2}{n} + \frac{2}{n-1} + \cdots + \frac{2}{4}\right] + \frac{E(X_2)}{3}$$

go away

$$= 2\ln + \frac{E(X_2)}{3}$$

$$E(X_n) = 2n \ln n + O(n)$$

$$1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \ln n + \varepsilon_n \qquad 0 < \varepsilon_n < 1$$

$$\int_1^n \frac{1}{x}\, dx = \ln n - \ln 1$$

$$= \ln n$$

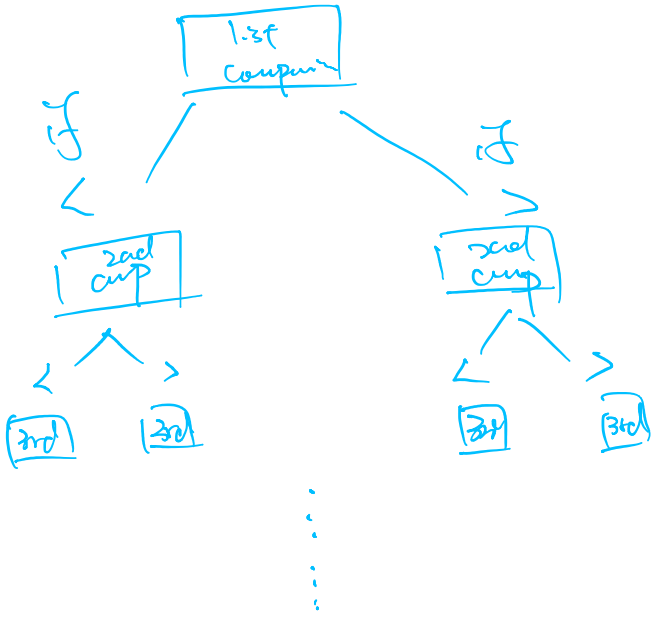
Jan 25

Monday, January 25, 2016 1:03 PM

Quick Sort — worst $O(n^2)$

avg $2n \ln n + O(n)$

Flow chart



1 comp → 2 course of action

2 comp → 2^2 ——

3 comp → 2^3 ——

k comp → 2^k comp——

in order to sort an array of n entries with k comparisons,

we must have

$$2^k \geq n!$$

$$k \geq \log_2 n! = \Theta(n \log_2 n)$$

more precisely

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

error $\leq \frac{\ln n}{2}$

$= \sum \ln x$ = sum of the areas of the —

$= \int_1^n \ln x\,dx$ + error

$= x \ln x \Big|_1^n - \int_1^n x \frac{1}{x}\,dx$ + error.

$= n \ln n - n + 1$ + error.

$= n \ln n - n + O(\ln n)$

$$\lim_{n \to \infty} \frac{\ln n!}{n \ln n} = 1$$

"perfect" algorithm takes at least $n \log_2 n + O(n)$ comparisons

$(\Omega(n \log_2 n))$

$$\frac{2n \ln n}{n \log_2 n} = 2 / \log_2 e = 2 \ln 2 \approx 1.3862$$

Heap Sort

$O(n \log n)$ worst.

A Heap is a data structure that we can organize to binary tree.

① every level of tree (except the last) have both left and right children.

② at most 1 child node —.

① every level of tree (except the last) have both left and right children.

② at most 1 child node — .

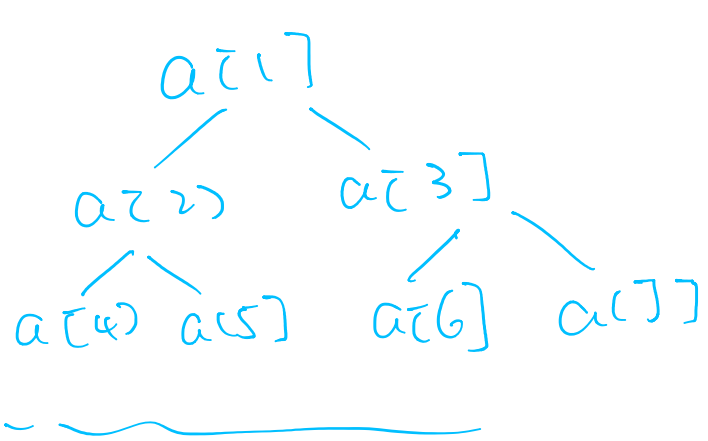
Data for the parent is at least as big as children.

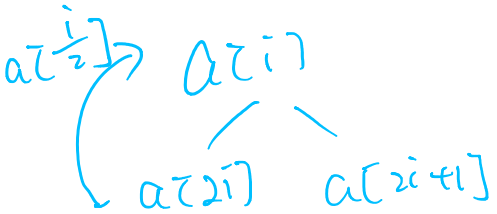
Jan 27

Wednesday, January 27, 2016 1:02 PM

Heap sort

build a heap on array





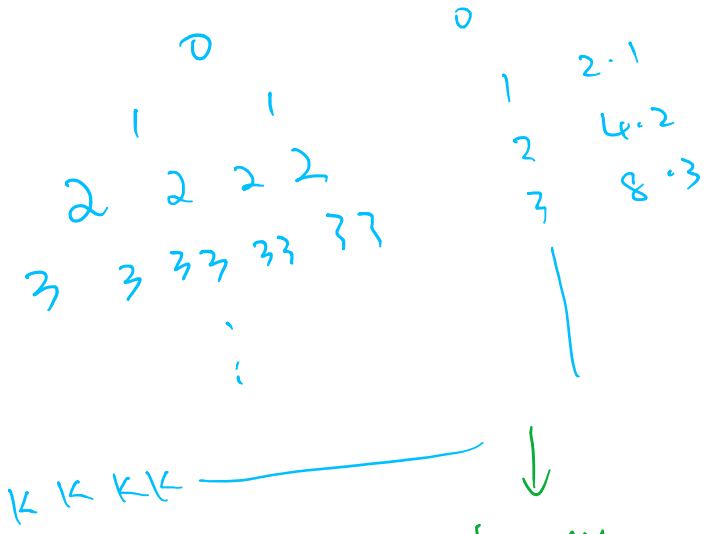
want to ensure the such that

$a[i] \geq a[2i]$

and

$a[i] \geq a[2i+1]$

Top down, bottom up

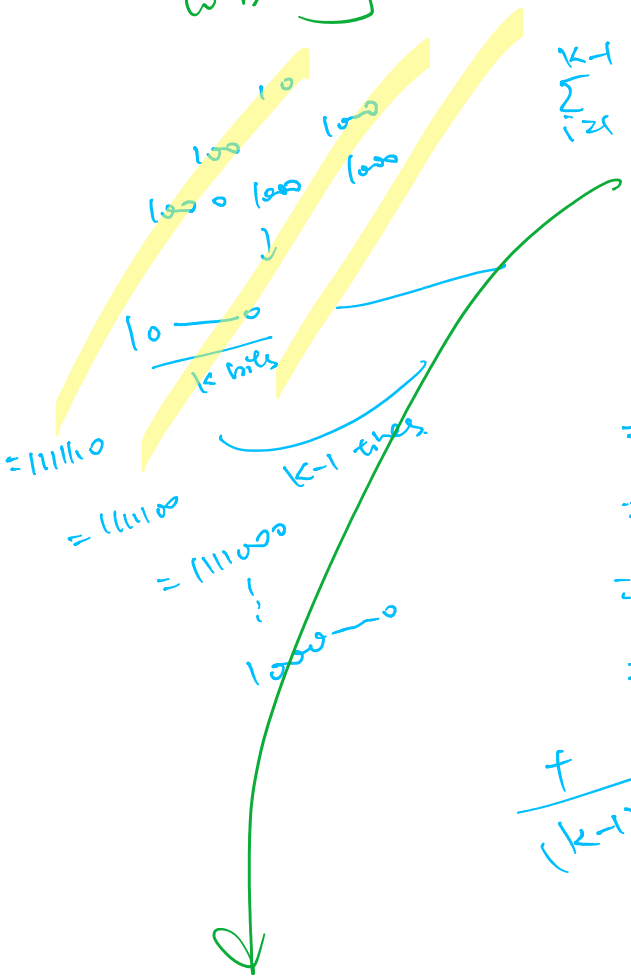


k depth of the tree.

$2^k \leq n < 2^{k+1}$

$k \leq \log_2 n < k+1$

$k = \lfloor \log_2 n \rfloor$



$\sum_{i=1}^{k-1} i \cdot 2^i$ + (# of leftover k's)

↓

$k(n-2^k+1)$

$= 2^k - 2$

$= 2^k - 4$

$= 2^k - 8$

$= 2^k - 2^{k-1}$

$+$

$(k-1)2^k - (2^k - 2)$

$= (k-1)2^k + (2^k - 2) + k(n-2^k+1)$

$= -2^k - 2^k + 2 + kn + k = kn - 2 \cdot 2^k + k + 2$ ($kn \geq n$, $\log n \geq 1$)

$\Theta(k \log n)$

Jan 29

Friday, January 29, 2016 12:58 PM

Heapsort

① Top down

→ $\Theta(n \log n)$ steps

② bottom up

# of swaps for every new elt to fall to bottom.

$\lfloor \frac{n}{2} \rfloor + \lfloor \frac{\lfloor n/2 \rfloor}{2} \rfloor + \cdots$

$n = b_k b_{k-1} \cdots b_2 b_1$

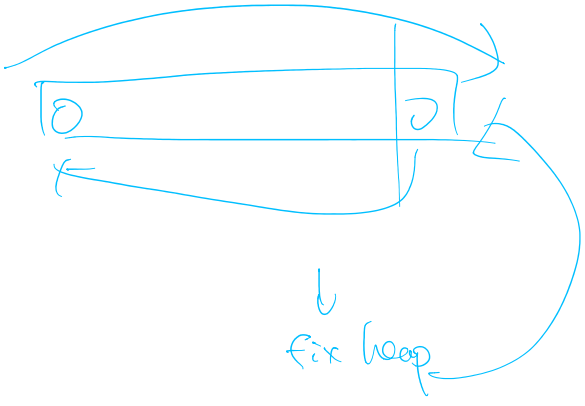
$\lfloor \frac{n}{2} \rfloor = \cdots b_2 \cdot 0.$

→ shifting

$b_k(2^{k-1}-1) + b_{k-1}(2^{k-2}-1)$

$+ \cdots + b_2(2-1) + b_1(1-1).$

$= n$



$\log n + \log(n-1) + \log(n-2)$

$+ \cdots + \log 2^2 + \log 2^1$

$\geq \log_2 n!$

→ $\Theta(n \log n)$

heap sort → priority queue.

heap sort → priority queue.

Shift to search an array.

Problem: given an array a[n] and a key — determine if the key is in the array

input array, value = key

output i, where a[i] = key if found.

① Sequential search

check a[i]

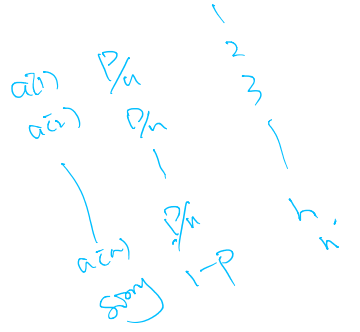
a[n]

worst O(n) comparisons

Avg $\Theta(n)$ ① the key is likely to appear in any of the place —.

② the probm of not found

let p be the probability to will be found.



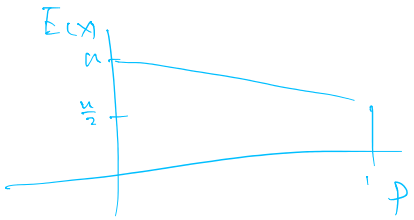
$$E(X) = \frac{p}{n}\cdot 1 + \frac{p}{n}\cdot 2 + \cdots + \frac{p}{n}\cdot n + (1-p)n$$

$$= \frac{p}{n}(1+2+\cdots+n) + (1-p)n$$

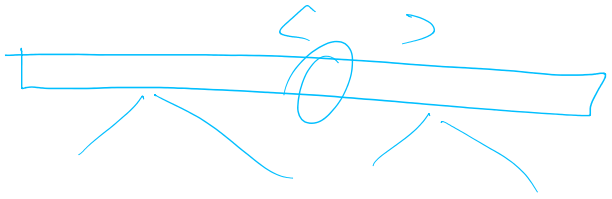
$$= \frac{p}{n}\cdot\frac{n(n+1)}{2} + (1-p)n$$

$$= \frac{p(n+1)}{2} + (1-p)n$$

$= \frac{p(n+1)}{2} + (1-p)\, n$



Binary Search



$T(n)$ = worst case

$= 1 + T(\lceil \frac{n-1}{2} \rceil)$

$= 1 + T(\lfloor \frac{n}{2} \rfloor)$

$= T(\lfloor \frac{n}{2} \rfloor) + 1 \rightarrow \log_2 1 = 0 \rightarrow T(n) = O(n^0 \log n)$

$= \Theta(\log n)$

Feb 1

Monday, February 1, 2016 12:55 PM

Binary Search

$$T(n) = T(\lceil \tfrac{n-1}{2} \rceil) + 1$$

$$= T(\lfloor \tfrac{n}{2} \rfloor) + 1$$

We want the exact solution to this recurrence

$$n = b_k b_{k-1} \cdots b_1$$

$$T(n) = 1 + T(b_k b_{k-1} \cdots b_2)$$

$$= 1 + 1 + T(b_k b_{k-1} \cdots b_3)$$

$$= \underbrace{1 + 1 + \cdots + 1}_{k} = 1 + \lfloor \log_2 n \rfloor$$

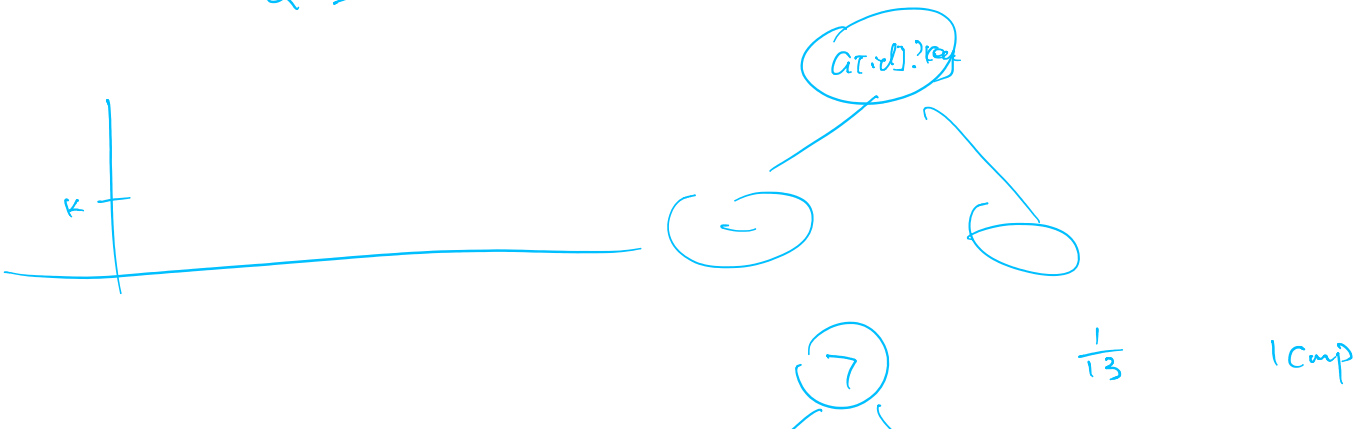
So $T(n) = k = 1 + \lfloor \log_2 n \rfloor$

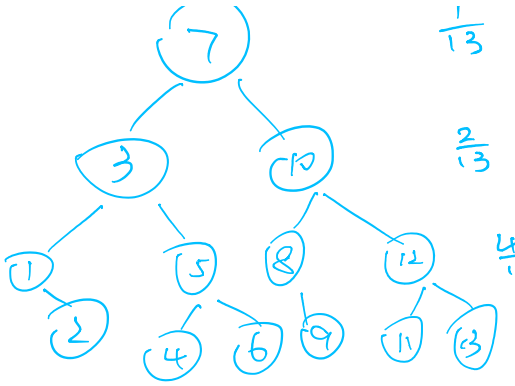
Any algorithm which searches a sorted array with an ~~array~~ takes at least $1 + \lfloor \log_2 n \rfloor$ comp.

if k is the numb of comp

then

$2^k \geq n+1 > n$





$\frac{1}{13}$ 1 cmp

$\frac{2}{13}$ 2 cmp

$\frac{4}{13}$ 3 cmp

$\frac{6}{13}$ 4 cmp

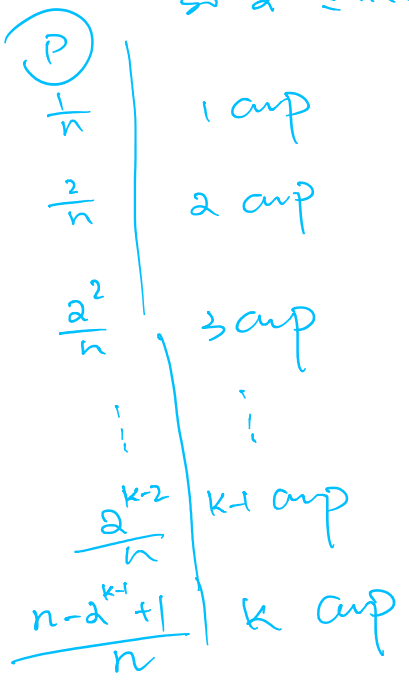
$E = \frac{41}{13} = 3\frac{2}{13} \to 4.$

General ←

For general n:

let k = # of bits.

so $2^{k-1} \le n < 2^k$.



$$E(x) = \frac{1}{n} \times 1 + \frac{2}{n} \times 2 + \cdots + \frac{2^{k-2}}{n}(k-1) + \left(\frac{n-2^{k-1}+1}{n}\right)k$$

$$= \frac{(k-2)2^{k-1}+1}{n} + \frac{(n-2^{k-1}+1)k}{n}$$

$$= \frac{nk - 2^k + 1 + k}{n}$$

$$= k - \left[\frac{2^k - k - 1}{n}\right]$$

Randomly generated trees.

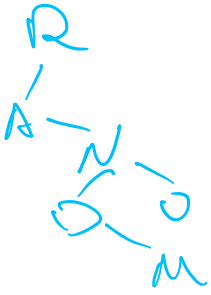
Feb 3

Wednesday, February 3, 2016 12:55 PM

Binary Search

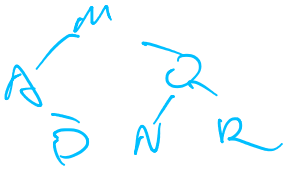
With randomly generated tree?

RANDOM

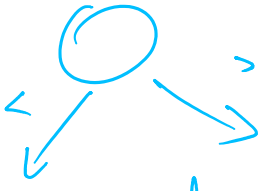


→

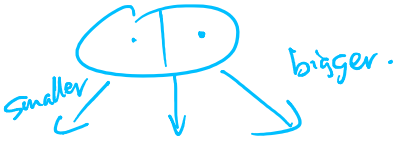
ADMNOR



Limit how we do fix



→



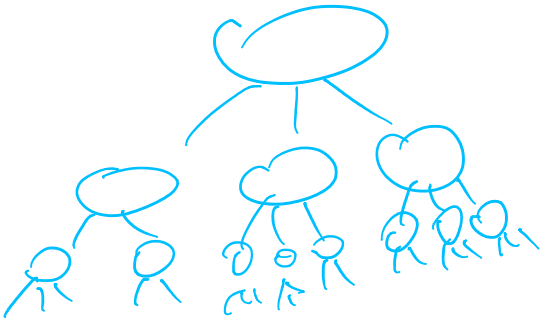
3-node

2 comp

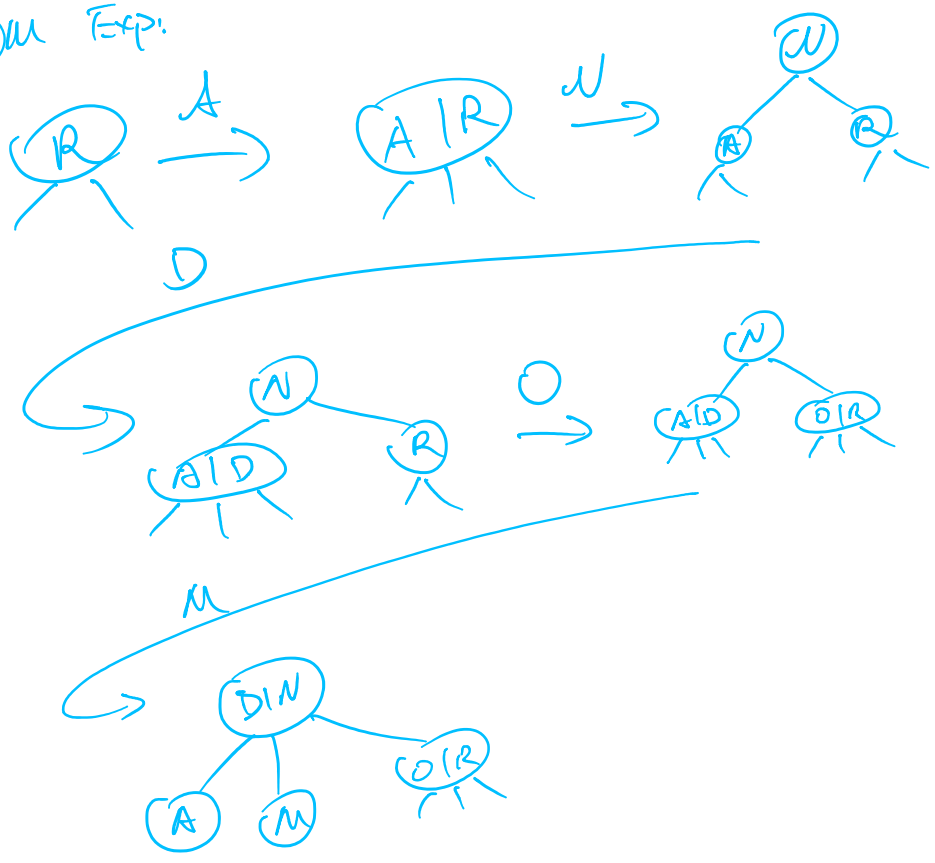
2-3 tree

is a tree made from 2-3 nodes

is a tree ... with the same depth on every path to the bottom.



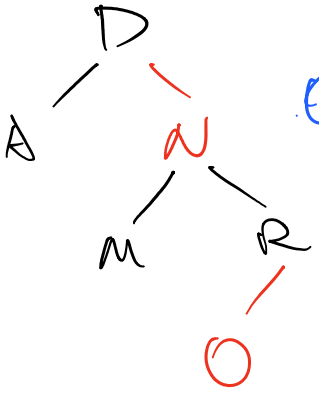
RANDOM Exp:

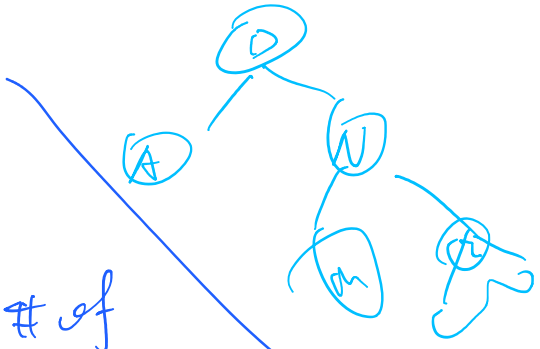


n

n log n

Convert to binary Tree.





① # of black links is the same on any path from T -> B

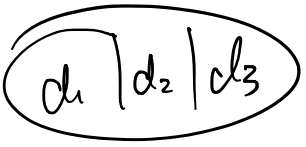
② No two reds in a row

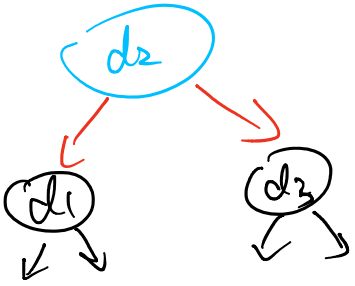
Feb 5

Friday, February 5, 2016 12:52 PM

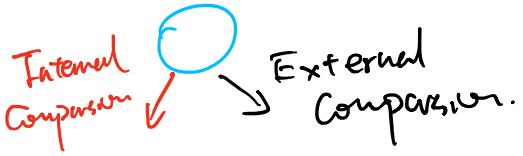
Red-black Tree

2-3-4 trees.



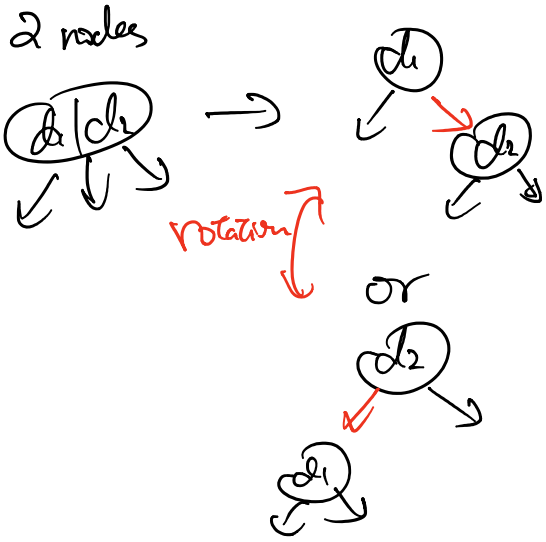


Rules:



---

put color on the node that is pointed by the arrow.



Top of the tree is always black.

Null pointer → black.

Rules:

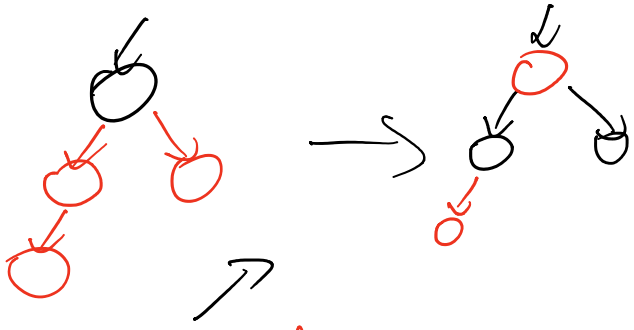
① no red child of a red parent.

② The black depth of the tree remains _ down any path from top to bottom.

③ any new node starts as a leaf of tree where ever fits.

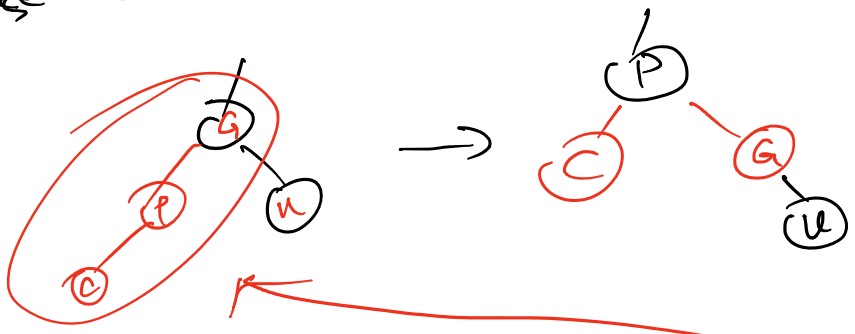
new node first be red

The only possible problem is the violation of red-red.

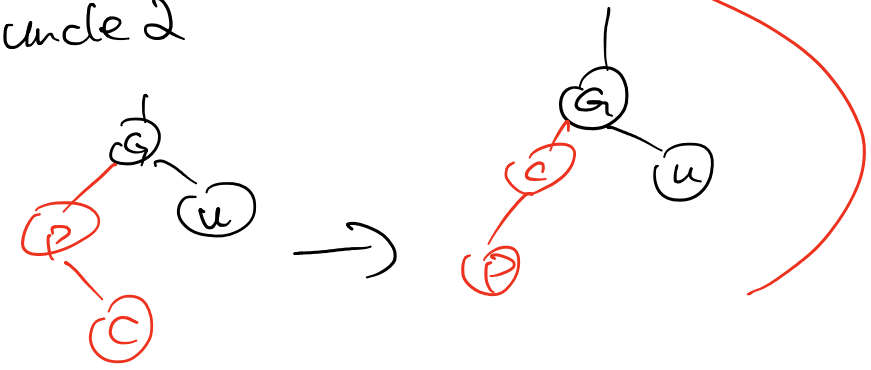


Case 1 : Red uncle

Case 2 : Black uncle



Black uncle 2



Feb 8

Monday, February 8, 2016 12:54 PM

Graph Algorithms.

Definition Graph $G(V, E)$

V = set of vertices = $\{v_1, v_2 \dots v_n\}$

E = set of Edges = $\{e_1, e_2 \dots e_k\}$.

When $E_1 = (V_1, V_2)$

if order matter:
directed graph.

else

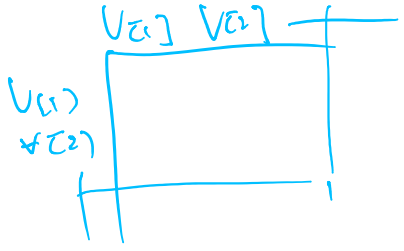
$\{v_1, v_2\}$ undirected graph.

Representation

① adjacency matrix

V[1], V[2] ——.

Represent using 2D array.



V[i] → V[j]

1 = edge exist

0 = ——

When two vertices are connected by edge then they are adjacent.

if value in matrix is not just 0 or 1, then we call graph "weighted"

loop = edge between vertex and itself.

$|V|$ = # of elements in V

$|E|$ = # of edges

# of bits is $|V|^2$ $(O(|V|^2))$

space $|V|^2$

---

② linked list representation.

Every vertex has a list of vertices that are adjacent to it.

Overall $\Theta(|V|+|E|)$

When $|E|$ is not (less than $|V|^2$, we call the Graph sparse.

Feb 10

Wednesday, February 10, 2016 12:54 PM

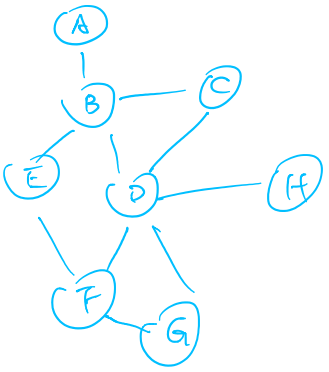
Graphs

Searching Graphs

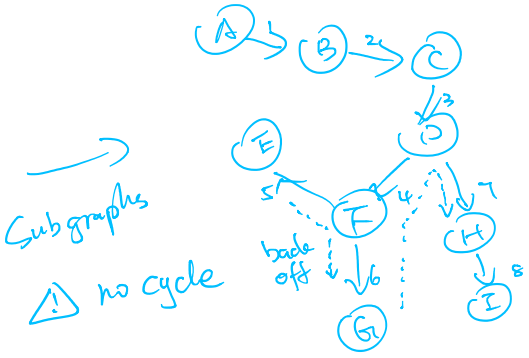
Exploring

DFS

BFS



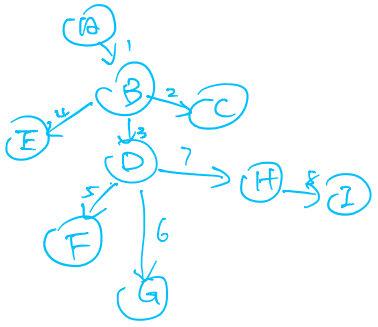
DFS



Subgraphs

⚠ no cycle

BFS



Cycle free graph that includes all the vertices of a connected graph is called tree.

DFS → recursion

DFS (G(V, E), V)

mark as visited

Find a new unvisited vertex

if found, DFS(G(V,E), V_i)

call(V_i)

if not, return to back off.

DFS + stack learn about connectivity

DFS + Stack learn about connectivity

BFS

Queue

A B

B C D E

C D E

D E F G H

E F G H

F G H I

G H I

H I.

I

Midterm

Big O ⟶ Θ, Ω, O

Sort

Search → seq, binary

how sorting works

run time

insertion

quick

heap

merge

selection

master Theorem

$T(n) = aT(n/b) + f(a)$

$\log_b a$ = power.

Case 1: at minimum

if $f(n) \in O(n^{\log_b a - c})$

then $T(n) = \Theta(n^{\log_b a})$

case 2: $f(n) = \Theta(n^{\log_b a})$

then $T(n) = \Theta(n^{\log_b a} \cdot \log n)$

case 3: $f(n) = \Omega(n^{\log_b a + \varepsilon})$

then $T(n) = \Theta(f(n))$

if regularity cond:

else $T(n) = \Omega(f(n))$

$a f(\frac{n}{b}) \leq k f(n)$

for some $k < 1$

for some $k \geq 1$

Ex: $T(n) = 1 \quad n \leq 2$

$T(n) = 8T(\frac{n}{2}) + 3n^{k}$

$\log_2 8 = 3$ So case 1

$T(n) = \Theta(n^3)$

$8 \to 4$ then case 2, $T(n) = \Theta(n^2 \log n)$

$8 \to 3$ then case 3

reg check:

$2f(\frac{n}{2}) = 3 \cdot 3(\frac{n}{2})^2$

$= (\frac{3}{2})n^2$

$= (\frac{1}{2})f(n)$.

< 1 so $T(n) = \Theta(n^2)$

Ex(3):

$f(n)$ = # of bits for sig

$g(n)$ = # of bits for zero.

bitsum = # of (1)

$g(n) = n - \text{bitsum}(n)$

$\frac{f(n)}{g(n)} = \Theta(n)$

$\log n! \to n \log n$.

$\frac{f(n)}{g(n)} \geq \frac{n \log n - \text{minor}}{\log n}$

Feb 17

Wednesday, February 17, 2016 12:57 PM

Graphs

An articulation node in a connected graph is one whose deletion along with associated edges leave a disconnected sub graph.

A graph that has no articulation nodes is called biconnected.

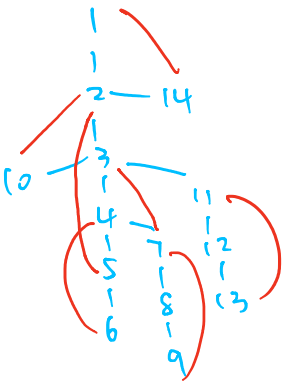
---

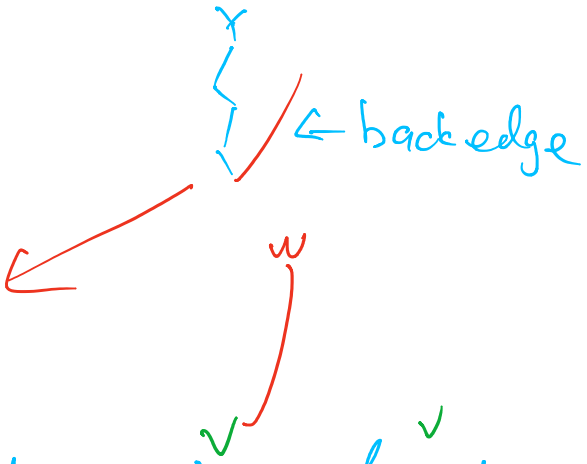
modify DFS to find articulation nodes and biconnected components. (use DFS on stack)

let val(v) = DFS. value of vertex v.
= order in DFS that v is visited.

let backval(v) = back value of vertex v.

DFS tree





backval(v) = minimum of val(w) for all vertices w that terminate a path of - followed by one back edge.

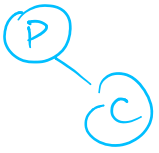
backval(1) = 1
backval(2) = 1
backval(3) = 2
backval(4) = 2

update parent's back value

Feb 19

Friday, February 19, 2016 12:54 PM

backval(v) = min(val(v), val(w), backval(c))



p != root of tree.

if backval(c) ≥ val(p)

else if backval(c) < val(p)

then there is a back trace

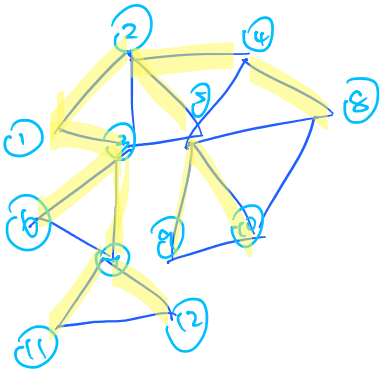
DFS start over again.

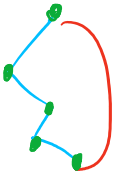
(2nd run)

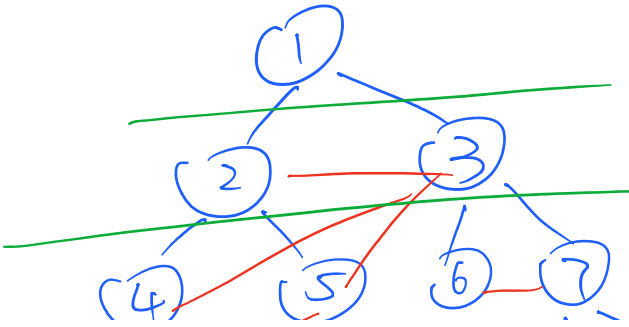
then multiple child

root is articulate node

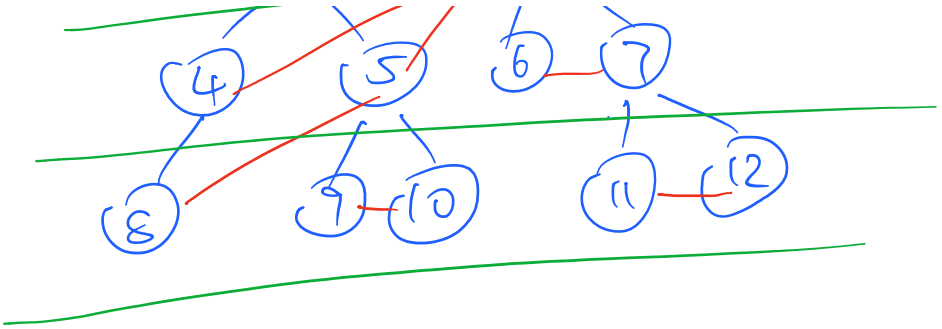
BFS







← layered tree



The distance between the root of BFS tree and any other vertex is the generation of the target vertex (consider top = 0)

Shortest path problem

Given a weighted graph, find the cheapest path from any vertex to any other vertex

what if weight is ≥ 0

Feb 22

Monday, February 22, 2016 12:56 PM

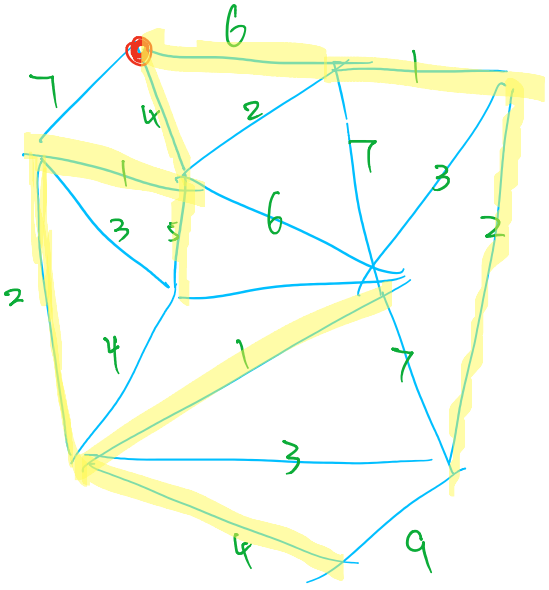
Shortest path.

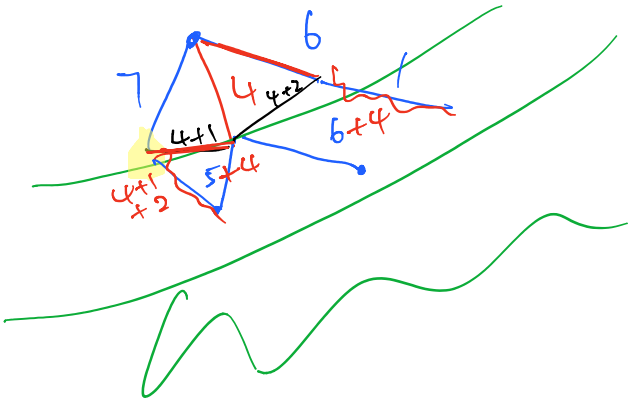
input: weighted graph (every edge has an associated weight)

Assume: $w(e) \geq 0$

output: The cheapest path from one vertex to the other.

frontier





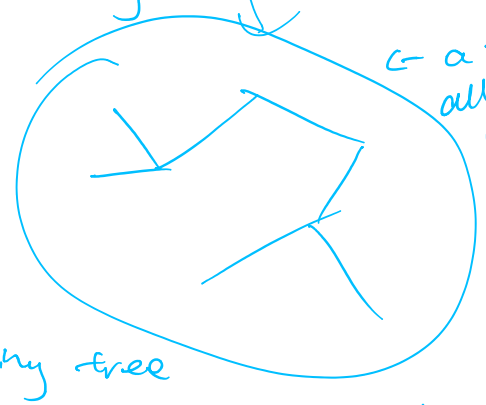
Visit all vertices

back trace (who brings me in?)

Dijkstra's Algorithm

Related

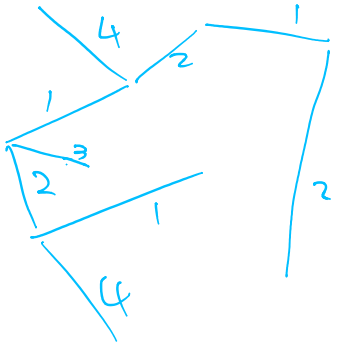
Minimum spanning Tree



← a tree containing all vertexes in the graph

Find a spanning tree for the graph with minimum total weight.

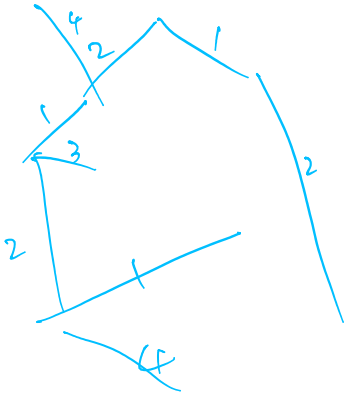
Consider edge only



Prim's Alg

Kruskal's Alg.

or



Feb 24

Wednesday, February 24, 2016 1:00 PM

$\ln(n!) = \Theta(n \log n)$

$\ln((n^2)!) = \Theta(n^2 \log(n^2)) = O(n^2 \log n)$

$\ln(n!)^2 = O(n \log n)^2 = O(n^2 \log^2 n)$

$\frac{n(n-1)}{6} = \frac{9 \cdot 8}{6} = 18.$

For (2) it's (14)

$m \le n < (m+1)^2$

$m \le \sqrt{n} < m+1$

$m + [n - m^2 + 1] \le m + 2m + 1 = 3m + 1$

$\Theta(\sqrt{n})$

Minimal spining Tree

Prim's Alg.

Kruskal's Alg always add cheapest, until it completes

$\Theta(|E| \log |E|)$

cheapest edge must be in the tree

with every new edge

1. if it connects two unseen vertices, start a new component.
2. if it connects a known vertex to an unseen vertex, add the unseen vertex to the component.
3. if it connects two known vertices, check and see if it's in same component.
   - if so skip
   - else, add the edge and fuse the components.

Feb 29

Monday, February 29, 2016 12:50 PM

Union-Find Data structure

```
Find (n)
    if parent (n) = n
        return n
    else
        return find (parent)
```

```
Union (m, n)
    if Find(m) = Find(n)
        Do nothing
    else
```

parent(find(u)) = find(u)

or

parent(find(u)) = find(u)

] choose by rank. (Depth)

only change if 2 ranks are the same

Set parent(u) = parent(parent)

return find(parent(u))

Mar 2

Wednesday, March 2, 2016 12:56 PM

Computational Algorithms

Multiply polynomials

Multiply matrices.

Eva polynomial

$P(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0$

$a_n \neq 0$

n = degree of $P(x)$

Input: x, $a[i]$ ← array of coeffs. $(0 \leq i \leq n)$ (deg of poly)

Output: $P(x)$

Q: How many operations are used to eva poly?

x^n first # of multiplications $= n + (n-1) + (n-2) + \cdots + 2 + 1 + 0$

$= \frac{n(n+1)}{2} = \Theta(n^2)$

# of additions $= n$

---

x first $\to x^n$.

# of multiplications $= (n-1) + n$ (a.)

# of additions $= n$

$\Theta(n^2) \to \Theta(n)$

Horner's method

addition to help

$a_n x^n + a_{n-1} x^{n-1}$

$x^{n-1}(nx + a_{n-1})$ →

$(\cdots((a_n x + a_{n-1})x + a_{n-2})x + a_{n-3})x \cdots)a_1)x + a_0$

$a_n x^n + a_{n-1} x$

$x^{n-1}(ax + a_{n-1})$ →

technique: straight line program.

$S_i = x \ (op) \ y$ (op) some operations like $+, -, *, \div$

x, y const, user input previous computations.

For (V2.)

$S_1 = x * x$

$S_2 = S_1 * x$

$S_3 = S_2 * x$

⋮

$S_{n-1} = S_{n-2} * x$

$S_n = a_1 * x$

$S_{n+1} = a_2 * S_1$

$S_{n+2} = a_3 * S_2$

⋮

$S_{2n-1} = a_n * S_{n-1}$

$S_{2n} = a_0 + S_n$

$S_{2n+1} = S_{2n} + S_{n+1}$

$S_{2n+2} = S_{2n+1} + S_{n+2}$

⋮

$S_{3n-1} = S_{3n-2} + S_{2n-1}$

For (Horner's)

$S_1 = a_n * x$

$S_2 = S_1 + a_{n-1}$

$S_3 = S_2 \times x$

$S_4 = S_3 + a_{n-2}$.

⋮

$S_{2i-1} = S_{2i-2} * x$

$S_{2i} = S_{2i-1} + a_{n-i}$

⋮

$S_{2n-1} = S_{2n-2} * x$

$S_{2n} = S_{2n-1} + a_0$

# of mult = n

# of add = n

Them: Any straight line program which computes $p(x)$ of degree n must have at least n multiplications and n additions

Ex: $x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + \cdots + x + 1$ 14 * 15 +

$S = x^{15} + x^{14} + \cdots + x + 1$

$xS = x^{16} + x^{15} + x^{14} + \cdots + x^2 + x$

$xS - S = x^{16} - 1$

$S = \frac{x^{16} - 1}{x - 1}$ →

$S_1 = x * x$

$S_2 = S_1 * S_1$

$S_3 = S_2 * S_2$ → 5 * 2 +

$S = \frac{x}{x-1}$ 1

$S_3 = S_2 * S_2$ → 2 +

$S_4 = S_3 * S_3$

$S_5 = S_4 - 1$

$S_6 = x - 1$

$S_7 = S_5 \div S_6$.

Lemma

Any straight line program which computes

$x_n + x_{n-1} + \cdots + x_0$

requires at least n additions / subtractions.

Proof by induction:

Base case: $n = 0$, x_0 At least 0 add.

Hypothesis step: Assume the statement is true for $n-1$,
want to show that's true for n.

Given an arbitrary S.L.P that computes $x_n + x_{n-1} + \cdots + x_1 + x_0$.

Given $x_n + x_{n-1} + x_{n-2} + \cdots$ as input.

$S_2 =$ output of $x_n + x_{n-1} + \cdots$

End first one that does $x_n \pm$ or $\pm x_n$.

replace that with 0

then it has at least $n-1$ additions.

So go back —

Mar 4

Friday, March 4, 2016 12:56 PM

Matrix multiplication

$A = [a_{ij}]$ is an $m \times n$ matrix (2-dim array)

(row, col)

$$= \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & & & & \\ a_{m1} & a_{m2} & a_{m3} & \cdots & a_{mn} \end{bmatrix}$$

$A + B$

$$[a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}]$$

Vector.

$$\begin{bmatrix} & A & \end{bmatrix} \begin{bmatrix} \\ \vdots \\ \end{bmatrix} = \begin{bmatrix} \\ \vdots \\ \end{bmatrix} \begin{matrix} \rightarrow a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \cdots + a_{1n}x_n \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \\ \vdots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n. \end{matrix}$$

$n \times 1$ matrix

$$\underset{m \times n}{\begin{bmatrix} A \\ \equiv \end{bmatrix}} \underset{n \times p}{\begin{bmatrix} B \\ \| \end{bmatrix}} = \underset{m \times p}{\begin{bmatrix} AB \\ c_{ij} \end{bmatrix}}$$

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + a_{i3}b_{3j} + \cdots a_{in}b_{nj}$$

each entry of AB uses n multiplications and $n-1$ additions.

All together, it requires mnp multiplications and $m(n-1)p$ additions

if $m=n=p$: n^3 multiplications, n^3-n^2 additions $\Big\}\ \Theta(n^3)$

How to reduce?

Use special case first.

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

$c_{11} = a_{11}b_{11} + a_{12}b_{21}$

$c_{12} = a_{11}b_{12} + a_{12}b_{22}$

$c_{21} = a_{21}b_{11} + a_{22}b_{21}$

$c_{22} = a_{21}b_{12} + a_{22}b_{22}$.

Strassen's Alg

↓

Can be used to calculate larger matrix.

4x4 mult → 7 2x2 multip.

18 2x2 addition

↓ ↓

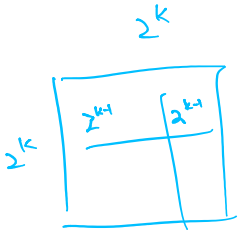
64 multip 49 multip

48 add. 18 + 7·18 addi = 198 add.

48 add. 18 + 7·18 addi
= 198 add.

How many *s and adds for strassen alg?

$2^k \times 2^k$ matrix.



$2^k \times 2^k \rightarrow 7 \cdot (2^{k-1} \times 2^{k-1})$ mult

↓

7^k mult. vs. 8^k

$n = 2^k \rightarrow n^3$ mult.

$k = \log_2 n$

$7^k = 7^{\log_2 n} = n^{\log_2 7} = n^{2.81}$

additions.

↓

$T(n) = 7\,T(\frac{n}{2}) + 18(\frac{n}{2})^2$

$= 7\,T(\frac{n}{2}) + \frac{9}{2}n^2$.

$\log_2 7 = 2.81 > 2$.

So. $\Theta(n^{2.81})$ vs $n^3 - n^2$

↓.

---

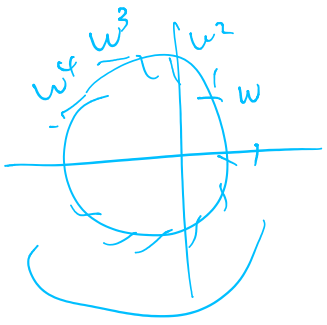
back to polynomials.

We want to choose values for the variable x which can lead to lots of regides when evaluating

which can lead to lots of recycles when evaluating

Tardiness.

Complex number



Geometric
Seq.

Mar 7

Monday, March 7, 2016 12:57 PM

Complex arithmetic:

Discrete Fourier Transformation Example

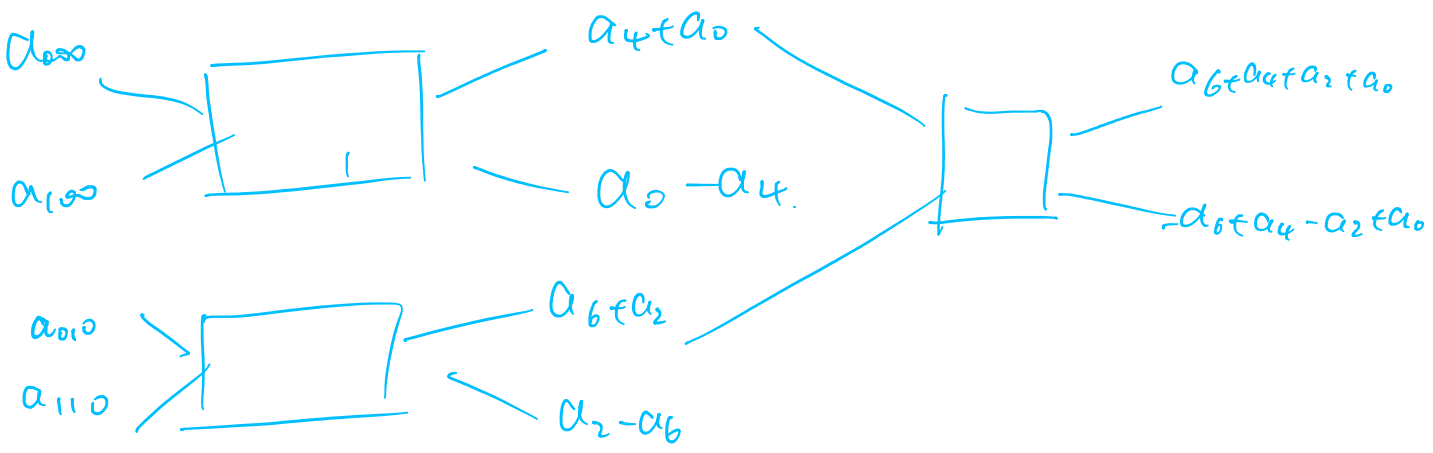
Mar 9

Wednesday, March 9, 2016 12:56 PM

Fast Fourier Transform.

$$P(z) = zP_o(z^2) + P_E(z^2).$$

butterfly switch.



polynomial multiplication.

$$a_7 z^7 + a_6 z^6 + a_5 z^5 + \cdots + a_1 z + a_0$$

$$b_7 z^7 + b_6 z^6 + b_5 z^5 + \cdots + b_1 z + b_0$$

$$\downarrow$$

$$a_7 b_7 z^{14} + (a_6 b_7 + a_7 b_6) z^{13} + (a_5 b_7 + a_6 b_6 + a_7 b_5) z^{12} +$$

Alt Alg

p → FFT → $\begin{bmatrix} p(1) \\ p(\omega) \\ \vdots \\ p(\omega^n) \end{bmatrix}$

q → FFT → $\begin{bmatrix} q(1) \\ q(\omega) \\ \vdots \\ q(\omega^n) \end{bmatrix}$

$\begin{bmatrix} p(1)q(1) \\ p(\omega)q(\omega) \\ \vdots \\ p(\omega^n)q(\omega^n) \end{bmatrix}$ → Inverse FFT → $\begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_n \end{bmatrix}$

$\Theta(n \log n) + \Theta(n) + \Theta(n \log n)$

$\Theta(n \log n)$

Mar 11

Friday, March 11, 2016 1:01 PM

Office hour Mon 10-3 OC 32

Eva polynomial

input x_1 —— x_n

(x)

$\Theta(n \log^2 n)$

FFT : $\Theta(n \log n)$ Comp

$$x^2 + 2x + 7 \;\overline{)\; x^3 - 5x^2 + 4x + 3}$$

$x - 7$

$-(x^3 + 2x^2 + 7x)$

$-7x^2 - 3x + 3$

$-(-7x^2 - 14x - 49)$

$11x + 52$ ← R.

$$\frac{x^3 - 5x^2 + 4x + 3}{x^2 + 2x + 7} = x - 7 + \frac{11x + 52}{x^2 + 2x + 7}$$

$$x^3 - 5x^2 + 4x + 3 = (x^2 + 2x + 7)(x - 7) + 11x + 52.$$