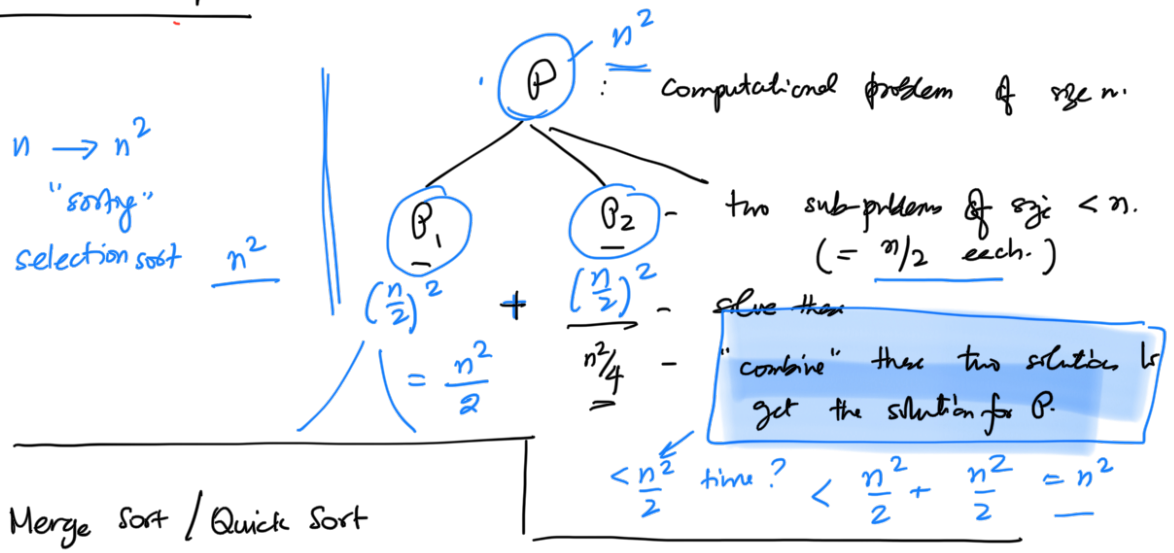
4/10/22

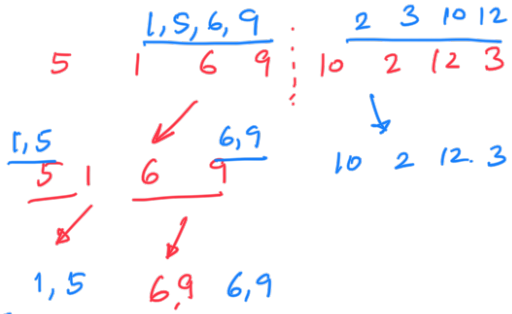
Divide and Conquer :



Sort n elements {

MergeSort	QuickSort	
(i) Easy	(i) Non-trivial	(i) Divide the input into 2 smaller inputs.
(iii) Non-trivial	(ii) Easy	(ii) Recursively solve the two smaller inputs.
		(iii) "Combine" the two solutions.

}.



MergeSort (A, n) { if n=1 ✓

n/4 ✓ A_1 ← First Half of A

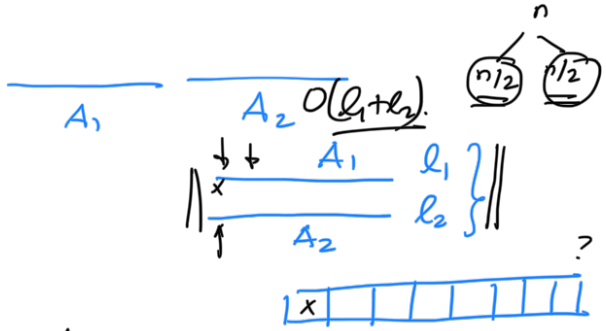
3n/4 ✓ A_2 ← Second Half of A

MergeSort (A_1, n/2)

MergeSort (A_2, n/2)

A ← Merge (A_1, A_2) ✓??

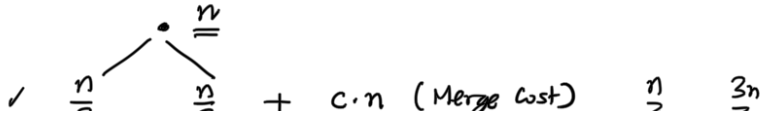
}.



Recurrence: T(n) : (worst-case) time to sort n elements ←

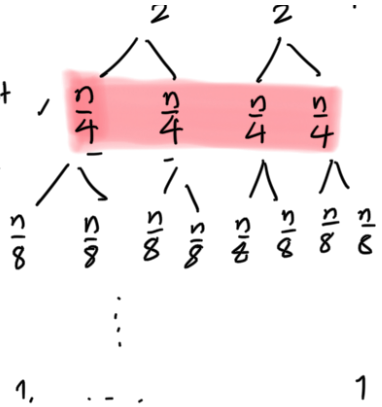
$$T(n) \leq T(n/2) + T(n/2) + c \cdot n$$ (c: constant.)

$$T(n) \leq 2T(n/2) + O(n).$$ ✓ Solution is $O(n \log n)$ ✓



Time

$T(n) \le cn \cdot \text{Height}$, $\le cn \cdot \log_2 n$.



$= \quad + c\cdot(\frac{n}{2}) + c(\frac{n}{2}) = cn.$

$c\left(\frac{n}{4}\right)\cdot 4 = cn.$

Examples:

$T(n/10) + T(9n/10)$

① $T(n) \le T(n/4) + T(3n/4) + cn.$

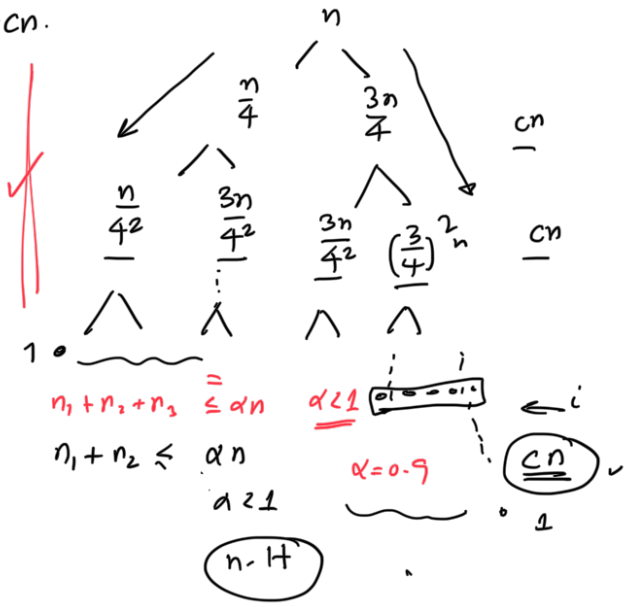
$= O(n \log_{4/3} n)$

$\left(\frac{3}{4}\right)^1 n = 1$

Height $\le \log_{4/3} n$

$\log_{4/3} n > \log_2 n$

$\log_2 n \cdot \log_{4/3} 2 > 1$



$T(n_1) + T(n_2) + T(n_3)$

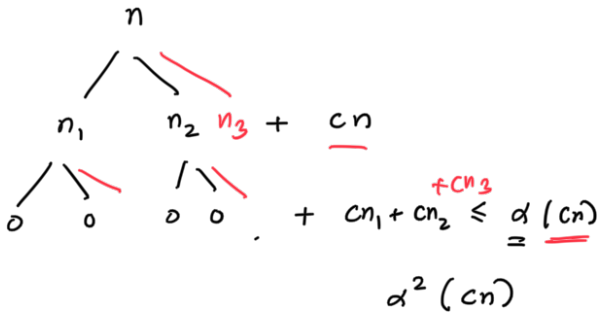
② $T(n) \le T(n_1) + T(n_2) + cn$

$T(n) = ?$ $O(n)$

$cn + \alpha(cn) + \alpha^2(cn) + \cdots$

$\frac{cn}{1-\alpha} = O(n).$

$T(n) \le T(n/2) + cn$



$T(n) = T(n/2) + O(1) \quad \leftarrow \quad T(n) = O(\log n)$

Quick Sort. (A, n) { ✓

$x \leftarrow A[1]$

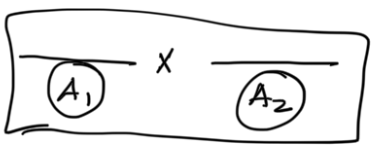
n_1 $A_1 \leftarrow$ all elements less than x

n_2 $A_2 \leftarrow$ all elements large the x

Output

A_1, x, A_2

}



Partition $c \cdot n$

$n_1 + n_2 = n-1.$

$T(n) \le T(n_1) + T(n_2) + cn$ ✓

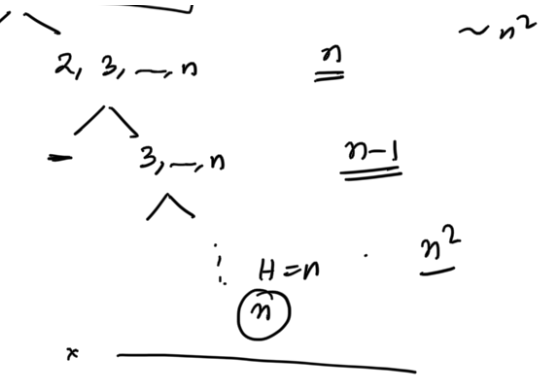
$\sim n/4 \quad 3n/4$

1, 2, 3, ..., n

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

Fix this? pick x "randomly"

Expected running time?



$T(n) = 2T(n/2) + O(n)$.

① Maximal points. $O(n \log n)$?

Input: set of n points on the plane

A point p is maximal (x_p, y_p) if there is no other point p' with $x_{p'} \geq x_p$ and $y_{p'} \geq y_q$

Report all the maximal points

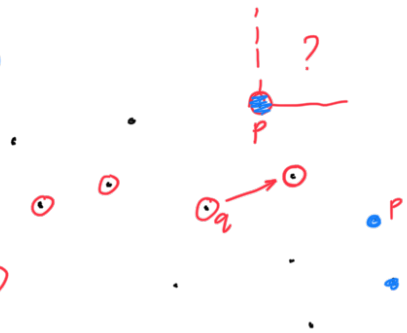
$O(n \log n)$ time?

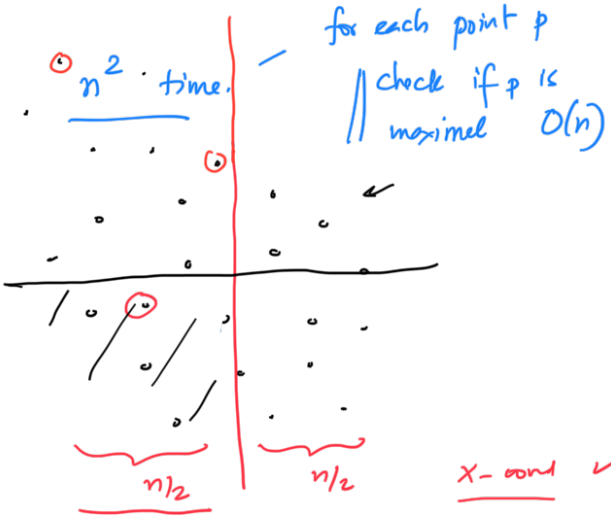
$T(n) = 2T(n/2) + O(n)$ ← $O(n)$

based on the "middle" x-coordinate

?? — P_1, P_2

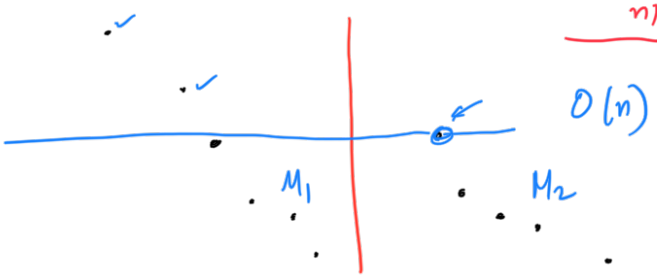
Solve the problem P_1, P_2 recursively





for each point p // check if p is maximal $O(n)$

n^2 · time.



$O(n)$

P_1 P_2 ✓

Combined answer?!

M_2: max. pts in P_2

M_1: max. " P_1

Add P_2 to M.

Find out the highest y-coord. in M_2 — q

M_1' ← set of points in M_1 with y-coord. higher than q

Add M_1' to M.

$T(n) \leq 2T(n/2) + O(n)$.

For
- sort the points on x-coord. $n \log n$,
- for each recursive call, we will maintain the sorted points.

points are in an array arranged by x-coord.

Maximal (P, n) {

$p \leftarrow P[\lceil n/2 \rceil]$ ←

$n \log n$ ✓

$P_1 \leftarrow$ left half of P

$P_2 \leftarrow$ right half of P

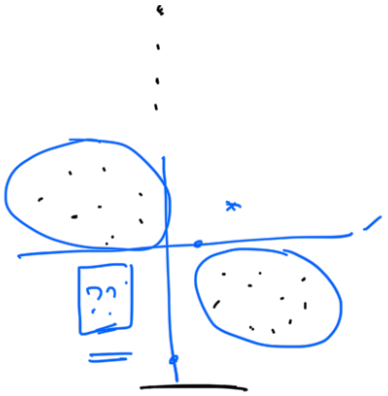
$M_1 \leftarrow$ Maximal $(P_1, n/2)$

$M_2 \leftarrow$ Maximal $(P_2, n/2)$

$M \leftarrow$ Combine (M_1, M_2)

return M

}.



(2) Closest pair of points : $O(n \log n)$ time? $T(n) = 2T(n/2) + O(n)$ ✓

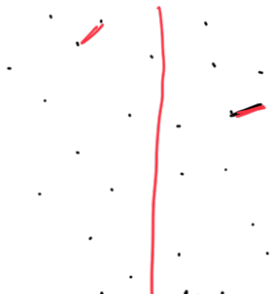
n points in the plane.

Find the closest pair of points.

n^2 time alg.

(x_1, y_1) (x_2, y_2) $O(n^2)$.

$(x_1 - x_2)^2 + (y_1 - y_2)^2$



arrayed according to x-coord.

Divide into two equal parts based on the x-coord.

Closest (P, n) {

$P[1] \ldots P[n/2]$ // $P[n/2+1] \ldots P[n]$

$P_1 \leftarrow$ left half

$P_2 \leftarrow$ right half

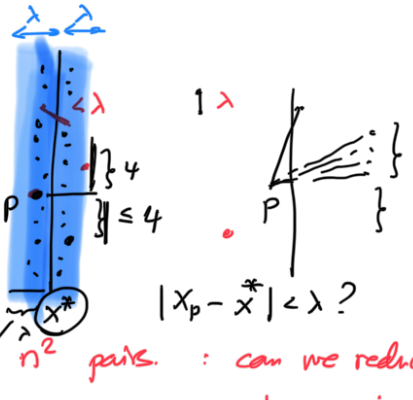
$p, p' \leftarrow$ Closest $(P_1, n/2)$

$q, q' \leftarrow$ Closest $(P_2, n/2)$

$\lambda_1 = \text{dist}(p, p')$, $\lambda_2 = \text{dist}(q, q')$

Output $\min(\lambda_1, \lambda_2)$??

$\lambda \leftarrow \min(\lambda_1, \lambda_2)$ ✓



$|x_p - x^*| < \lambda$?

n^2 pairs. : can we reduce this to n pairs only??

only have to look at points which lie in this slab

$\Rightarrow O(n)$ pairs ?

P_1', P_2': points of P_1, P_2 lying in the slab of width λ around x^*.

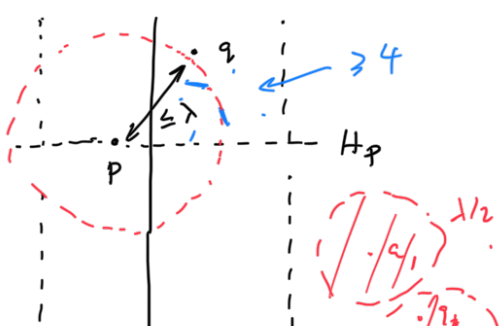
for each $p \in P_1'$ {

compare p with 4 points above it & 4 points below it in P_2'

}

Output the smallest dist. found.

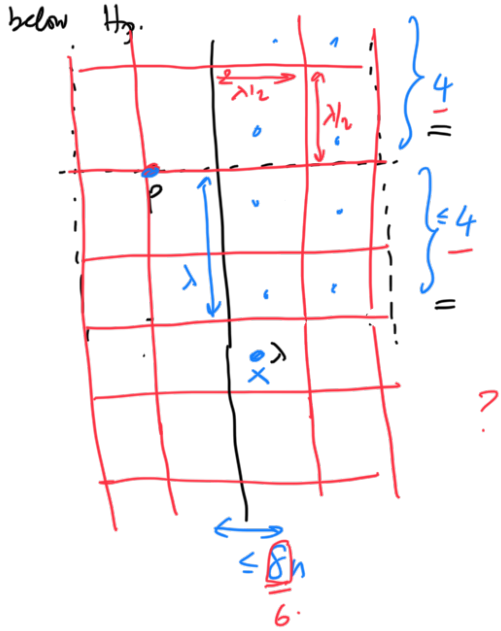
}

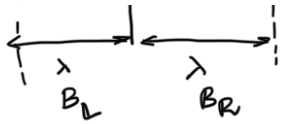


Fix a point $p \in B_L$.

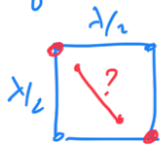
Consider all the points $q \in B_R$ dist $(p, q) \le \lambda$. Then there are at most 8 choices for q, in fact,

≤ 4 points above H_2 and ≤ 4 points below H_2.





Consider square of width $\lambda/2$



$\frac{\lambda}{2}\cdot\sqrt{2} < \lambda$

$O(n)$

? $|y_p - y_q| < \lambda$ at most 8 such pts.

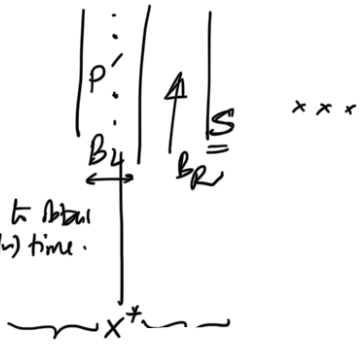
① P was sorted by the x-coord. We also need to maintain the point the points sorted by their y-coord.

p' Closest (P, P', n) { (P sorted by x, P' sorted by y-coord) $O(n)$ time

$P_1 \leftarrow P[1] \dots P[n/2]$

$P_2 \leftarrow P[n/2+1] \dots P[n]$

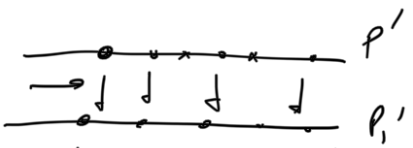
(P_1', P_2') ← use P' to obtain P_1' in $O(n)$ time.



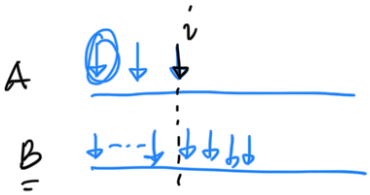
p': the points sorted by y-coord. (x,y)

x^*: median of x-coord

P_1': scan p' : (x,y) and $x \le x^*$ add it to P_1'



Similarly we can obtain the points B_L, B_R sorted according to their y-coord.



B[j], B[j-1], ... B[j-3]
B[j+1] ... B[j+4]
Merge.

i=0, j=0

repeat {

if A[i] > B[j], copy B[j] to C

j++,

A[i]

	1	2	3	4	5
A:	1	5	10	15	17
B:	2	3	7	9	12	13
	j	j	j	j	j	j

else copy A[i] to C.
increase i

}.

$O(n \lg n)$. ✗

(3) A : Find the median of A , $O(n)$ time.

A, k : Find the k^{th} smallest number (selection). $O(n)$ time.

$$T(n) = T(n_1) + T(n_2) + O(n) \quad \text{where} \quad n_1 + n_2 \le c \cdot n$$

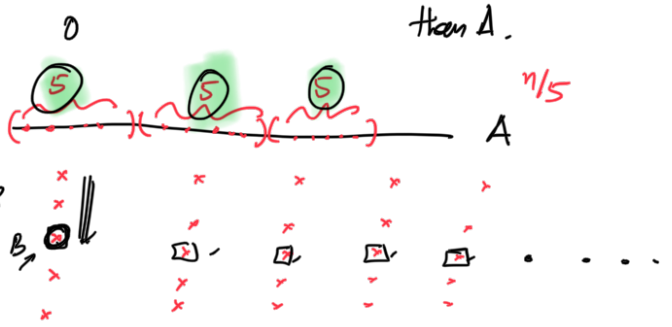
$T(n/5)$ where $c < 1$.

Select (A, k) {

Select (A', k') ←

A' will be a smaller array than A.

1. Break A into groups of 5 consecutive elements $O(n)$
2. Sort each such group. – $O(n)$ time
3. B ← new array formed by taking the middle element of B,

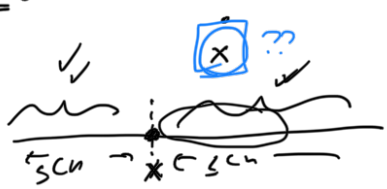


size of B? $n/5$.

(x) ← Median (B) Select (B, |B|/2) ✓

recursive call.

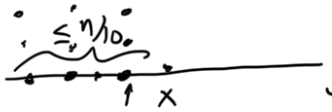
4. Partition the array A based on x

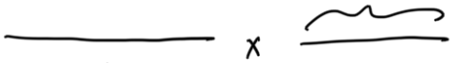


Claim: There are $\ge 3n/10$ ($n/3$) in A which are less than x.

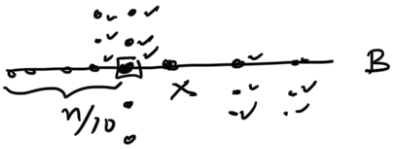
There are $\ge 3n/10$ in A which are more than x.

$T(n) = T(cn) + O(n)$ $O(n)$.



5. 

$O(n)$ time.



knowledge of x. (*)

if $k < |A_1| - 1$ then

Select (A_1, k) ✓

If $k = |A_1| - 1$, output x

if $k > |A_1| - 1$

Select (A_2, $k - (|A_1|+1)$) ✓

$$T(n) = T(n/5) + T(7n/10) + O(n)$$ ←

$\frac{7n}{10} + \frac{n}{5} = \frac{9n}{10}$

$T(n) = O(n)$.

$n/6$

1.

$\cdot n/6 \cdot x$

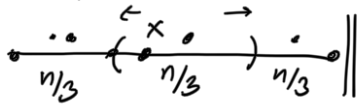
Could we have formed gps of 3 instead of gps of 5?

$T(n) = T(n/3) + T(2n/3) + O(n)$ $\Big\|$ $O(n \lg n)$. $\frac{n}{3} + \frac{2n}{3} = n$.

"Practical version"

A_1 * A_2

1. Pick a number x uniformly at random in A. $\Big\|$ T_n: running time of this alg. Expectation of T_n.
2. Follow ⊛ (Steps 4 and 5 wrt x).



3 times.

1/3. 2/3

Q: Suppose we have an experiment where the prob. of success is p. [How many times do we have to repeat it till we get a success? X

X: random variable.

$\Pr[X=1]$, $\Pr[X=i]$ $\qquad \Pr[X=i] = (1-p)^{i-1} \cdot p$

$\mathbb{E}[X] = \sum_{i=0}^{\infty} i \cdot \Pr[X=i]$

(Expectation of X)

Expected Value of X. $= \sum_{i=0}^{\infty} i \cdot (1-p)^{i-1} \cdot p$

$= 1/p$

$\Pr[H] = 1/2$ 1

$\Pr[T] = 1/2$ 0

$1/2$.

"sorted"

$\mathbb{E}[T_n]$?

A

$\frac{1}{n}$ $\Big\|$ If we happen to pick the i^{th} smallest no. as x then the array recursive call would have size either $i-1$ or $n-i$

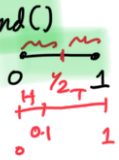
11/10/21. Selection Problem: A, k k^{th} smallest

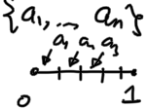
(i) Deterministic: fairly complex alg.

(ii) Randomized alg: uses randomness eg, $\Big\|$ toss a coin

pick an element from $\{a_1, \ldots, a_n\}$ u.a.r.

rand()





Random Variable X: (set of outcomes) $\to \mathbb{R}$.

$\omega_1, \omega_2, \ldots, \omega_\ell \to X(\omega_1), X(\omega_2), \ldots$

$p_1, p_2 \ldots p_\ell$

$X(\omega_2)$

with each outcome there is

$\sum_a$ "a"

a prob.

$\mathbb{E}X = \sum_{\text{outcomes } \omega} p_\omega \cdot X(\omega)$.

expectation.

Input I : execution of the algorithm is not fixed.

$\omega_1, \ldots, \omega_\ell$

executions of the alg. depending on the outcome rand().

X : running time

$X(\omega_i)$: # steps in ω_i

$\mathbb{E}[X]$ is $\mathbb{E}$(running time) =

eg: (i) Toss n unbiased coins. X : # heads. $\mathbb{E}X$.

$$\rightarrow \mathbb{E}X = \sum_a a \cdot \Pr[X=a] = \sum_{a=0}^{n} a \cdot \binom{n}{a} \frac{1}{2^n} = n/2$$

$$\mathbb{E}[X] = \sum_{\text{outcomes } \omega} p_\omega \, X(\omega) \quad (X(\omega) = a)$$

$$= \sum_a \sum_{\omega : X(\omega)=a} p_\omega \cdot a = \sum_a a \cdot \sum_{\omega : X(\omega)=a} p_\omega$$

$$= \sum_a a \cdot \Pr[X=a]$$

(1)

$X = X_1 + X_2$ $\quad X, X_1, X_2$ are random variables.

Then, $\mathbb{E}Y = \mathbb{E}X_1 + \mathbb{E}X_2$. ✓

Linearity of Expectation.

It does NOT impose any conditions on X_1, X_2.

$\rightarrow \mathbb{E}(X_1 X_2) = \mathbb{E}(X_1)\,\mathbb{E}(X_2)$ ✗ NOT true

X, X_1, X_2 : take values in $\{0, 1, 2, \ldots, n\}$

$$\mathbb{E}X = \sum_a a \cdot \Pr[X=a]$$

$$= \sum_a a \cdot \Pr[X_1 + X_2 = a]$$

$X_1 = 1, \; X_2 = a-1$

$X_1 = 2, \; X_2 = a-2$

$$= \sum_a a \cdot \sum_{b=0}^{n} \Pr[X_1 = b, \; X_2 = a-b] = \sum_a a \sum_{\substack{b,c \\ b+c=a}} \Pr[X_1 = b, \; X_2 = c]$$

$$= \sum_{b=0}^{n} \sum_{c=0}^{n} (b+c) \Pr[X_1 = b, \; X_2 = c]$$ ✓

$b=0 \quad c=0$ ← incorrect $\Pr[X_1=b] \cdot \Pr[X_2=c]$. ✓

$$= \sum_{b,c} b \cdot \Pr[X_1=b, X_2=c] + \sum_{b,c} c \cdot \Pr[X_1=b, X_2=c].$$

$$= \sum_b b \cdot \underbrace{\sum_c \Pr[X_1=b, X_2=c]}_{=\Pr[X_1=b]} + \cdots$$

$$\mathbb{E}X_1 + \mathbb{E}X_2.$$

$$X = (X_1 + X_2) + X_3$$
$$\mathbb{E}X = \mathbb{E}(X_1+X_2) + \mathbb{E}X_3$$
$$= \mathbb{E}X_1 + \mathbb{E}X_2 + \mathbb{E}X_3$$

$\mathbb{E}(X_1 + \cdots + X_n) = \mathbb{E}X_1 + \cdots + \mathbb{E}X_n$.

Application: (i) n unbiased coins $\quad \mathbb{E}[\#\text{heads}] = n/2$?

Indicator Random Variables: 0 or 1: $\mathbb{E}[Y] = 0 \cdot \Pr[Y=0] + 1 \cdot \Pr[Y=1]$
$= \Pr[Y=1]$.

$$X_i : \begin{cases} 1 & \text{if the } i^{\text{th}} \text{ coin toss is H} \\ 0 & \text{if the } i^{\text{th}} \text{ coin toss is T} \end{cases} \quad \mathbb{E}X_i = 1/2$$

$$\mathbb{E}[\#\text{heads}] = \mathbb{E}[X_1] + \mathbb{E}[X_2] + \cdots + \mathbb{E}[X_n]$$
$$= n/2.$$

Linearity of expectation

(ii) X random variable.

Suppose there is some other random variable Y. When $Y=i$, suppose $X = X_i$.

throw a dice, if dice shows i, toss an unbiased coin i times
X = # heads.

$$\mathbb{E}X = \sum_i \Pr[\text{dice} = i] \; \mathbb{E}[\#\text{heads when we throw } i \text{ coins}].$$
$$= \frac{1}{6}\left[\frac{1}{2} + \frac{2}{2} + \cdots + \frac{6}{2}\right]$$

$$X = \begin{cases} X_1 & \text{when } Y=1 \\ X_2 & \cdots \quad Y=2 \\ \vdots & \\ X_n & \cdots \quad Y=n \end{cases}$$

[Suppose Y is independent on the random variables $X_1, \ldots, X_n$. $\Rightarrow Z_1, Z_2, \ldots, Z_n$ are ind. of $X_1, \ldots, X_n$]

Then $\mathbb{E}X = \Pr[Y=1]\,\mathbb{E}X_1 + \Pr[Y=2]\,\mathbb{E}X_2 + \cdots + \Pr[Y=n]\,\mathbb{E}X_n$. ✓

Let $Z_i = \begin{cases} 1 & \text{if } Y=i \\ 0 & \text{otherwise.} \end{cases}$

$X = X_1 Z_1 + X_2 Z_2 + \cdots + X_n Z_n$.

$\mathbb{E}X = \mathbb{E}(X_1 Z_1) + \mathbb{E}(X_2 Z_2) + \cdots + \mathbb{E}[X_n Z_n]$

$= \mathbb{E}[Z_1] \cdot \mathbb{E}(X_1) + \cdots + \mathbb{E}(X_n)\,\mathbb{E}(Z_n)$

$= \Pr[Y=1]\,\mathbb{E}(X_1) + \cdots + \Pr[Y=n]\,\mathbb{E}(X_n)$.

---

(11) Selection (A, k) {

- pick an element x u.a.r. from A
- split A into

$\underbrace{< x}_{A_1} \times \underbrace{\geq x}_{A_2}$ $O(n)$ time.

- if $|A_1| = k-1$ ✓
- if $|A_1| \geq k$

  Selection (A_1, k)
- if $|A_1| < k-1$

  Selection $(A_2, k - |A_1| - 1)$

$T(n) = T(\alpha n) + O(n)$, $\alpha < 1$

---

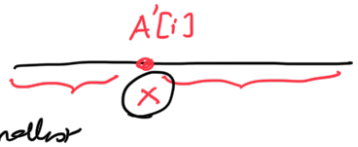
3. X_n : Expected Running Time on an array of size n.

→ T_n: # steps on an array of length n.

$X_n = \mathbb{E}\,T_n$.

→ $T_n = T_{i-1}$ or $T_{n-i} + O(n)$ if the element x is the ith smallest element

$T_n \leq \max(T_{i-1}, T_{n-i}) + O(n)$ if x is the i^{th} smallest

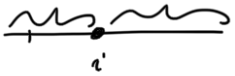


A' is the array A sorted in inc. order

→ $T_n \leq T_{\max(i-1,\, n-i)} + O(n)$ if x is the ith smallest # in A.

$\therefore$ $\mathbb{E}\,T_n \leq \sum_{i=1}^{n} \left[\mathbb{E}\,T_{\max(i-1,n-i)} + O(n)\right] \cdot \frac{1}{n}$ ✓

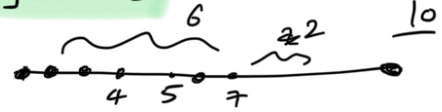
$X_n \leq \frac{1}{n} \sum_{i=1}^{n} X_{\max(i-1,n-i)} + O(n)$



$= \frac{1}{n}\left[X_{n-1} + X_{n-2} + X_{n-3} + \cdots + X_{n/2} + X_{n/2} + X_{n/2+1} + \cdots + X_{n-1}\right] + O(n)$

→ $X_n \leq \frac{2}{n}\left[X_{n/2} + X_{n/2+1} + \cdots + X_{n-1}\right] + O(n)$ ←

$\leq \frac{2}{n}\left[\frac{n}{4} \cdot X_{3n/4} + n/4 \cdot X_n\right]$ ←



$+O(n)$ $\quad X_9 + X_8 + X_7 + X_6 + X_5 + X_5 + X_6 + X_7 + \dots + X_9$.

$X_n \le X_{3n/4} + X_{\frac{n}{2}} + O(n)$ ✓

$X(n) \le X(an) + O(n)$, $a < 1$

$\boxed{X_n \le X_{3n/4} + O(n)}$ soln: $O(n)$

$X_i \ge X_{i-1}$. ✓

(ii) Randomized. Quicksort. n^2

$\boxed{X_{n/2} + X_{n/2+1}} \; \boxed{X_{n/2+2} \dots + X_n}$.

$\underbrace{X_{n/2} + \dots + X_{3n/4}}_{\le \frac{n}{4} \cdot X_{3n/4}} + \underbrace{X_{3n/4+1} \dots + X_n}_{\le \frac{n}{4} \cdot X_n}$

QuickSort(A, n) {

(i) choose $x \in A$ u.a.r.

(ii) Partition A:

A_1 | x | A_2 $\quad O(n)$.

(iii) QuickSort($A_1, |A_1|$), QuickSort($A_2, |A_2|$).

}

$T_n = T_{i-1} + T_{n-i} + O(n)$ w.p. $\frac{1}{n}$ (ie, if x is the ith smallest number).

$X_n = \frac{1}{n} \sum_{i=1}^{n} [X_{n-i} + X_{i-1}] + O(n)$

$\Rightarrow X_n = \frac{2}{n} [X_1 + \dots + X_{n-1}] + O(n)$ ✓

$O(n \log n)$.

we will solve this and show that $X_n = O(n \log n)$. ✓

10

$X_0 + X_9$

$X_1 \quad X_8$

$X_2 \quad X_7$

$\vdots$

$X_9 \quad X_0$

$T_n = [O(n) + T_{i-1} + T_{n-i}]$ $\quad \frac{1}{n} \cdot$ prob.

x : i^{th} smallest

$X_n = \mathbb{E}\, T_n$

$X_n = \frac{1}{n} \sum_{i=1}^{n} [O(n) + X_{i-1} + X_{n-i}]$

$\le \underbrace{O(n)}_{cn} + \frac{2}{n} [X_1 + X_2 + \dots + X_{n-1}]$.

$X_1 + X_2 + \dots + X_{n-2}$.

$$X_n = cn + \frac{2}{n}\left[X_1 + X_2 + \cdots + X_{n-1}\right]$$

$$X_{n-1} = c(n-1) + \frac{2}{n-1}\left[X_1 + X_2 + \cdots + X_{n-2}\right].$$

$$X_n = cn + \frac{2}{n}\left[X_{n-1} + \frac{(X_{n-1} - c(n-1))(n-1)}{2}\right]$$

$$= cn + \frac{2X_{n-1}}{n} + \left(1 - \frac{1}{n}\right)X_{n-1} - \frac{c(n-1)^2}{n}$$

$$= \frac{c(n^2 - 2n + 1)}{n} = c\left(n - 2 + \frac{1}{n}\right)$$

$$\leq \left(1 + \frac{1}{n}\right)X_{n-1} + 2c$$

$X_n \leq \left(1 + \frac{1}{n}\right)X_{n-1} + 2c$. solve this !

$$= \left(1 + \frac{1}{n}\right)\left[\left(1 + \frac{1}{n-1}\right)X_{n-2} + 2c\right] + 2c.$$

$$= \left(1 + \frac{1}{n}\right)\left(1 + \frac{1}{n-1}\right)X_{n-2} + 2c\left(1 + \frac{1}{n}\right) + 2c.$$

$$= \left(1 + \frac{1}{n}\right)\left(1 + \frac{1}{n-1}\right)\left(1 + \frac{1}{n-2}\right)X_{n-3} + 2c\left(1 + \frac{1}{n}\right)\left(1 + \frac{1}{n-1}\right) + 2c\left(1 + \frac{1}{n}\right) + 2c.$$

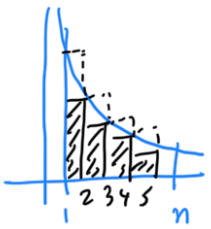
$i = n-1$

$$= \frac{n+1}{n-i}X_{n-i-1} + 2c\cdot\left[\frac{n+1}{n-1} + \frac{n+1}{n-2} + \cdots + \frac{n+1}{n-i}\right].$$

$$= n + 2c(n+1)\left[\underbrace{\frac{1}{n-1} + \frac{1}{n-1} + \frac{1}{n-3} + \cdots + 1}_{\ln n.}\right] = n + 2cn\ln n.$$

$$\ln n - 1 \leq 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} \leq \ln n + 1$$

$$\mathbb{E}[T_n] = O(n\log n).$$



$y = \frac{1}{x}$

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

$\frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$

n^2

$1, 2, 3, \ldots, n$

I

$$\frac{T_1 + \cdots + T_\ell}{\ell} \approx n\log n.$$

T_1
T_2
$\vdots$
T_ℓ

$$\Pr\left[T_i \geq 100\, n\log n\right] \leq 2^{-n}$$

Dynamic Programming: same as divide & conquer except that we "store" some of the recursive function calls.

$F_0 = F_1 = 1$: $F_n = F_{n-1} + F_{n-2}$.

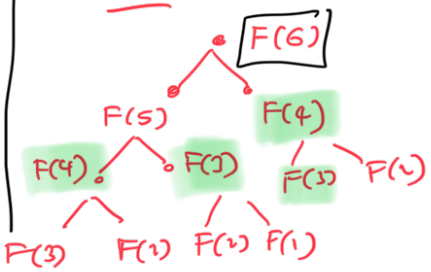
1, 1, 2, 3, 5, 8, 13, 21, 34, ... →

```
F(n) {
    if n = 0 or n = 1
        return 1
    else
        return F(n-1) + F(n-2)
}.
```

the same recursive call is being made multiple times.

running time is exponential in n.

n = 20.



1. Divide & Conquer Alg. : the running time is high because we are making the same recursive calls again and again

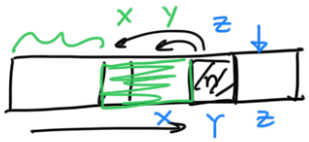
2. What are all the possible recursive calls?

$F(k)$, $k \le n$.

store the values for them in a DP Table.

A ✓ $A[k] = F(k)$

3. Compute the entries of A in a manner such that when we want to compute entries in the order such that no recursive call is needed.



$A[n] = A[n-1] + A[n-2]$ if $n \ge 3$,

$n = 1, 2$.

```
A[1] = A[2] = 1.
for i = 3 ... n
    A[i] = A[i-1] + A[i-2].
```

x = y + x

y = x

z = y + x

save space! O(1) space.

$x = y$
$y = z$

---

② Knapsack Problem:

items — indivisible — $1, \ldots, n$

item i: size s_i, profit p_i

Problem: Find a maximum profit subset of items which fit in the bag.

p_i / s_i

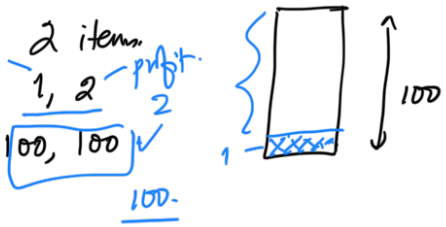
Divide & Conquer Alg.

subset of items.

int Knapsack (B, 1, 2, .., n) { ✓

bag cap — set of items.

if n=1 ....



item 1 — take this item. / not take this item.

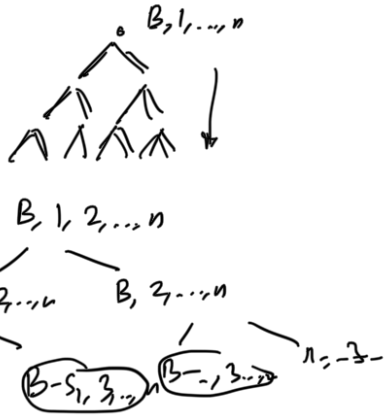
return max { Knapsack (B − s_1, 2, .., n) + p_1 ← if we choose to select item 1.
Knapsack (B, 2, .., n) }

}

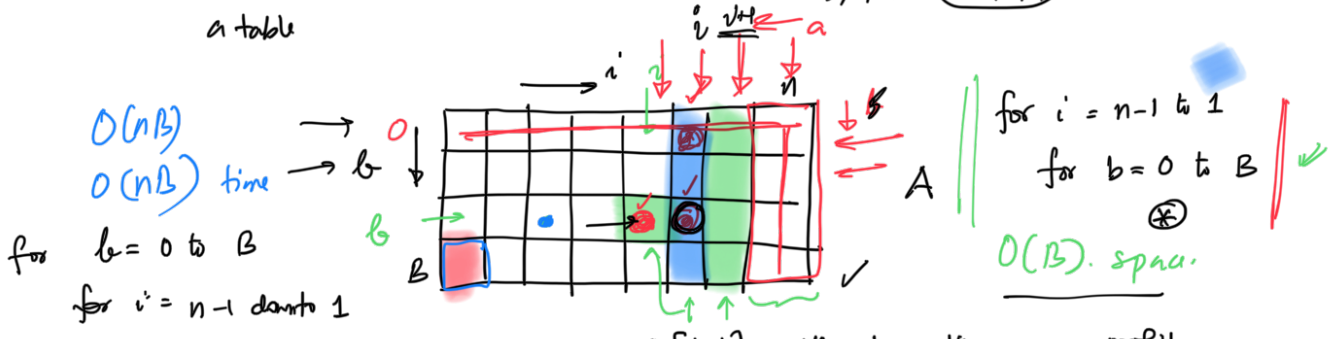
What are the possible recursive calls?

(b, i, .., n) — Bn

$0 \le b \le B$



Store all of these recursive calls in a table



O(nB)

O(nB) time

for b = 0 to B

for i = n−1 down to 1

A[b,i] will store the max. profit that we can get when bag has cap b and the items are $i, \ldots, n$

only if $b \ge s_i$

for i = 0 ... n

for b = 0 ... B ||x

A[B,1]

$$A[b,i] = \max\left(A[b-s_i, i+1] + p_i,\ A[b, i+1]\right) \quad (*)$$

Knapsack $(b, \underbrace{i, \ldots, n}_{=})$ {

$$\max \begin{cases} \text{Knapsack}(b - s_i, i+1, \ldots, n) + p_i & \text{only if } b \geq s_i \\ \text{Knapsack}(b, i+1, \ldots, n) \end{cases}$$

}.

bits to write down the input

x^2 x^3 ...

(1) Is this an efficient algorithm? NP Complete.

$O(nB)$. ← is this polynomial time?

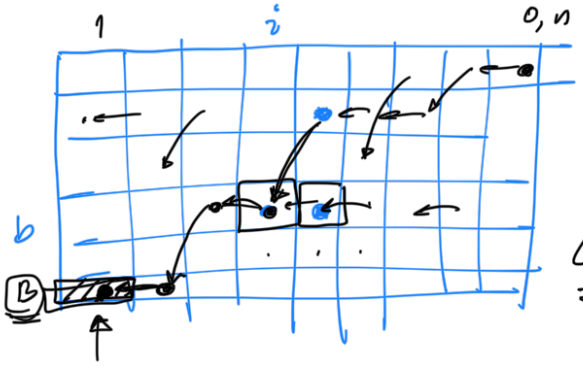
$n \log B$

exponential time alg.

(2) What if we also want to find out the actual set of items to pick?

$B = 10^{200} + 1$

201 bits.



$O(nB)$