Date: 20/04/23

Q. $\lim_{n\to\infty} \frac{7n^2-2n-5}{5n^2+3n+10} = \lim_{n\to\infty} \frac{14n-2}{10n+3} = \frac{7}{5}$

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

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

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \frac{5}{7}$

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

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

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

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

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

Q. $f(n) = n+10$

$g(n) = 5n^2 - 6n + 10$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n+10}{5n^2-6n+10} = 0$

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

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$ So, $g(n) \nleq f(n)$

$f(n) = \Omega(g(n))$ iff $\exists$ constants $c>0$, $n_0>0$

$\therefore f(n) \geq c \cdot g(n)$ for all $n \geq n_0$

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

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

$a \leq b$ iff $b \geq a$

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

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \quad , 0 < c \leq \infty$$

i) $f(n) = 7n^2 - 2n - 5$

$g(n) = 5n^2 + 3n + 10$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \frac{7}{5}$ $\therefore$ $f(n) \geq g(n)$ asymptotically

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

ii) $f(n) = n + 10$

$g(n) = 5n^2 - 6n + 10$

$\lim_{n\to\infty} \frac{g(n)}{f(n)} = \infty$ $\therefore$ $g(n) = \Omega(f(n))$

$f(n) = an^d + \text{---}$ $d \leq r-1$

$g(n) = bn^r + \text{---}$

$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0 \Rightarrow F(n) \leq g(n)$

$f(n) = \Theta(g(n))$ iff $\exists$ constants $c_1, c_2 > 0$, $n_0 > 0$

$c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$ for all $n \geq n_0$

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

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = c \quad , 0 < c < \infty$$

$\longrightarrow$ $f(n) = g(n)$ asymptotically.

$f(n) = o(g(n))$ iff for every constant $c > 0$, $\exists n_c > 0$

$f(n) < c\,g(n)$ for all $n \geq n_c$

Here, $f(n)$ is much smaller than $g(n)$

Not only differ by a constant

Equivalent definition

$f(n) = o(g(n))$ iff

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$$

Notation: $f(n) < g(n)$

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

$f(n) > c \cdot g(n)$ for all $n \geq n_0$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty$$

$f(n) = o(g(n))$ iff $g(n) = \omega(f(n))$

$a < b$ iff $b > a$

Example :

i) $f(n) = 6n^3 + 7n + 100$

$g(n) = 8n^2 + 10$

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty \quad \therefore f(n) > g(n)$$

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

ii) $f(n) = \log^2 n$, $g(n) = \frac{1}{2}\sqrt{n}$

$$\lim_{n \to \infty} \frac{\log^2 n}{\sqrt{n}/2} = \lim_{n \to \infty} \frac{(\log n)^2}{\sqrt{n}/2} = \lim_{n \to \infty} \frac{2 \log n \times \frac{1}{n}}{\frac{1}{2}\sqrt{n}} = 0$$

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

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

$\because \log^2 n < \sqrt{n}$

$\log^2 n = o(\sqrt{n})$

iii) $f(n) = 2^n$, $g(n) = 3^n$

$$\lim_{n \to \infty} \frac{2^n}{3^n} = \lim_{n \to \infty} \left(\frac{2}{3}\right)^n = 0$$

$2^n = O(3^n)$

iv) $$\lim_{n \to \infty} \frac{\log^{100} n}{n^{\frac{1}{200}}} = 0$$

v) Prove : $a^n = O(b^n) \; \forall \; a < b$

Date: 27/04/23

S.NO	city	Timestamp	Temp(C)
1	Delhi	5:00	27
2	Bangalore	6:00	21
3	Delhi	12:00	31
4	Bangalore	13:00	28
5	Delhi	15:00	32

In place & stable

Binary Search

Worst Case : $W(n) = W(\frac{n}{2}) + 2$

↓ Comparison

Recurrence Relation

i) $T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n-1)+1 & \text{if } n>1 \end{cases}$

→ $T(n) = T(n-1)+1$

$T(n-1) = T((n-1)-1)+1$

$= T(n-2)+1$ —①

$T(n-2) = T((n-2)-1)+1$

$= T(n-3)+1$ —②

$\therefore T(n) = (T(n-2)+1)+1$

$T(n) = [T(n-3)+1]+1+1$

!

$T(n) = [T(n-k)+1]+1+1+\cdots$ k-1 times

$n-k = 1$

$\boxed{k = n-1}$

$\therefore T(n) = T[(n-n+1)]+k$

$= T(1)+n-1$

$= 1+n-1$

$= n$

ii) $T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(n-1)+n & \text{if } n>1 \end{cases}$

→ $T(n) = T(n-1)+n \quad \text{if } n>1$

$T(n-1) = T((n-1)-1)+(n-1)$

$= T(n-2)+(n-1)$

$T(n-2) = T((n-2)-1)+(n-2)$

$= T(n-3)+(n-2)$

$T(n) = T(n-2)+(n-1)+n$

$T(n) = T(n-3)+(n-2)+(n-1)+n$

⋮

$T(n) = T(n-k)+[n-(k-1)]+(n-(k-2))+---+n$

$n-k = 1$

$k = n-1$

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

$= n-n+1+1$

$= 2$

$\therefore \quad T(n) = T(n-(n-1))+[2+3+4+---+n]$

$= T(n-n+1)+[2+3+4+---+n]$

$= T(1)+2+3+4+--+n$

$= 1+2+3+4+---+n$

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

Date : 02/05/23

Q. $T(n) = \begin{cases} c\,0 & \text{if } n=1 \\ 2T(\frac{n}{2}) + n^{1} & \text{if } n>1 \end{cases}$

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

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

$T(n) = T(\frac{n}{4}) + 1 + 1$

$T(\frac{n}{4}) = T(\frac{n}{8}) + 1$

$T(n) = T(\frac{n}{8}) + 1 + 1 + 1$

$T(n) = T(\frac{n}{2^3}) + 1 + 1 + 1$

$\vdots$

$T(n) = T(\frac{n}{2^k}) + 1 + 1 + 1 + \cdots + k \text{ times}$

$\frac{n}{2^k} = 1$

$n = 2^k$

$\log n = \log 2^k$

$\log n = k \log 2$

$\boxed{k = \log_2 n}$

$T(n) = T(\frac{n}{2^{\log_2 n}}) + \log_2 n$

$= T(1) + \log_2 n$

$= \log_2 n$

Q. $T(n) = \begin{cases} c & \text{if } n=1 \\ 2T(\frac{n}{2}) + n & \text{if } n>1 \end{cases}$

→ $T(n) = 2T(\frac{n}{2}) + n$

$T(\frac{n}{2}) = 2T(\frac{n}{4}) + \frac{n}{2}$ —(I)

$T(\frac{n}{4}) = 2T(\frac{n}{8}) + \frac{n}{4}$ —(II)

$T(n) = 2 \cdot (2T(\frac{n}{4}) + \frac{n}{2}) + n$

$T(n) = 2 \cdot (2 \cdot 2T(\frac{n}{8}) + \frac{n}{4}) + \frac{n}{2}) + n$

Q. $T(n) = \begin{cases} 1 & \text{if } n=1 \\ T(\frac{n}{2})+n & \text{if } n>1 \end{cases}$

$\rightarrow$ $T(n) = T(\frac{n}{2})+n$ if $n>1$

$T(\frac{n}{2}) = T(\frac{n}{4}) + \frac{n}{2}$ —(i)

$T(n) = \left(T(\frac{n}{4}) + \frac{n}{2}\right) + n = \left[T(\frac{n}{2^2}) + \frac{n}{2}\right] + n$

$T(\frac{n}{4}) = T(\frac{n}{8}) + \frac{n}{4}$ —(ii)

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

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

$\vdots$

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

$\frac{n}{2^k} = 1$

$k = \log_2 n$

$T(n) = T(\frac{n}{2^{\log_2 n}}) + \frac{n}{2^{\log_2 n - 1}} + \frac{n}{2^{\log_2 n - 2}} + \cdots + n$

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

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

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

$= T(\frac{n}{2^k}) + n\left[1 + \frac{1}{2} + \frac{1}{2^2} + \cdots + \frac{1}{2^{k-1}}\right]$

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

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

$= 1 + 2(n-1)$

$= 2n - 1$

Sum of GP $= \frac{a(r^n - 1)}{r - 1}$

Q. $T(n) = \begin{cases} 1 & \text{if } n=1 \\ 2T(\frac{n}{2})+1 & \text{if } n>1 \end{cases}$

→ $T(n) = 2T(\frac{n}{2})+1$

$T(\frac{n}{2}) = 2 \cdot T(\frac{n}{4})+1$ —①

$T(\frac{n}{4}) = 2T(\frac{n}{2^3})+1$ —②

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

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

$T(n) = 2^2 \cdot [2T(\frac{n}{2^3})+1]+2 = 2^3 T(\frac{n}{2^3}) + (2^2+2+1)$

!

$T(n) = 2^k \cdot T(\frac{n}{2^k}) +$ ~~[illegible] k times~~ $(2^{k-1} + \dots + 2+1)$

$\frac{n}{2^k} = 1$

$k = \log n$

~~$T(n) = 2 \log \frac{n}{2} + \log \frac{n}{2}$~~

~~$T(n) = n + \log \frac{n}{2}$~~

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

$= n + n - 1$

$T(n) = 2n - 1$

Q. $T(n) = \begin{cases} 0 & , \text{if } n=1 \\ 2T(\frac{n}{2})+n & , \text{if } n>1 \end{cases}$

→ $T(n) = 2T(\frac{n}{2})+n$

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

$T(\frac{n}{4}) = 2T(\frac{n}{2^3}) + \frac{n}{4}$ —②

$T(n) = 2[2T(\frac{n}{2^2}) + \frac{n}{2}] + n$

$= 2^2 T(\frac{n}{2^2}) + 2 \times \frac{n}{2} + n = 2^2 T(\frac{n}{2^2}) + n + n$

$T(n) = 2^2 [2T(\frac{n}{2^3}) + \frac{n}{4}] + n + n$

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

$T(n) = 2^k T\left(\frac{n}{2^k}\right) + n + n + \cdots + k \text{ times}$

$T(n) = 2^k T\left(\frac{n}{2^k}\right) + nk$

$\frac{n}{2^k} = 1$

$k = \log_2 n$

$\boxed{T(n) = n + n\log_2 n}$

Binary Search

In worst case:

$W(n) = W\left(\frac{n}{2}\right) + 1$

$= \Theta(\log n)$

Best case: $W(n) = 1$

Merge Sort

Example having $m+n-1$ Comparisons

1	3	5	7

2	4	6	8

Best case : $\min(m, n)$

Worst case : $n+m$

Implementation of merge sort with using atmost n extra

Date: 04/05/23

Quick Sort

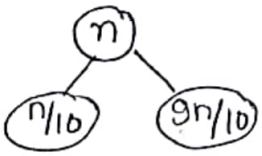
$T(n) = T(q-p) + T(r-q) + n$ ← for Partition

$p = 1 \;, r = n$

$T(n) = T(q-1) + T(n-q) + n$

$T(n) = T\left(\frac{n}{10}\right) + T\left(\frac{9n}{10}\right) + n$

$= \Theta(n \log n)$



Date : 08/05/23

Comparison Sorts

coorst case : $\Omega(n \log n)$

Count Sort

Time Complexity : $n+m$

Space complexity : $n+m$

WAP to implement Count sort by using cumulative Frequency

4	5	2	0	4	2	0	2

Frequency table

index	0	1	2	3	4	5
CF Count	2	2	5	5	7 (6)	8

Output

index	1	2	3	4	5	6	7	8
data	0	0	2	2	2	4	4	5

time $= \Theta(n+m)$

Space $= n+m$.

Not inplace, ~~not stable~~

Scan from last then it is stable.

Date : 09/05/23

Data

110014	110005	110002	110020	110004	110022	110010	100012

Frequency table

data	0	2	4	5
Freq	2	3	2	1

CF

index	0	1	2	3	4	5
count	2 (1)	2	5 (4)	5	7 (6)	8

Output

index	0	1	2	3	4	5	6	7	8
data		110020	110010	110002	110022	110012	110014	110004	110005

Sorting a/c 2nd last digit

Frequency table

data	0	1	2
Freq	3	3	2

CF

index	0	1	2	3
count	~~3~~ 2	6	8	8

Output

index	1	2	3	4	5	6	7	8
data	110002	110004	110005	110010	110012	110014	110020	110022

Scanning From Front

index	0	1	2	3	4	5
Count	2	2	5	5	~~7~~ 6	8

Output:

index	1	2	3	4	5	6	7	8
data	110010	110020	110012	110022	110002	110004	110014	110005

Sorting a/c 2nd last digit

Frequency table

data	0	1	2
Freq.	3	3	2

CF

index	0	1	2	3
Count	~~3~~ 2	~~6~~ 5	~~8~~ 7	8

Output

index	1	2	3	4	5	6	7	8
data	110005	110004	110002	110014	110012	110010	110022	11007

Radix Sort

329	720	720	329
457	355	329	355
657	436	436	436
839 →	457 →	839 →	457
436	657	355	657
720	329	457	720
355	839	657	839

Time Complexity
$= C(n+m)$

$0 \dots 10^3 - 1$

$m = 10$ (radix)

$C = 3$

$0 \dots 9$ (range)

For one run of counting sort
$(n+10)^{2m}$ space
$(n+10)^{2m}$ time

Use counting sort as stable algorithm

3 run total time required $= 3(n+10)$

space required $= n+10$

→ $(0 \dots m^c - 1)$ can be represented as

␣ ␣ ... ␣

c base-m digit

If we use counting sort then time & space would be $n+m^c$ which is not linear

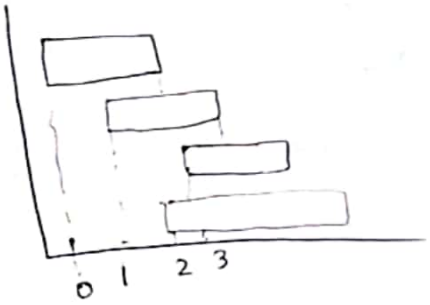
329	329	720
355	720	355
457	436 →	436
436 →	839	457
657	355	657
720	457	329
839	657	839

Minimum Spanning tree

Date : 11/05/23

Greedy algorithm

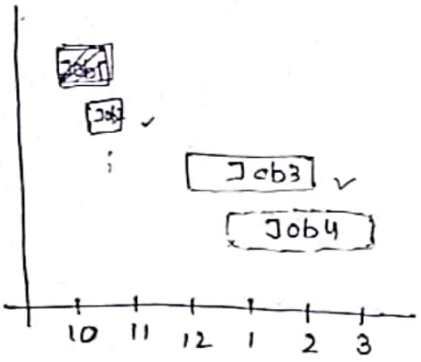
incr. Starting time



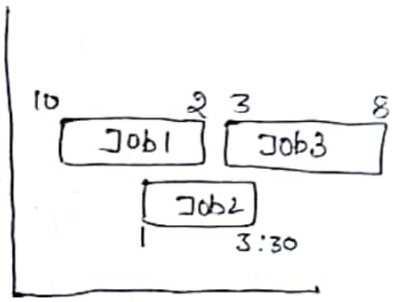
(I) Shortest Job First
(II) Earliest finishing time first
(III) Earliest start time first

Will (I) always give me an optimal soln ?

Job1 - 10 : 11
Job2 - 10:30 to 11
Job3 - 12 : 2
Job4 - 12:30 to 3

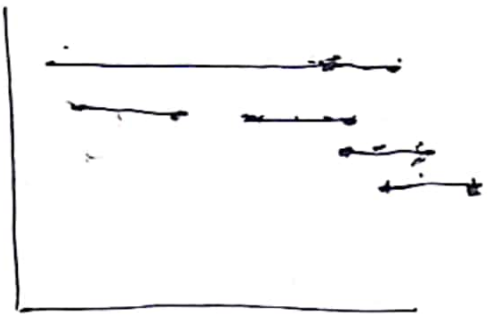


i) Job2, Job3
ii) Job1, Job3
iii) Job1, Job3



i) Job 2 Here (I) will not work

(III)

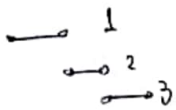


***Greedy

Date: 16/05/23

Assume:

Whether two intervals conflict with each other can be checked in constant time.



Conflict (q_i, q_j) → yes, NO

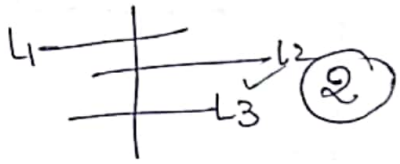
Weaker algorithm implement

We will check conflict 1 with 2, it it does not conflict we will include this & next time we will compare with the new one

$|S| \leq d \leq OPT$

algorithm Also $OPT \leq |S|$

$OPT = |S| = d$



$\{1, 2, \ldots, d\}$

↓ no. of rooms

Sort the intervals in increasing order of start time

Scheduling Jobs with processing times & deadlines

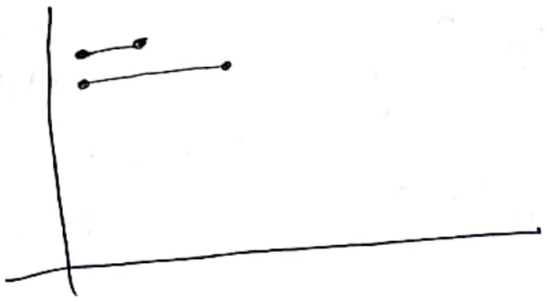
Jobs	p_i ↓	d_i ↓	L_1	L_2	L_3
J_1	2	5	0	2	4
J_2	3	4	1	0	3
J_3	4	6	3	3	0
			Max 3	Max 3	4

Case 1: J_1, J_2, J_3 → optimal

Case 3: J_3, J_2, J_1

Case 2: J_2, J_1, J_3 → optimal

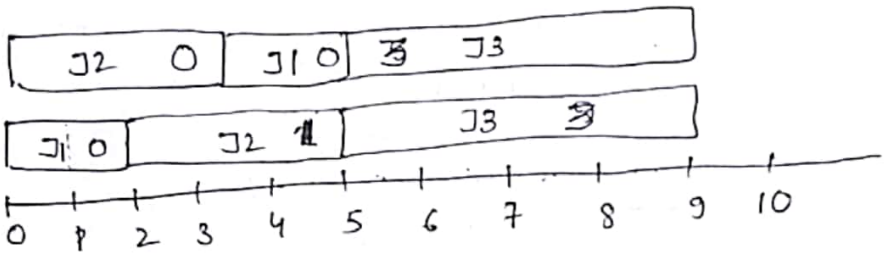
* Shortest Job first not work



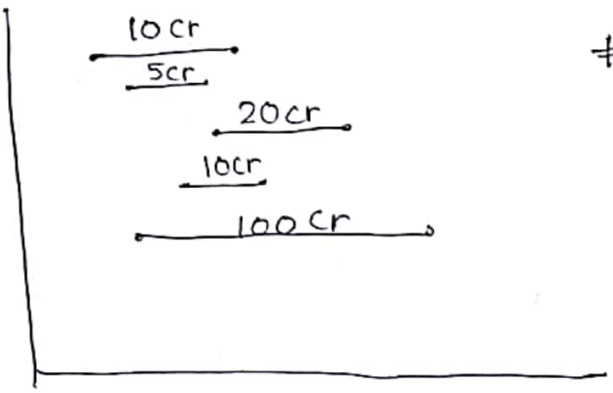
Slack time = $d_i - p_i$

Jobs	p_i	d_i	s_i
J_1	1	11	10
J_2	10	10	0

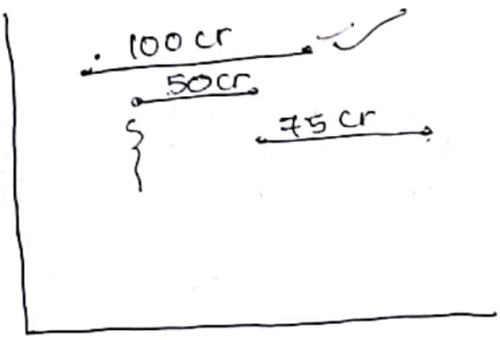
Jobs	P_i	d_i	S_i
J_1	2	5	3
J_2	3	4	1
J_3	4	6	2



#



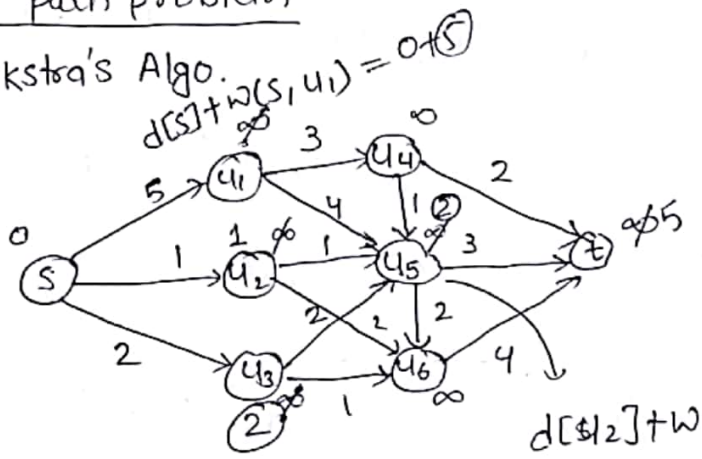
#



Date: 18/05/23

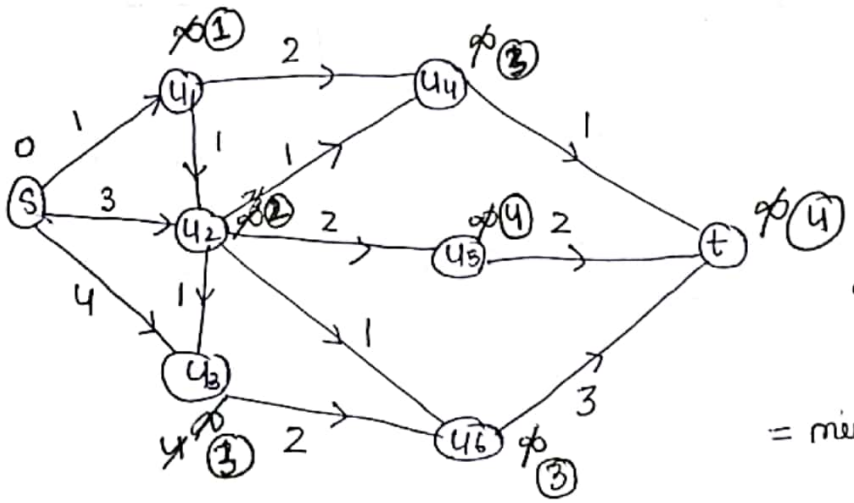
Shortest path problem

Djikstra's Algo.



d[u2]+w(u2,u5)

Π(v) → Parent of v



Initially

$d[s] \leftarrow 0$

$d[u] = \infty$

$\forall u \neq s$

$d(u) = \min\{$

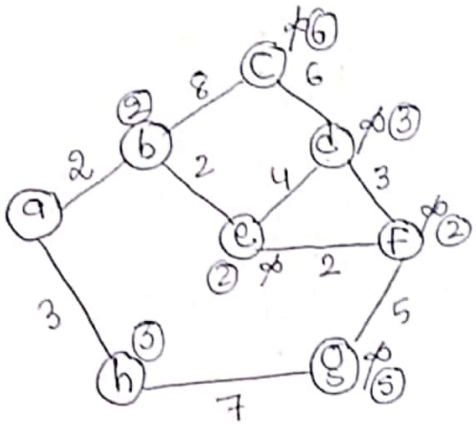
$d(u)$, new value?

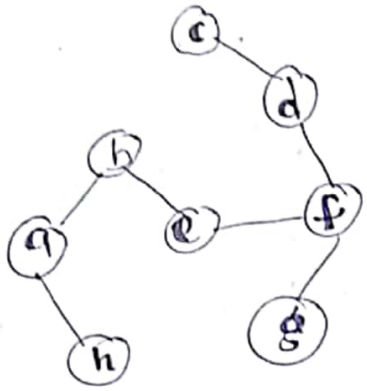
$= \min\{ d(u), d(x) + w(x,u) \}$

Date : 22/05/23

Prim's Algorithm

Connectivity with minm cost





$O((E+V)\log V)$

Cleinberg - merge sort
Interval scheduling/Job Scheduling - Cormen

Kruskal algorithm

I) Sort the edges in the increasing order of their weights

II) while there are more edges and we have selected < n-1 edges do
select the next edge if it does not form a cycle and discard it otherwise.

How to find whether new edge added will form a cycle or not

Let find(u) be a function that returns the component to which it belong.

For the next edge e(u,v)
if find(u) == find(v) then discard
else
include the edge

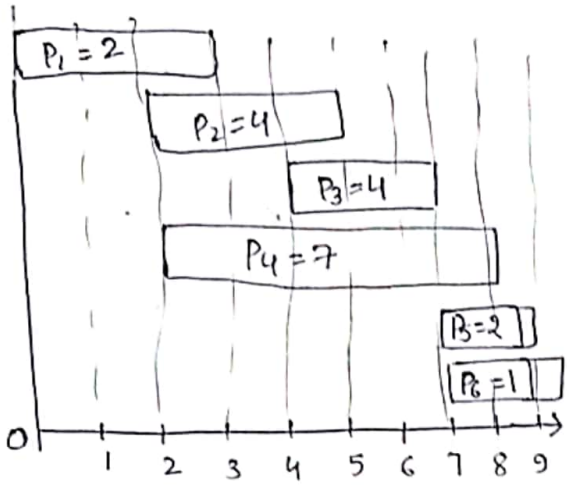
Find is called m no of times
↓
no of edges

Dynamic Programming

↓

bottom up

Eg:- Fibonacci sequence time complexity for finding nth term



C[1] $P_1 = 0$
C[2] $P_2 = 0$
C[3] $P_3 = 1$
C[4] $P_4 = 0$
C[5] $P_5 = 3$
C[6] $P_6 = 3$

Opt(6)

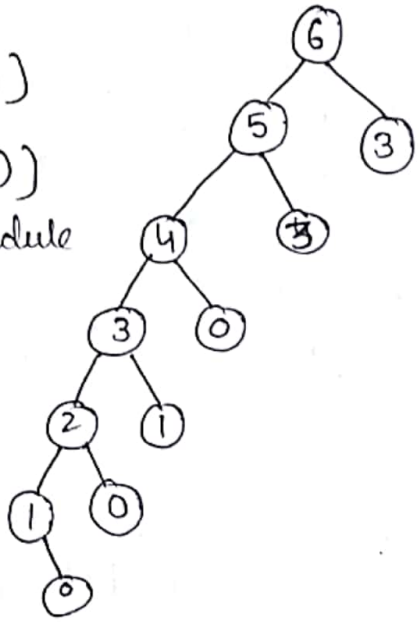
Opt[C[3]]

(6)

$Opt[n] = \max([P_n + Opt(C[n])], Opt(n-1))$

$Opt[6] = \max([P_6 + Opt(C[6])], Opt(5))$

Opt(n) → profit of the optimal schedule from the first n jobs.



opt[j]	0	2	4	5	7		
j	0	1	2	3	4	5	6

$Opt[j] = \max(P_j + Opt[C[j]], Opt[j-1])$

$Opt[1] = \max(P_1 + Opt[C[1]], Opt[0])$

$Opt[2] = \max(P_2 + Opt[C[2]], Opt[1]) = \max[4, 2]$

Opt[5] =

Date : 25/05/23

***Brute force algo

∀ possible soln s
check if s satisfies our criteria.

Sorting
I/p: array
O/p: all possible permutation
Time complexity = O(n!)

$\Omega(n!) = \Theta(n^n)$

Interval scheduling

Input : $\langle s_i, f_i \rangle$ $i = 1, \ldots n$

max ← 0

∀ possible subsets S
check if S is feasible such that no two intervals are conflicting
iff feasible max ← max{max, |S|}

Shortest path problem

I/p : $G = (V, E)$

O/p: Sequence of edges/vertices

Minimum spanning tree

O/p: ∀ Subset of edges $E \pm \Omega(2^{|E|})$

weighted interval scheduling

Input : $\langle s_i, f_i, P_i \rangle$ $i = 1$ to n

O/p: ∀ possible subsets of intervals (Brute force)

DP soln:

i) Arrange the intervals in increasing order of their finishing time
Let this order be $I_1, I_2, \ldots I_n$

ii) Consider the nth job and decide whether to include nth job in my solution or not

Let opt(n) denote the optimal solution value from the first n jobs.

not include : opt(n) = opt(n-1)
include : opt(n) = opt(c(n)) + P(n)

opt(n) = max{opt(n-1), opt(c(n)) + P(n)}

Tute Rajni ma'am Date: 25/05/23

Date: 29/05/23

Matrix multiplication

$A_1 A_2 A_3$: $(A_1A_2)A_3 \rightarrow (2\times3\times4)+(2\times4\times5) = 64$

$2\times3 \quad 3\times4 \quad 4\times5$ $\quad A_1(A_2A_3) \rightarrow (2\times3\times5)+(3\times4\times5) = 30+60 = 90$

$\begin{bmatrix} \otimes & \otimes & \otimes \\ x & x & x \end{bmatrix}_{2\times3} \begin{bmatrix} \otimes & x & x \\ \otimes & x & x \\ \otimes & x & x \end{bmatrix}_{3\times3} \rightarrow \begin{bmatrix} x & & \\ & & \end{bmatrix}_{2\times3}$ 3 Multiplications Per element of the product

Total $\rightarrow 2\times3\times3$

$[\ ]_{m\times n}\ [\ ]_{n\times p} \rightarrow [\ ]_{m\times p}$ $(m\times n\times p)$ ↓ multiplication

\# $A_1 A_2 \cdots A_n$

$d_1\times d_2,\ d_2\times d_3 \cdots d_n\times d_{n+1}$

Search best K s.t

$(A_1^{d_1\times d_{k+1}} \cdots A_k)(A_{k+1}^{d_{k+1}\times d_{n+1}} \cdots A_n)$

↳ best split

opt(1,n) be the best solution value to multiply matrices $A_1 \cdots A_n$

Consider the split at some K

$Opt_K(1,n) = \min(Opt(1,K) + Opt(K+1,n) + d_1 \times d_{K+1} \times d_{n+1})$

~~void findmult(int mat[][], int m, int n)~~

~~{ if (~~

↑ to multiply two matrices

multiply(A,B)

{ i,j ; if (A[j] == B[i])

multiply_seq_of_matrices ($A_1, A_2, \cdots A_n$)

{ if (K<n)

P

merge sort (arr, low, high)

{ while (low < high)

{ int mid = (low+high)/2;

mergesort (arr, low, mid)

mergesort (arr, mid+1, high)

merge (arr, low, high, mid)

} }

mergesort(arr, low, K)

mergesort(arr, K+1, n)

Opt(i,j) be the best soln value to multiply values from $A_i, \ldots, A_j$

$Opt(i,j) = \min_k \{ Opt(i,k) + Opt(k+1,j) + d_i d_{k+1} d_{j+1} \}$

No of Subproblems = no of cells = n^2 (general)

Date: 30/05/23

Opt(i,j) → minimum no of multiplications required to multiply $(A_i, \ldots A_j)$

$Opt(i,j) = \min_k \{ A_k + B_k + C_k \} = O(n)$

(constant)

	A_1	A_2	A_3	A_4	A_5	A_6
A_1	0	60	70	112	130	202
A_2		0	40	110	112	218
A_3			0	56	66	154
A_4				0	42	90
A_5					0	168
A_6						0

$A_1 = 3 \times 5$

$A_2 = 5 \times 4$

$A_3 = 4 \times 2$

$A_4 = 2 \times 7$

$A_5 = 7 \times 3$

$A_6 = 3 \times 8$

$A_1 A_2 A_3 = \min \{ (A_1 A_2) A_3 , A_1 (A_2 A_3) \}$

$A_4 A_5 A_6 = \min \{ (A_4 A_5) A_6 , A_4 (A_5 A_6) \}$

$= \{ 2 \times 7 \times 3 + 2 \times 3 \times 8 , \; 2 \times 7 \times 8 + 7 \times 3 \times 8 \}$

$= \{ 42 + 48 , \; 112 + 168 \}$

$= \{ 90 , 280 \}$

$A_1 A_2 A_3 A_4 \rightarrow$ 3 split

$(A_1 A_2 A_3) A_4$

$(A_1 A_2)(A_3 A_4)$

$A_1 (A_2 A_3 A_4)$

$= \min (70 + 3 \times 2 \times 7 , \; 60 + 56 , \; 110 + 3 \times 5 \times 7)$

$= \min (112 , 116 , 215)$

$= 112$

$A_2 A_3 A_4 A_5 \rightarrow (A_2 A_3 A_4) A_5$

$(A_2 A_3)(A_4 A_5)$

$A_2 (A_3 A_4 A_5)$

$= \min (110 + 5 \times 7 \times 3 , \; 40 + 42 , \; 5 \times 4 \times 3 + 66)$

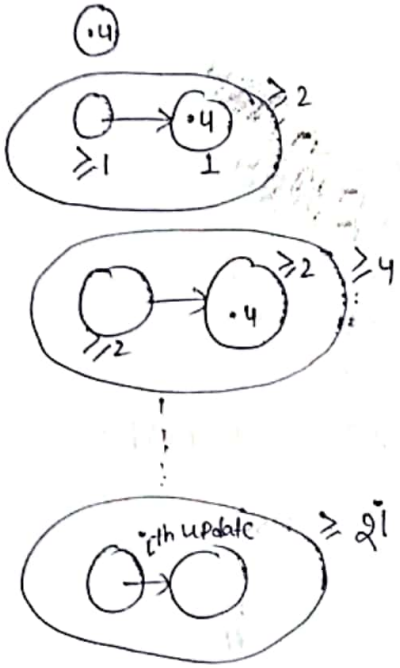
$= \min (215 , 82 , 126)$

Kruskal analysis

Total time over all union opr $= \sum_{u \in V}$ No of times Parent pointer is updated for vertex u over all unions

$\leq \sum_{u \in V} n \log n$

$= n \log n$



For a vertex its pointer is updated atmost $\log_2 n$.

$2^i = n$

$i = \log_2 n$

① Sort the edges = $O(m \log m)$

② No of find operations → $O(|E|)$

③ No of union operations → $n-1$ edges

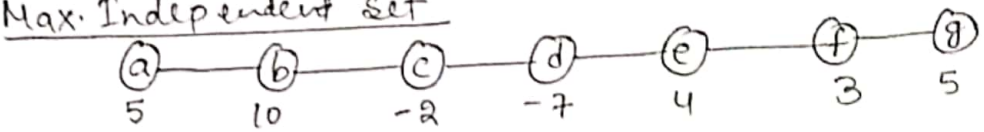
$O(m \log m) + O(m \log n) = O((m+n) \log m)$

Date : 01/06/23 Aditya Pancholi sir

DP
↳ Efficient brute force

Linked list → Line Graph

Max. Independent set



Independent set
{ }
{a}
{a, c}
{b}
⋮

① Store in an array

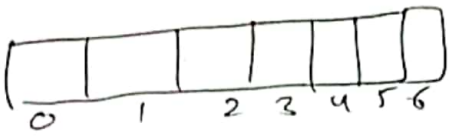
5	10	-2	-7	4	3	5

② Start chopping from back

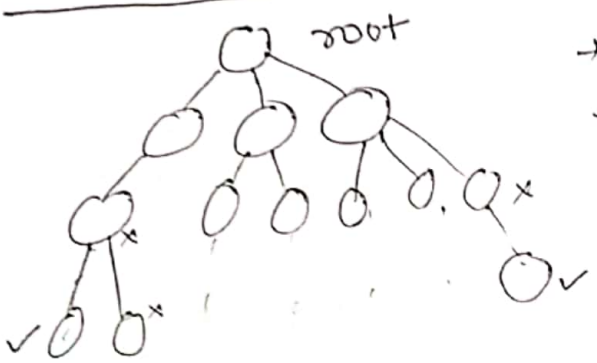
A

5	10	-2	-7	4	3	5
0	1	2	3	4	5	6

MIS =



Max Independent set on a tree



→ no knowledge of child
→ Start from root

① root included
↳ root → child → child + (root → child → child → child)

② root not included
↳ root → child

root

Goal = MIS[root]

MIS[i] → MIS that can be achieved on the tree rooted as 'i'.

Base case:

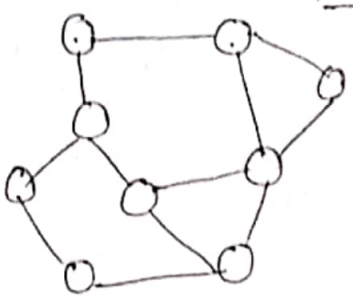
MIS[leaf node] = { 0 if data → leaf }

MIS[0] = 0

$$MIS[i] = \max \begin{cases} i \text{ is included} \\ \text{Value}[i] + \sum_{j} MIS[j] \text{ where } j \text{ is the grandchild of } i \\ i \text{ is not included} \\ \sum_{j} MIS[j] \text{ where } j \text{ is the child of } i \end{cases}$$

MIS on a graph

NP-hard



Quiz Solution

(1) sorted array contains 2^n elements → time $O(2^n)$ to search

(II) Binary search case → $O(\log 2^n) \rightarrow O(n)$

(2) Sorted array given

(I) Insertion sort time taken – $O(n)$

(II) Merge sort $O(n \log n)$

(III) Quick sort $O(n^2)$

(IV) Count sort $O(n+K)$ where K is the largest element

(3) (a) false

A_1 algo → $O(n^2 \log n)$ time in worst case

A_2 → Best time $\Omega(n \log n)$

} no comparison as these are loose bounds

(b) false

(c) false

(d) false Binary search & Ternary search time complexity are approx same.

$$\log_b^a = \log_c^a$$

(e) $O(f(n)) \cap \Omega(f(n)) = \phi \rightarrow$ false

↓ Average is not a property of set

$\Theta(f(n))$

this line is the intersection

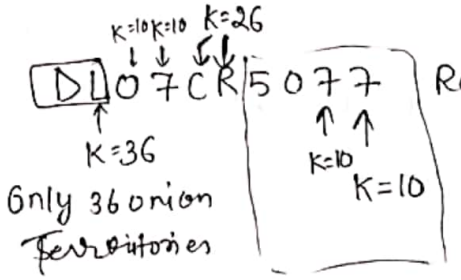
Q4. An array which has exactly 10 distinct elements. Use Quick sort and pivot is choosen randomly. For worst case position of partition is

↓ either largest or smallest

$\frac{1}{10} + \frac{1}{10} = 0.2$

Q5. $O(n^2)$ = $\sqrt{O(n^2)}$ element
time = $O(n)$ element
element sorted

If time is $O(n^2 \log n)$ then element sorted $\sqrt{n^2 \log n}$
$= n\sqrt{\log n}$ element

Q6. 

Radix sort

$O(d(n+K)$
↓ constant
$O(d(n)$
↓ no of digits

Q7. Given, 1, 2, ---- N nos
n! permutation

For insertion sort

1 2 3 --- N $O(n)$ time

$n, n-1, n-2, \ldots 3, 2, 1$ $\Theta(n^2)$

$\Theta(n \log n)$ time → which permutation
↓
$\log n$ elements each taking $O(n)$ time

$(n - \log n)$ elements ---- $O(1)$ time

$n - \log n + 1, n - \log n + 2, \ldots n \log n, \log n - 1, \log n - 2, \ldots 2, 1.$

Date : 05/06/23

Knapsack Problem : Given a set of objects with weights w_i and profit v_i and a knapsack capacity K

w_i	v_i
20kg	4000
30kg	1500
10kg	700

Suppose we are allowed to pick objects partially

Greedy: Pick in the order of $\frac{\text{profit}}{\text{weight}}$

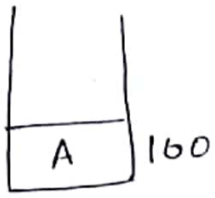
w_i	v_i
30 kg	1000
20kg	800
10 kg	500

K = 30kg

Greedy:
Pick or leave in decreasing order of Profit/weight

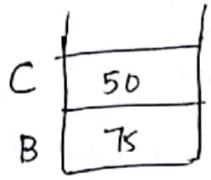
w_i	v_i	v_i/w_i
20 kg	1500	75
10 kg	1000	100
40 kg	2000	050

K=30



	w_i	v_i	v_i/w_i
A	20kg	2000	100
B	10 kg	750	75
C	10kg	500	50

K=20



optimal

$x_1, x_2, \dots x_n$

if x_n is not picked then $opt(w, n) = opt(w, n-1)$

If x_n is picked by optimal,

then $opt(w, n) = opt(w - w_n, n-1) + P_n$

So, $opt(w, n) = \max \begin{cases} opt(w, n-1) \\ opt(w - w_n, n-1) + P_n \end{cases}$

where $w = 1$ to k

Date : 08/06/23

Divide-n-Conq, Greedy & DP
↓
Pardos & Cleinberg
back ch-exercise

Searching, sorting, asymptotic notation
recursion, Mathematical Induction
↓
common

Dnc, Gr, DP

atleast First 5, 6

Next Mon, Tue
Revision class

String matching

↳ Naive method
↳ Finite Automata

Given a text string T[0..n-1] and a pattern P[0..m-1]
find all occurrences

Text : a b a a b a a a c a a a b a a a c

Pattern : a a a f

S_0 S_1 S_0 S_1 S_2 S_0 S_1 S_2 S_3 S_0 S_1 S_2 S_3 f, backtrace S_0

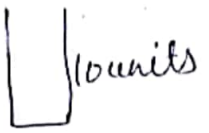
- O(n) time

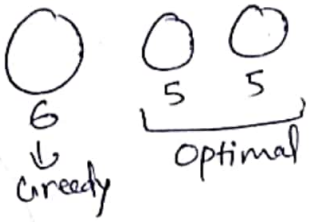
Text: a b a b a b a b f

Pattern: a b a b f

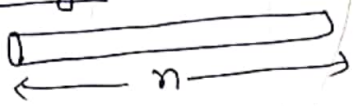
Date: 08/06/23 Aditya Pancholi sir

Wt.	P/unit	Price
10	80	800
5	60	300





Rod cutting



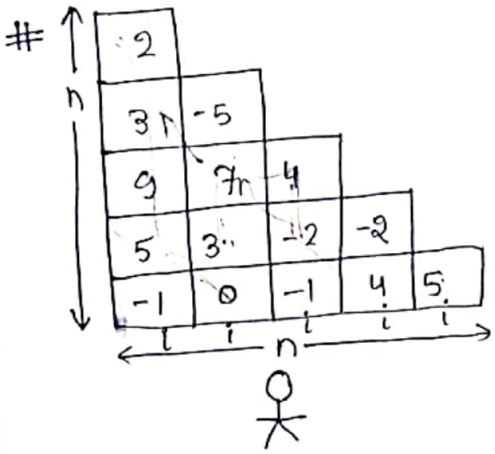
Find the optimal profit

Goal: what is the maxm. profit we can achieve

Note: you have the machine to cut the rod, hence it can be done free of cost.

Size	Profit
1	5 P_1
2	7 P_2
3	13
⋮	
n	100 P_n

Time = $O(n^2)$
Space = $O(n)$



goup (↑) or goup (left)

Goal: Collect the maximum points

P[i][j] = Value in the Pyr. at cells i,j

Pcol[i][j] = max profit that you can achieve from the pyramid rooted at (i,j)

Intermediate

Pcol[0][0]

$$Pcol[0][0] = \begin{cases} Pcol[0][0] + P[ob[1][0] \\ Pcol[0][0] + Pcol[1][1] \end{cases} \max$$

```
for (i = n-1 down to 0)
{   for (j = 0 to i)
        Calculate PCol[i][j]
}
```

→ max { P[i][j] + PCol[i+1][j] ; P[i][j] + PCol[i+1][j+1] }

Time: $O(n^2)$
Space: $O(n^2)$

DP mindset

look at what opt can do

↓

arrange / consider the input in some order

↓

define the goal { Try to see / define the intermediate results & write the recursive fun.

↓

Analyze the order of recursion

↓

Reverse order is the order factual calculation

↙

Fill the intermediate results (iterative fashion)

↙

Reach your goal

Date: 13/06/23

7.1 KMP

Fail [K] = $\gamma+1$

Let γ be the length of longest prefix of $P_1 \ldots P_{K-1}$ that matches suffix of $P_1 \ldots P_{K-1}$

Pattern P: Compute Fail[K] recursively

Given: Fail [1] ... Fail[K-1]

P: P_1 P_2 P_3 P_4 P_5 P_6 P_7 P_8 P_9 P_{10} P_{11} P_{12} P_{13}

a b a a b a b a a b a a b

Fail[1] = 0

$s'+1 = 1$ $P_{s'+1}$ with P_{K-1} (K=3)

0	1	1	2	2	3	4	3	4	5	6	7	5
Fail 1	2	3	4	5	6	7	8	9	10	11	12	13

P_1 with P_2

aba

To compute Fail[2] look at Fail[1]

$(t = 1 > 0 \;\&\&\; P_1 \neq P_2)$

$t = 0$

Fail [3] = 0+1 = 1

K=4

t = Fail[3-1]

$t = 1$

$1 > 0 \;\&\&\; P_1 \neq P_3$ F

t = Fail[1]

$t = 0$

Fail[4] = 0+1

K=5

t = Fail [5-1]

$t = 1$

$1 > 0 \;\&\&\; P_1 \neq P_4$ F

Fail[5] = 1+1 = 2