Introduction to Algorithms → corman

Fundmentals of computer algorithms

↳ Horwitz

→ sahni

* বই এর নাম
* research team আছে কিনা

google, microsoft (Algorithm)

৫ ২০ জন

↳ researcher দের প্রোফাইল,

undergrade থেকে ...

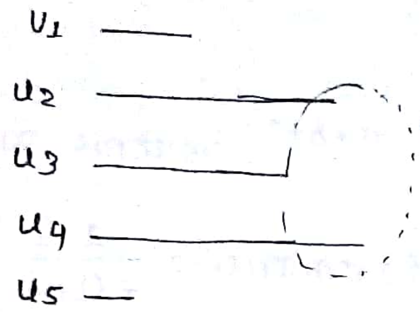
Scanned by CamScanner

University : create knowledge

spread "

School, college → spread knowledge

faster growing component → Asymptotic behaviour



When a bad algorithm run faster:

1) strategy

2) hardware, compiler

$c \rightarrow I \nearrow$ JVM

'Byte code'

Logic অনুযায়ী কোন algorithm better.

→ time

→ success

→

Asymptotic Behaviour: logic এর উপর focus, ছোটখাট বিষয়গুলো ignore করা যায়।

$a = b+c$

$d = a+e$ | steps / statement

$f = d+g$

3

matric (step no.)

$d = a+b+c$ matric 01

performance = f(LCE)

Alg1 = 3

Alg2 = 1

$\text{performance} = \dfrac{1}{f(Lc)}$

matric এর value বেশি হলে performance খারাপ।

Induction: গাণিতিক আরোহ

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

$$0 = 0$$

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

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

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

Assumption

i) Uniprocessor model

ii) All memory access are equally expensive.

iii) Primitive steps should be counted.

(complex step count করবো না)

$x = 5(a+b)+c$ → count করবো না। expand করে নিতে হবে

Insertion sort:

```
n        for(i=2; ---> n)
(n-1)        key = A[i]
              j = i-1;
           while (A[j] > key && j>0)
T-1          { A[j+1] = A[j];
T-1             j--;
              }
n-1          A[j+1] = key;
           }
```

(3)

A	2	5	7	
		j		

2	3	5	

$T = t_2 + t_3 + \dots t_n$

2	7	5	3

key = 7 check করে key = 5 ধরতে হবে।

$$c_1 n + c_2(n-1) + c_3 + c_4(T - \overset{n-1}{1}) + c_5(T - \overset{(n-1)}{1}) + c_6(n-1)$$

$$= c_7 n + c_8 T + c_9$$

for(i=2 → n) → loop statement 3 बार)

for(i=1 → 3)

i++;

i=i+1;

i=i+1;

i=i+1;

Merge sort

low mid ↓ high

a | 2 | 5 | a | 3 | 4 | 7 | 9 |

i j

b | 2 | 3 | 4 | 5 | 7 | 9 |

n

pseudocode (genetic)

① if(a[1/i] < a[mid/j]

b[n] = a[i];

i = i+1;

else

{ b[n] = a[j];

j = j+1; }

n = n+1; n

while(i<mid && j ≤ high) { ①

→ while(j ≤ high) → n/2

if (i> mid-1) ①

{

b[n] = a[j]; ①

j = j+1; ①

n = n+1; ①

}

if(j> high) → (n+6)

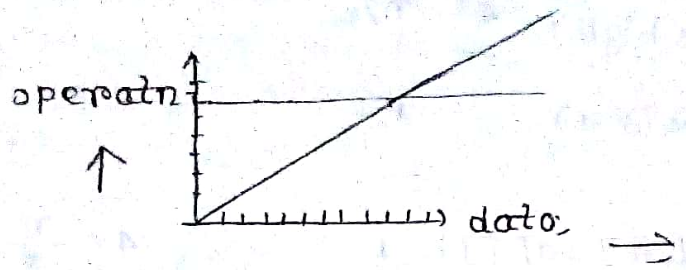
{

{

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

$\Rightarrow n+6$

$n+6$



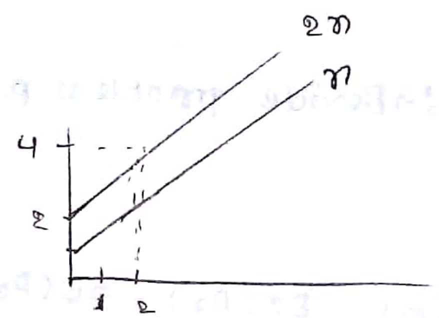
n vary করে, 6 করে না। তাই 6 ignore করব।

n linear (asytopic behaviour)

$n + n = 2n \rightarrow n$

$an+b$

$y=mx+c$



combine করার কাজ n/linear ভাবে করতে পারি।

```
while (i ≤ high)   ← n/2
{ if (i > mid-1)              1
    {
      b[n] = a[i];   1
      i = i + 1;        1
      n = n + 1;       1
    }
}
```

$4 \times \frac{n}{2}$

c n

constant

Devide and conquer: → Devide problem p into subproblems

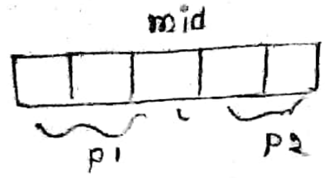
p_1, p_2, p_3

→ solve subproblems $DC(P_1)$, $DC(P_2)$..

→ combine solutions combine ($DC(P_1)$, $DC(P_2)$, - -)

$\boxed{2}$ একটা element হলে devide possible না। তাই

code এর শুরুতে condition দিয়ে দিতে হবে।



$(l < h)$ {

$m = \left\lfloor \frac{l+h}{2} \right\rfloor$

$P_1 = DC(l, m)$

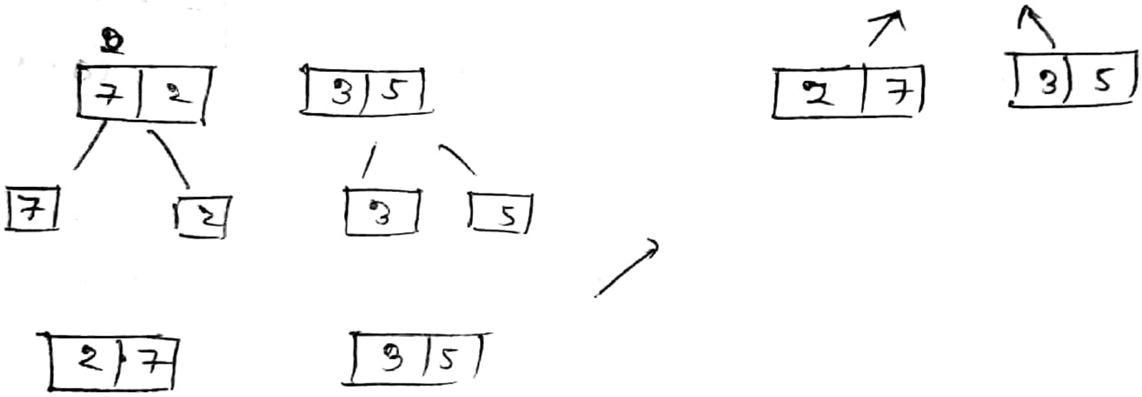
$P_2 = DC(m+1, h)$

combine (P_1, P_2)

}

7 2

2 7 3 5



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

* Devide & conquire

recursive এ করা যায়

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

$2T(n/2) + (n-1)$ $\quad i=1$

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

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

$= (n-1) + (n-2) + 4\{2T(\frac{n}{8}) + (\frac{n}{4} - 1)\}$

$= (n-1) + (n-2) + (n-4) + 8T(n/8)$ $\quad i=3$

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

$= (n-1) + (n-2) + \cdots + (n-2^{\log_2 n - 1} + nT(1)$ $\quad \Big|\ 2^i = n,\ i = \log_2 n$

$= (n-1) + (n-2) + \cdots (n - \frac{n}{2}) + 0$

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

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

$= n\log_2 n - cn$ $\quad$ [T বিপরীত eqn]

Scanned by CamScanner

Recurrance Relation:

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

merge

→ comparison

$= \cdots\cdots = n\log n$

Binary search: Devide and con..

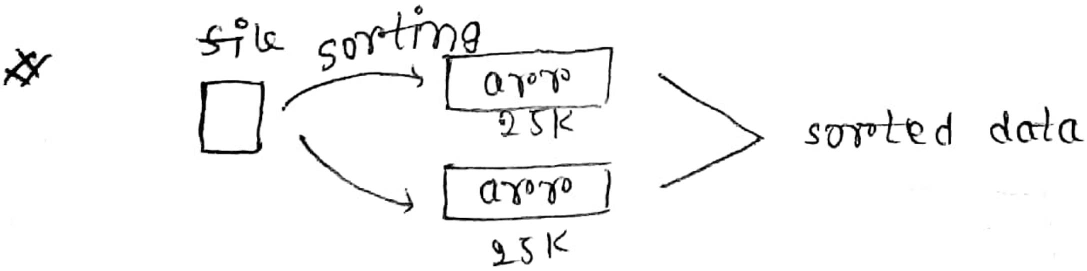
3	7	2		

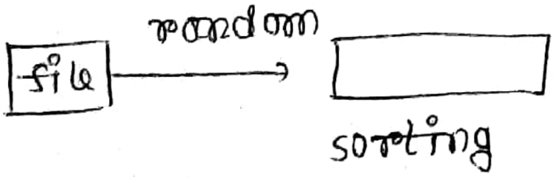
find MAX

```
MAX = -999
for (i = 1 → n){
    if (a[i] > max)
        MAX = a[i];
}
```

(n)

inser

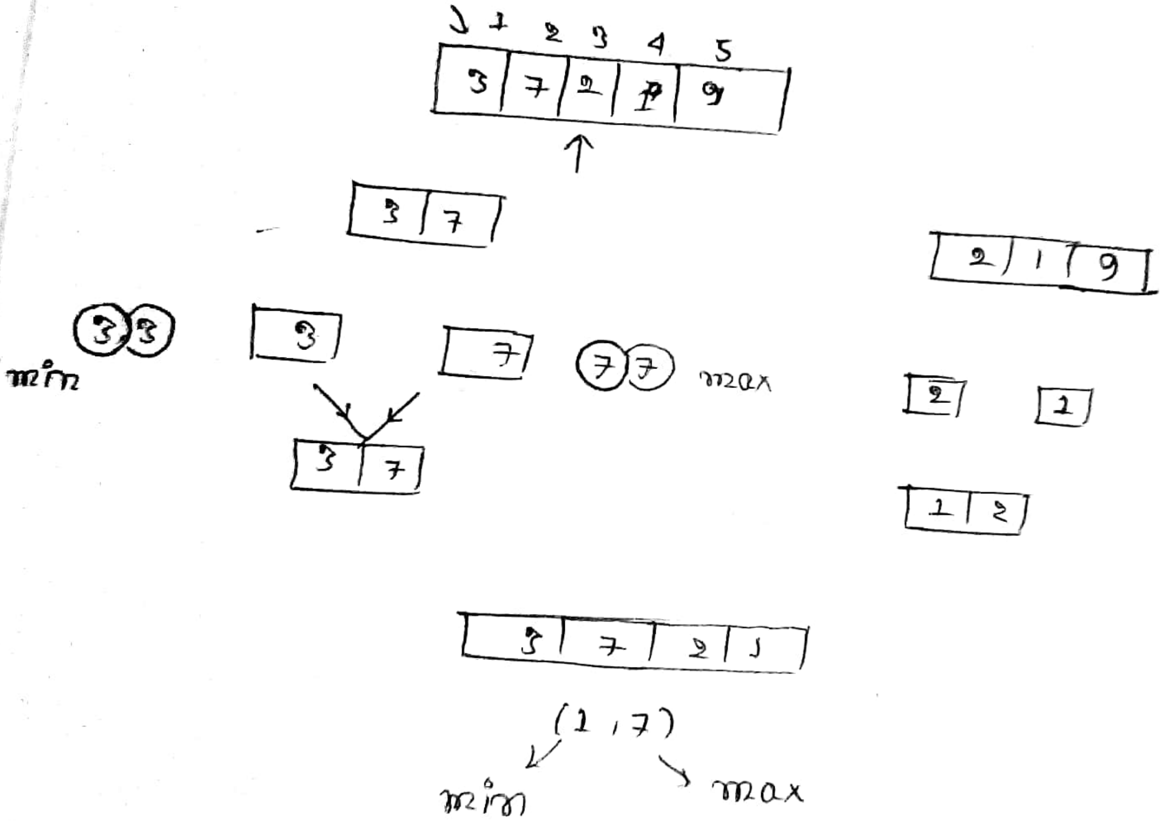




Merge sort	Online Insertion	Offline Insertion	Bubble sort

Merge sort :

$$\frac{t_1 + t_2 + \dots t_5}{5}$$



DCMM ( ) {

(when $n > 1$)

else

minL, maxL ← DCMM $(l, \lfloor \frac{l+high}{2} \rfloor)$

১ একটা element

minR, maxR ← DCMM $(\lfloor \frac{l+high}{2} \rfloor + 1, h)$

}

$\boxed{\text{if } (n \cancel{\geq 1} \ \&\& \ l = high) \quad MAX = MIN = A[l];}$

* ২টাতে তত্ত্ব +

2 টা comparison

(2 টা if statement

* n=1 এর জন্য

if (MinL < MinR)

MIN = MINL;

else MIN = MINR;

→ if (MAXL < MAXR)

MAX = MAXL;

else,

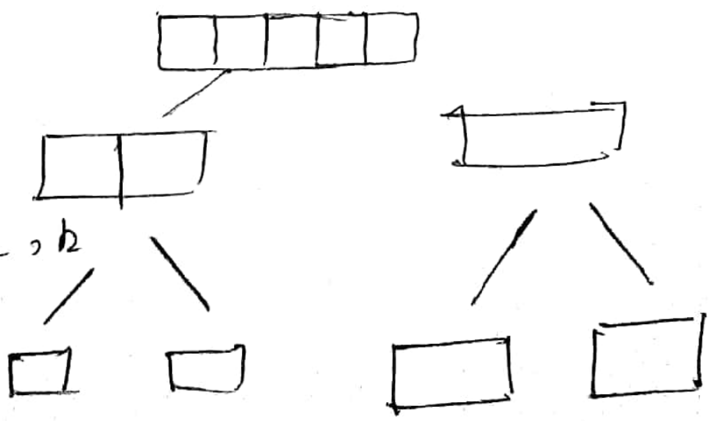
MAX = MAXR;

comparison

$T(n) = \begin{cases} 2T(\frac{n}{2}) + 2 & n > 1 \\ 0 & n = 1 \end{cases}$

~~tree~~

5 টা element



প্রত্যেক step এ l, h

MIN, MAX

MINL, MAXL

MINR, MAXR

DATA marr

2	4	6	8	1	3

1 3 4 6 8 9

2	5	7	11	13	15

5

DATA

1 marge → compar

2. ,, → একটার পিছনে

3. online

4. Offline

5. পুরাতন এনে merge data

$aT\left(\frac{n}{b}\right) + cn$

4, 6, 8 · ভাগে ভাগ করার দরকার নাই। 2×2, 2×3, 2×4

Master Theorem:

Recurrence relation থেকে O

$$\begin{array}{r} 115 \\ 23 \\ \hline 345 \\ 2300 \\ \hline 2645 \end{array}$$

$115 \times 23 = 2645$

$(100 + 10 + 5) \times 23 = 2300 +$

$23 +$

// $(100 + 15)(20 + 3)$

$= 2000 + 300 + 300 + 45$

$= 2645$

$$\begin{array}{c} 15 \\ \times \\ 13 \\ \hline \end{array}$$

: $n=2$

multiplication $4 = 2^2$

$n=3$ হলে $3^2 = 9$

$$\begin{array}{c} 3415 \\ 2523 \\ \hline \end{array} \longrightarrow (34\times10^2+15)\times(25\times10^2+23)$$

$= (34\times25)\times10^2\times10^2 + 34\cdot23\cdot10^2$

$+25\cdot15\cdot10^2 + 15\cdot23$

$= 34*25\cdot10^4 + (34*23 + 25*15)10^2$

$+15*23$

multiplication 4.

$n=2$ (2 digit)

original টা 4 digit

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

$=$?

Scanned by CamScanner

10110 → 2 বিট্টু গুনে left shift.

$$a_1 a_0 \quad b_1 b_0 = (a_1 \times 10^{n/2} + a_0)(b_1 \times 10^{n/2} + b_0)$$

$$= a_1 b_1 10^{n} + a_1 b_0 10^{n/2} + a_0 b_1 10^{n/2} + a_0 b_0$$

$$= \underbrace{a_1 b_1}_{c_2} 10^{n} + \underbrace{(a_1 b_0 + a_0 b_1)}_{c_1} 10^{n/2} + \underbrace{a_0 b_0}_{c_0}$$

$$c_1 = (a_0 + a_1)(b_0 + b_1) - (c_2 + c_0)$$

$$= a_0 b_0 + a_1 b_0 + \underbrace{\boxed{a_0 b_1 + a_1 b_1}}_{c_1} - a_1 b_1 - a_0 b_0$$

$3T(\frac{n}{2}) + O(n)$ | $Res = c_2 10^{n} + c_1 10^{n/2} + c_0 \times 10^{0}$

$c_0 = a_0 b_0$ 0

Master theorem:

$$T(n) = \begin{cases} c & n=1 \\ aT(\frac{n}{b}) + cn & n>1 \end{cases}$$

$$T(n) = a^2 T(\frac{n}{b^2}) + \frac{a}{b} cn + cn$$

$$= \cdots \quad \cdots$$

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

$$= a^k c + cn(\frac{a^{k-1}}{b^{k-1}} + \cdots \frac{a}{b} + 1)$$

| $b^k = n$

$\exists\, k = \log_b n$

$$= a^k c \frac{n}{b^k} + cn(\cdots)$$

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

if $(a == b)$

$$T(n) = cn(k+1)$$

| $n = b^k$

$k = \log_b n$

$$= cn(\log_b n + 1)$$

$$= \Theta(n \log_b n)$$

$n =$

if $(a < b)$

$$= cn \left\{ \frac{\left(\frac{a}{b}\right)^{k+1} - 1}{\left(\frac{a}{b}\right) - 1} \right\}$$

$$= cn \left\{ \frac{1 - (a/b)^{k+1}}{1 - \left(\frac{a}{b}\right)} < \frac{1}{1 - (a/b)} \right.$$

$$= cn\,\Theta(1)$$

$$= \Theta(n)$$

if $(a > b)$

$$cn \left\{ \frac{(a/b)^{k+1} - 1}{(a/b) - 1} \right\}$$

$$= cn\,\Theta\left(\frac{a}{b}\right)^{k}$$

$$= cn\,\Theta\left(\frac{a^{\log_b n}}{b^{\log_b n}}\right)$$

$$= cn\,\Theta\left(\frac{n^{\log_b a}}{n}\right)$$

$$= \Theta\left(cn \cdot \frac{n^{\log_b a}}{n}\right)$$

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

$$T(n) = \begin{cases} aT\left(\frac{n}{b}\right) + f(n) & n>1 \\ c & n=1 \end{cases} \qquad f(n) \quad O(n^d)$$

$$T(n) = \begin{cases} \Theta(n^d \log_b n) & a = b^d \\ \Theta(n^d) & a < b^d \\ \Theta(n^{\log_b a}) & a > b^d \end{cases}$$

$a \geq 1$, $b \geq 2$, $c > 0$, $d \geq 0$

= master theorem. . .

Binary search এর মত

* $T(n) = T\left(\frac{n}{2}\right) + \frac{1}{2}n^2 + n$

$b = 2$ $\quad d = 2$ $\quad \Theta(n^2)$ $\quad f(n) = \frac{1}{2}n^2 + n$

$a = 1$

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

* $T(n) = 3T\left(\frac{n}{2}\right) + O(n)$

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

a	b	d		
3	2	1	$O(n^{\log_2 3})$	$= O(n^{1.\cdots})$
4	2	1	$O(n^{\log_2 4})$	$= O(n^2)$

Scanned by CamScanner

2nd MAX = 7 MAX = 9

1	7	2	9	4	3

DCMAX (A, l, h) = 9

DCMAX2 (

DCMM $T(n) = 2T\left(\frac{n}{2}\right) + 2$ | $2 \times T(n) = 4T\left(\frac{n}{2}\right) + 4$

a	b	d
4	2	0

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

1\. $\Theta(n^{\log_2 4})$ $\Theta(n^2)$ ২ বার কল করলে { DCMAX, DCMAX2 }

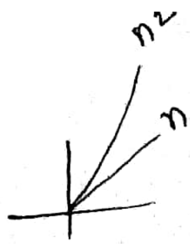
2\. If, else এর সংখ্যা বাড়ালে

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

a	b	d
2	2	0

| $\Theta(n^{\log_b a})$

$\Theta(n)$



7	5	6	2	1	18	20

Median value?

⇒ MID

even হলে,

২টা

Bubble → n^2

merge $n \log n$

Insertion → n^2

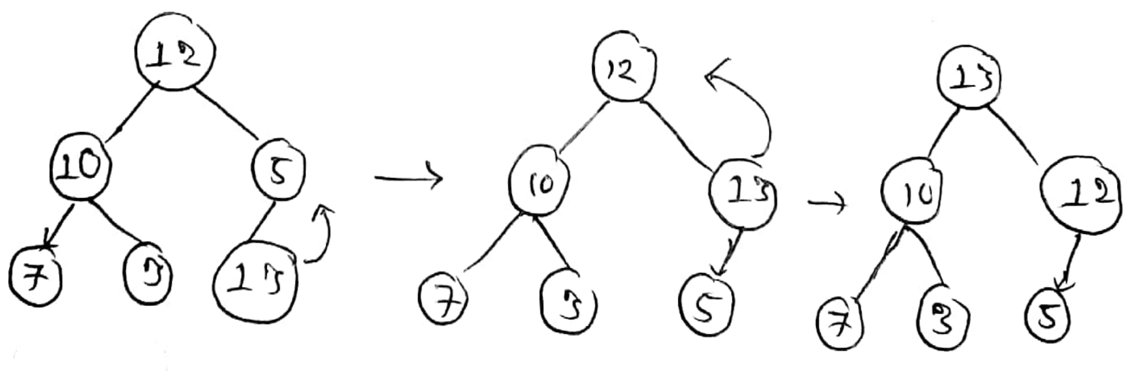
$\boxed{\text{Better solution:}}$ try

* Heap Sort:

→ Complete Binary tree

Binary Heap: Deletion:

12, 10, 5, 7, 3, 13



* Data কিভাবে রাখা হয়? array / linked list / tree / graph

tree আলাদা data structure . . .

1'1'2

structural Representation.

1. Grammer

2. Regular Language / expression

1^* (set of unlimited 1)

1, 11, 111 (যেকোনো সংখ্যক 1 acceptable)

5A. Iteration/ Iterative method.

Master method.

Priority Queue:

F rare

FIFO

Q

10	5	12	13	2
5	7	2	4	9

→ priority

priority value...

Optimization problem:

MAX priority

dequeue (13, 4)

print (4 deque

1) Unordered

2) Ordered

4. S/W verifying system.

Elements of a finite automata state diagram.

O → circle → node → state

→ = edge → one node ~~from~~ another

$\left.\begin{array}{l} \xrightarrow{0} \\ \xrightarrow{1} \\ \xrightarrow{a} \end{array}\right\}$ → edge → ~~link for one node from another.~~

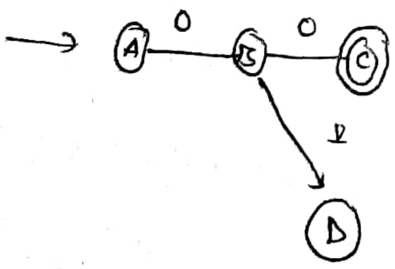
has input ~~that~~ shift one node to another.

state ~~dige~~ diagram

(A) = A is node nam

→O = starting node

◎ = ending node



00 → accepted string

01 →

parent - child:

n (index)

2n   2n+1

Insert : Insert as the last node of tree.

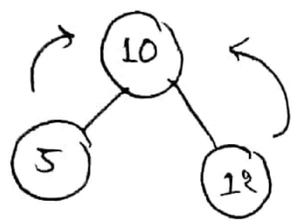
int par = $\frac{pos}{2}$,

Heapify: MAX

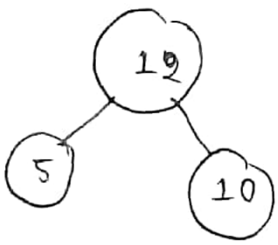
if (H[pos] > H[$\frac{pos}{2}$] .

if (H [pos] > H[par])

swap (pos, parent);



swap



* Insert as the last node of tree

* while (pos != root) → index

{ int ~~n~~ par = $\frac{pos}{2}$;

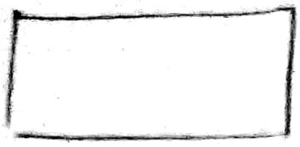
if (H[pos] > H[par])

swap(pos, par)

else break

10 , 5

UA



12 ,5 ,10

DA (Disending)

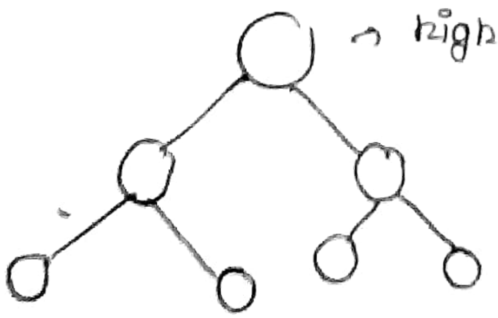
2, 5, 10, 12, 13

Heap: 1) Complete Binary Tree

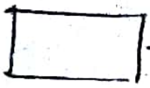
2) Heap property

→ parent is greater (less) then child in max

Heap

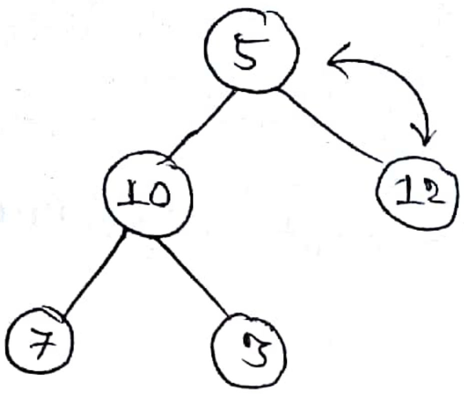


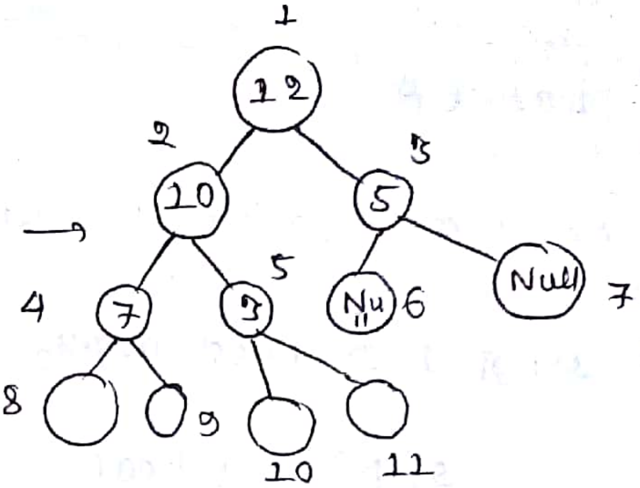
MAX

 → max বড় হবে।

Dequeue max,

Root থেকে delete. [Root থেকে value, আর node delete হবে নিচ থেকে]



→ 

→ Remove/ Return root value

→ Reduce root value with last node's value.

→ Heapify (down) until leaf node is found.

Strassen's Matrix multiplication...

এ

Binary heap বাদে

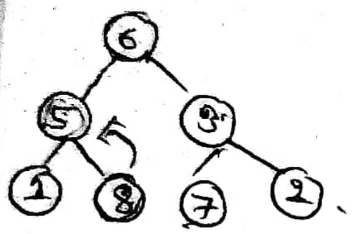
6, 5, 3, 1, 8, 7, 2

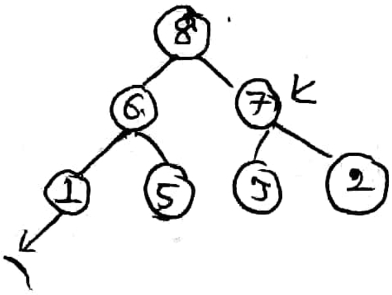
$\frac{2-1}{2} = 0.5$

8, 6, 7, 1, 5, 3, 2

0 1 2 4

$\frac{3}{2} = 1.5$





Task: Lab 6A

Implement max priority queue operation (enqueue and dequeue) using 1. ordered array

2. binary heap

Take random size, find the time difference for a max priority queue's enqueue operation and deque operation for both way.

Deterministic → Tractable

Nondeterministic → Intractable

10	5	7	20

$m = \left(\frac{l+h}{2}\right)$ / rand (1, 4)

(nondeterministic)

A

5	13	2	14	20	9	8

Linear → n

BS → ($\log_2 n$ + sorting

↳ n^2 / $n\log n$ ✓

10000000

7

1	2	3	4	5
1		1		1

5 আছে

3

1 যেগুলো আছে সেগুলোর index এ value 01..

⋮

random index checking

Quick sort

A 10, ~~20~~ 5, 50, | 30 | 40, 30, 90

Pivot element

wall

0	1	2	3	4	5	6
10	5	50	30	40	~~30~~	90
↑		3			50	↑
						SS

```
pivot = A[0];

while (ss > sb)

while (A[sb] <= pivot)

    sb = sb+1;

while (A[ss] >= pivot)

    ss = ss-1;

if (ss>sb)

    swap(A,[sb], ss);

        sb++, ss--;

swap(A, 0, ss)
```

QS( )

{ x = A[l];

Index = partition(A, l, r x);

QS(A, l, pivot−1);

QS(A, pivit +1, r);

6E

QS( )

x ← select a pivot element,

partition based on k;

QS(partition1);

QS(partition2);

pivot

1	3	2	5	7	20	9

sorted array তে quicksort এর complexity n^2 (bubble sort এর মত) unnecessary comparison কিভাবে কমানো যায়?

। pivot point কে select করতে হবে randomly

QS()

x ← select a pivot element;

partition based on random select

QS (partition).

QS (partition);

}

Sorted data , QS এর জন্য worst case

কাজ :

n^2

1. worst case এ complexity কত?

2. random case add করলে complexity কত?

random নাম্বার generate

i) Las vegas

ii) Monte carlo

$$\begin{vmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{vmatrix}$$

> 0 left

< 0 right

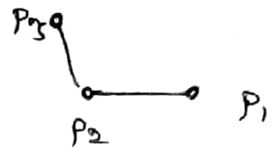
$= 0$ colinear

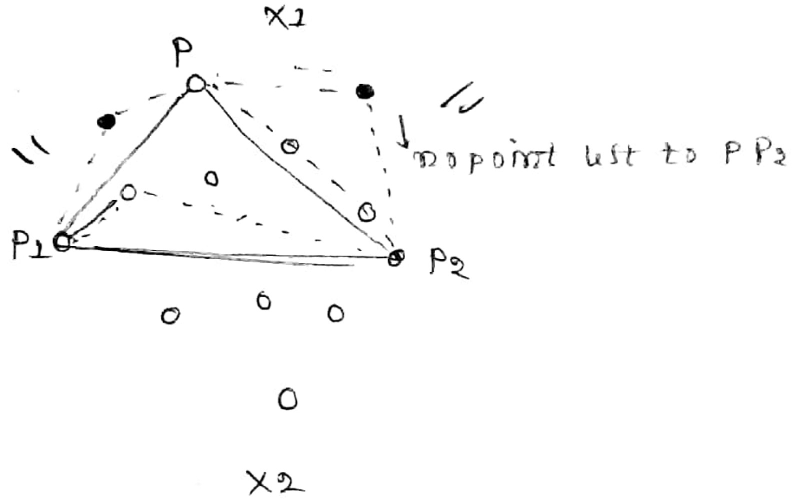
একটা point কোন position এ আছে?

$P_1 (x_1, y_1)$

$P_2 (x_2, y_2)$ $\quad \angle P_1 P_2 q$?

$q (x_3, y_3)$





compute Hull (X, P1, P2)

for each point p in X $\quad$ | O (number of points)

find area of triangle $P_1 P P_2$

select the p with maximum area

X_1 = points to the left of $P_1 P$ $\longrightarrow O(n)$

X_2 = points to the ~~left~~ left of $P P_2$ $\longrightarrow O(n)$

CH1 = compute Hull (X_1, P_1, P); T_{n_1} $\quad \boxed{\text{return } \{CH1(), CH2\}}$

CH2 = " (X_2, P, P_2); T_{n_2}

worst

$n_1 < n$
$n_2 < n$

Quicksort

$T(n_1) + T(n_2) + O(n)$

$T(n_1)+T(n_2)+O(n)$

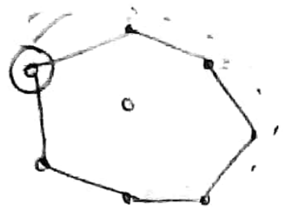
$\frac{n}{2}$ হলে $O(n\log n)$

$\boxed{\text{substitution method:}}$

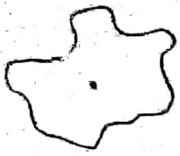
→ compare

n^3

triangle এ গুলা



- ☐ linear time a median finding.
- ☐ QS → randomis () : worstcase এ কাজে দিবে।



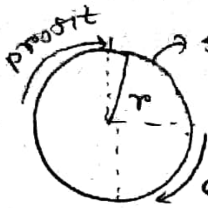
concave shape

↙

কে convex এ

transform

করতে হবে।

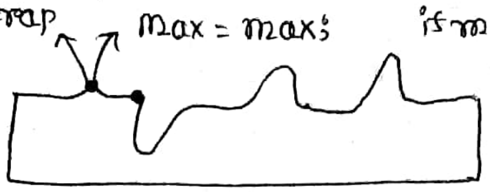


→ surface এর যেকোনটা point profit

error কমানো

convex shape

convex optimisation . .

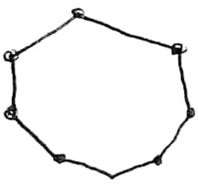


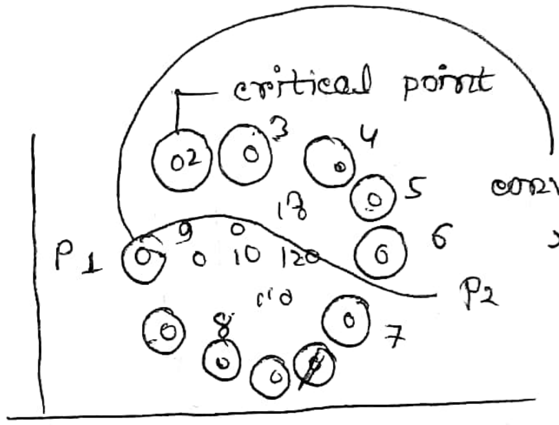
max = max;

if max < . . .

no change

Convex Hull





convex Hull

খুঁজতে হবে।

$P_1 \rightarrow$ lowest x

$P_2 \rightarrow$ highest x;

lowest excordinate এর কাছে.

highest "

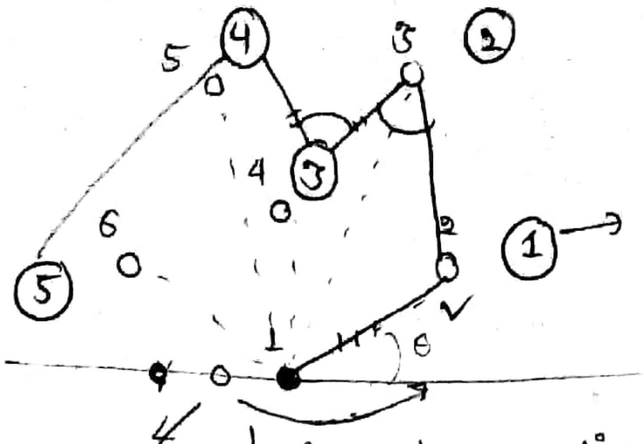
compute
~~convex~~ hull (X_1, P_1, P_2);

compute
~~convex~~ hull (X_2, P_1, P_2);

[surface এর pointগুলো
থাকে]

Scanned by CamScanner

Graham Scan ~~for~~ convex Hull



P_2 point convex এ থাকবে কিনা সেটা বোঝা যাবে left/right turn থেকে

1 → ~~P1~~ অবশ্যই convex hull এ থাকবে।

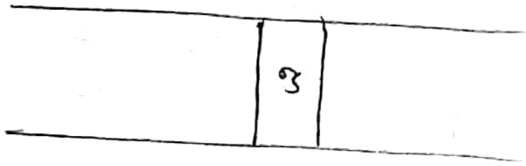
angle measure করে sorting

↳ lowest y cordinate

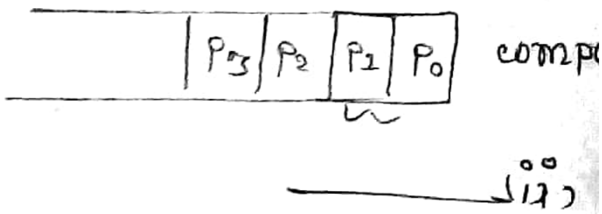
left one select

থেকে ~~অ~~ anticlockwise

sort (P_i); $i=1 \longrightarrow n$

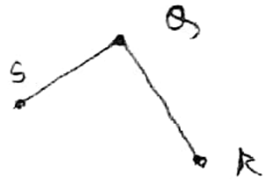


তিনটা নিয়ে compare



i=3

যদি না লাগে তাহলে pop করবো।



P= sort (P_i); $i=1 \longrightarrow n$ $n\log n$

push (P_0);
push (P_1);   O(1)
push (P_2);

$i=3$

while ($i<5$) ⟶ O(n)

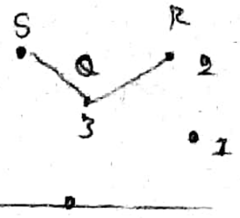
Q = pop stack (top); 1

R = stack (top-1); 1

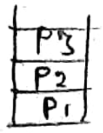
S: P[i] [i নং array থেকে] 1

~~if (i != 5)~~

if (det (R, Q, S) > 0) → right turn হলে pop

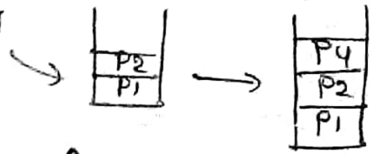


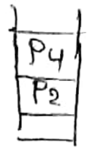

```
        pop ();   ।
else
    push (s);    ।
```

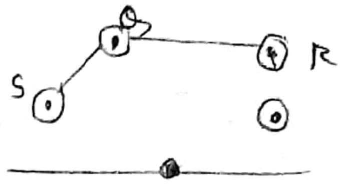



```
if ( i == 5)
        s = P0 ;    ।
}
```

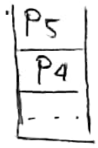
pop করলে

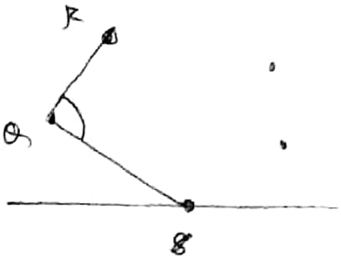




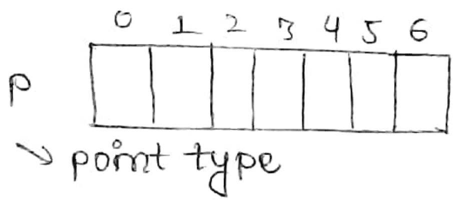


☐ sorting complexity এর উপর depend করবে।






```
struct point {
    int x;
    int y;
}
```



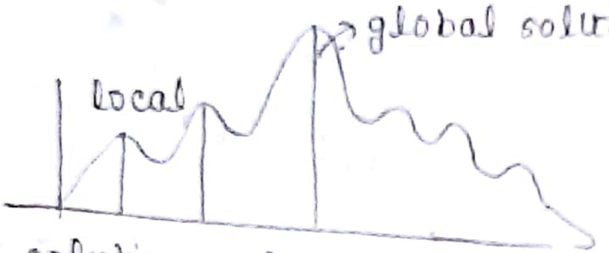
R.x
R.y

" convex hull → shahni "

Greedy choice : A global optimum can be arrived at by selecting a local optimum.

Optimal substructure : An optimal solution continuous an optimal solution to subproblem

Feasible Solutions



global solution (local solution গুলোর মধ্যে optimum)

সবচেয়ে ভালো solution প্রতি approach এ কাজ করে।

8 → 6 select । যেকোনো 6টা select করলে local optimal তবে 6টার মার্কস যদি সর্বোচ্চ হবে, তাহলে global

substructure : sub problem এর solution.

Greedy approach : n solve করলে তবে solution পাওয়া যায়।

Sequentially.

constraints: 8 → 6

→ 3 from each

যতগুলো constraint satisfy করে তারা feasible solⁿ

Encoding: এক form থেকে আরেক form এ

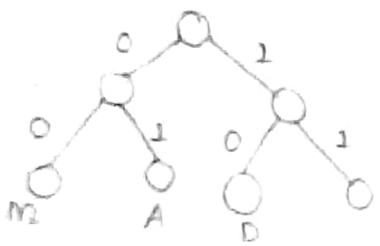
MADAM   3 bit এ
binary তে নিলে 2 bit লাগবে।

0001100100

| M → 00   MADAM লিখতে
| A → 01   10 bit লাগবে।
| D → 10

010010 → 6

constant length

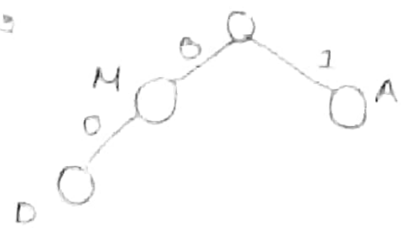


M → 0

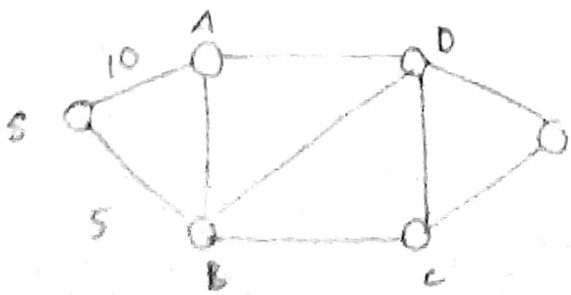
A → 1

D → 00

Optimum:



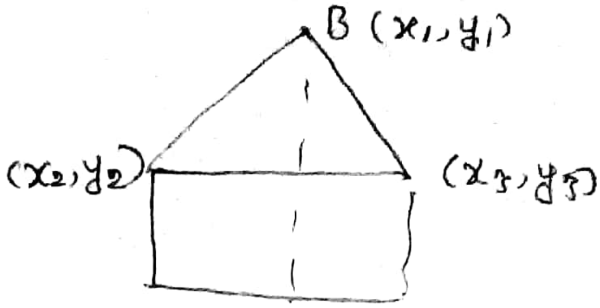
Huffman coding: Dijkstra:



Scanned by CamScanner

Area of a triangle

$\frac{1}{2}(x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2))$



no of points

2D point (x,y) coordinates.

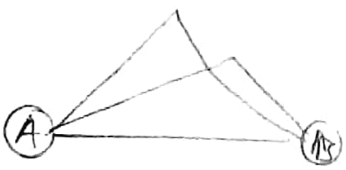
surface

$\boxed{8B}$

CT :

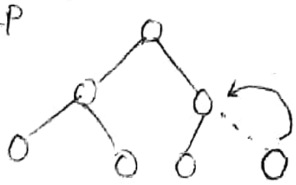
1. sorted

2.



3. sorting এর উপর depende

4. binary heap



logp

5. Ordered Array

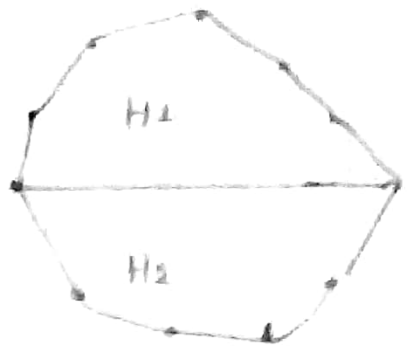
10 20 30 40 $\boxed{25}$

nlogn

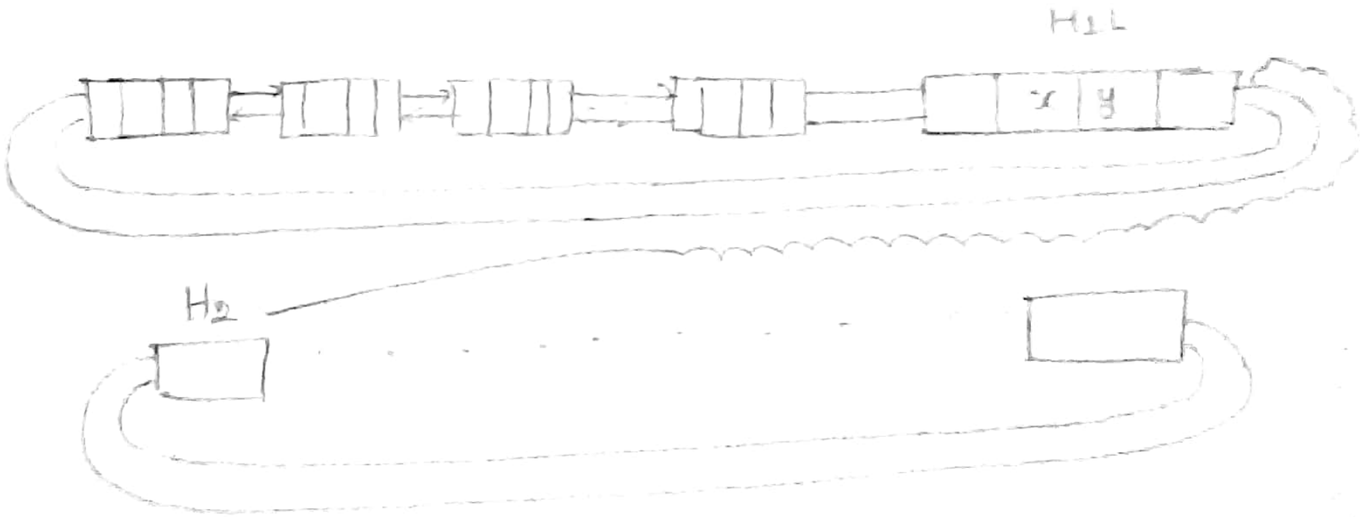
Unorder Array (1)

$H_1 = S$

H_2



combine



$H1L \to NN = H2$

$H2L \to PN = H1L$

$H2L \to NN = H1$

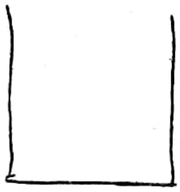
$H1B \to PN = H2L$

$H1L = H1 \to PN$

$H2L = H2 \to PN$

0/1 knapsack

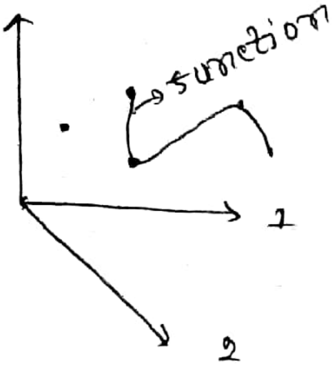
i	2	2	3	4	5
wi	24	70	10	7	
pi	12	9	9	5	



xi	1	0	0	1	

আগে 0 থাকবে

WB = 31 kg (capacity)



$$\sum_{i=1}^{n} w_i x_i \le WB$$

$$f = \sum_{i=1}^{n} p_i x_i$$

→ nlogn

RW = WB;

→ sort (pi); for (i=1; i ≤ n; i++) { n

~~RW ≤ WB~~

if (W[i] ≤ RW)

{

x[i] = 1;

RW = RW − W[i];

{

BF approach এ

$O(2^n)$

}

// profit

for (i=1; i ≤ n; i++)

profit = p[i] x[i];

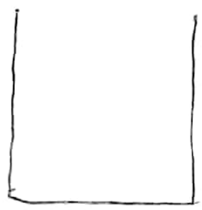
O(n) + O(nlogn)

weight এর সাপেক্ষে Greedy.

$RW = W;$
sort (wi) → ascending

```
for (i=1; i<n; i++)
{
    if (w[i] ≤ RW)
    {
        x[i] = 1;
        RW = RW - w[i];
    }
}
```

~~A=2~~



$W = 25$

Greedy WRT → Profit → 12

Greedy WRT → weight 24

But soln → 28 [Local minimum problem]

Fractional knapsack (একটা object কে ভাঙ্গা নেওয়া যাবে।

$x_i = [0, 1]$

$x = 0$ or 1 →

(4) Huffman tree :

1) Unique letter বের করে নিতে হবে।

R A J S H A H I ⟶ 8×3 ⟶ 24 bits

1 1 1 1 2 1

unique ⟶ 6(n)

R J S I A H
1 1 1 1 2 2

PREfix code :

R A J S H I

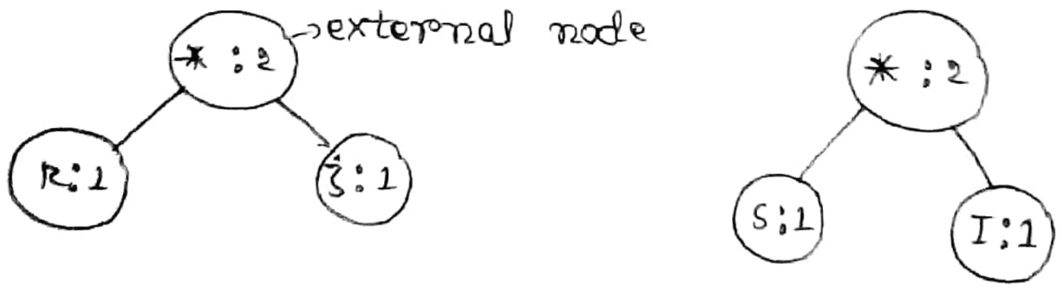
110 0 001 010 1 1

A H H
0 1 1

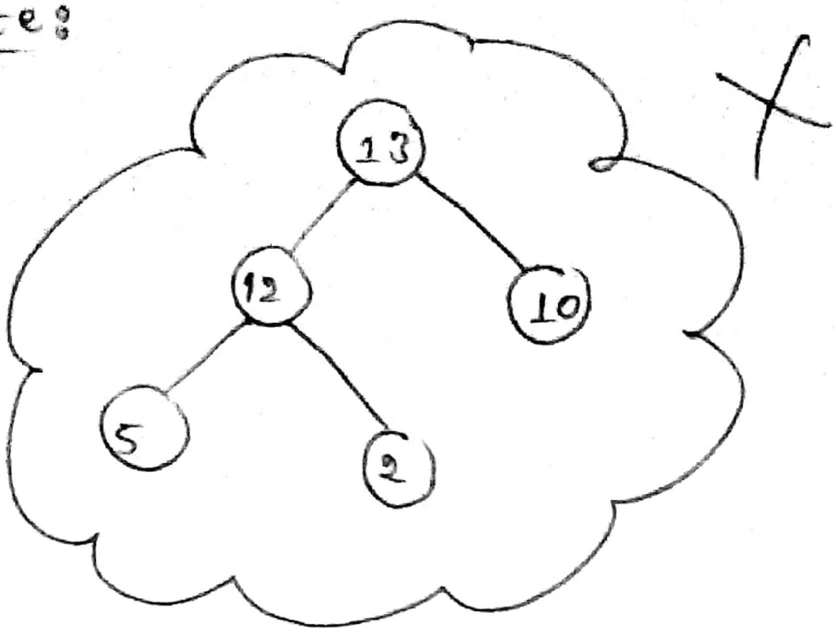
distinguish করা যাবে না।

২টা minimum frequency নিব। তারপর যোগ করব।

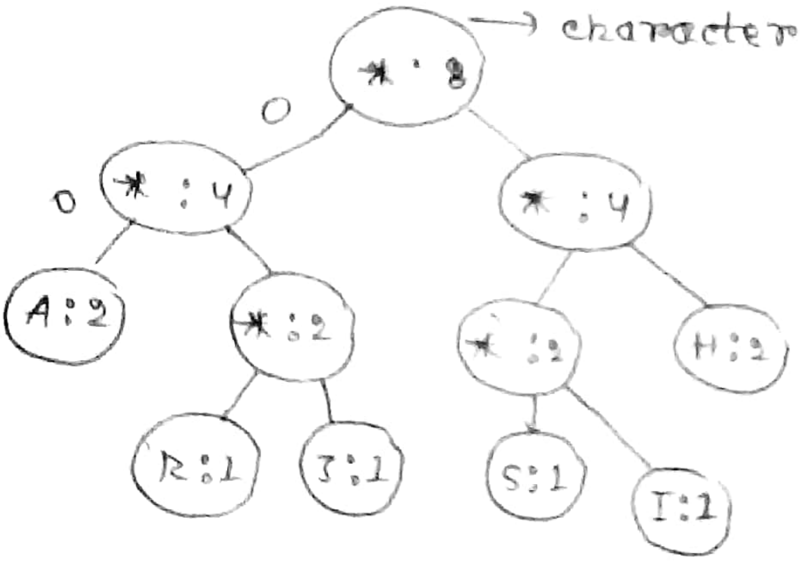
edge আমরা কি দ্বারা represent করছি।



Delete:



RASH



→ character এর value

A → 00

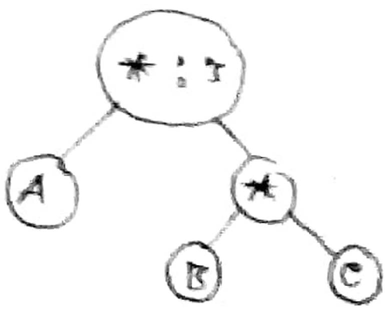
R → 010

J → 011

S → 100

I → 101

H → 11



P.W.

✱ Lab:

Take no. of objects n from users.

→ Find a global optimal solution for 0/1 knapsack problem, using bruteforce technique by generating 2^n candidate solution selecting approach

Apply greedy approach to achieve same solution

n	Bruteforce	Greedy
	Time	

str length
$n = |A|$

	1	2	3	4	5	6	
	R	A	J	S	H	I	*
Alphabets	1	2	1	1	2	1	2

for (i=1 → (n-1))

O(1) NN = create_new();

NN→L = Find-min(); queue এর জন্য Dequeue

NN→R = Find_min();

প্রথমটাকে খুঁজে বের করে delete করে দিতে হবে।

NN→f = (NN→L)→f + (NN→R)→f

NN→letter = '*';

}

NN এর leter assign করলাম।

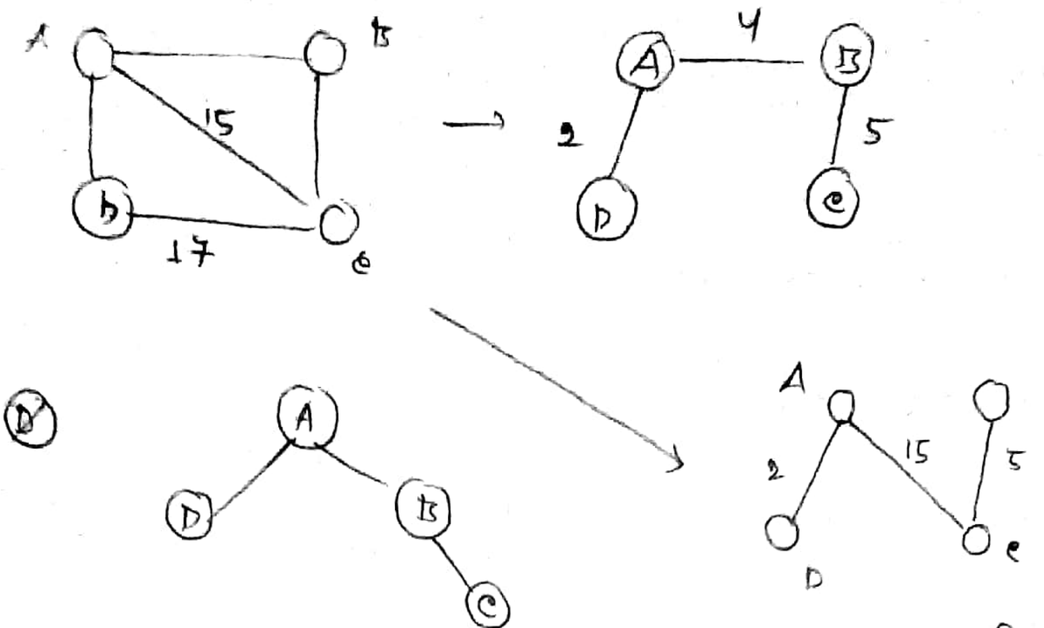
enqueue(NN); → নতুনটা push করব।

complexity = nlogn..

Greedy তে subproblem solve করে তারপর আর সেই object কে নিয়ে কোনো চিন্তা করে না।

greedy sequentially sol^n বের করে।

minimum Spanning Tree



cost বেশি হবে।

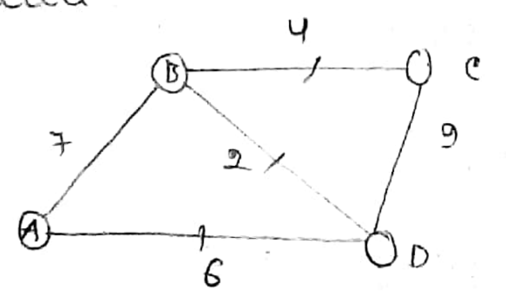
10 A

Application of mst

1) Prim's Algorithm

2) Kruskal's Algorithm

undirected



Edge

BD → 2
BC – 4
AD → 6
AB → 7

| tree

× CD → 9 (loop /graph create হবে)

→ store all edges into PQ (Priority Queue)

for (i = 1 ... |E|)

→ { Extract edge (V_1, V_2) from PQ

// V_1 → B

// V_2 → D

→ IF V_1, V_2 donot create any loop

MST = MST ∪ (V_1, V_2)

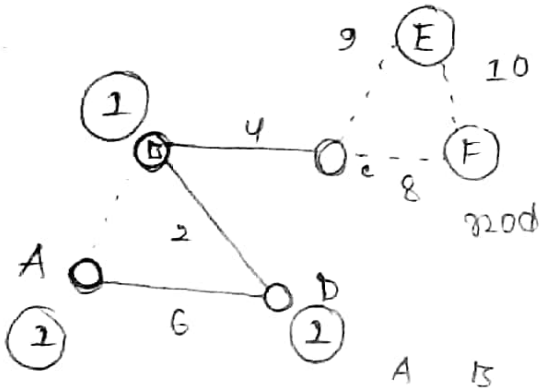
}

Edge এর set

{ AB
BC
BD
CD
AD }

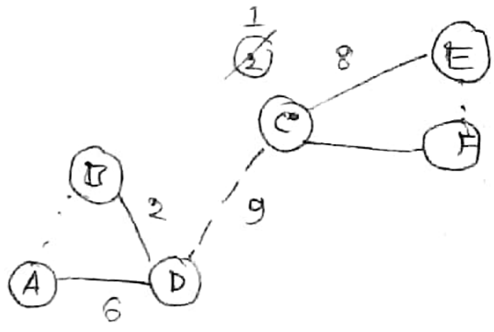
MST { first এ থাকবে

}



node গুলো visited...

A B
V_1 V_2



	A	B	C	D	
	1	2	3	4	5
mark	1	1		1	

if (mark [V_1] != mark [V_2])

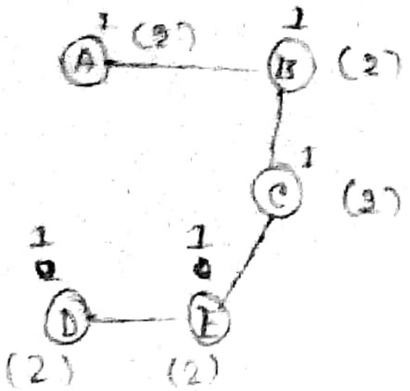
printf ("Take");

else

printf ("Discard");

(mark [V_1] == 1 && mark [V_2] == 1)

A — B



Take a string Bangladesh

count frequency of letters B A

Encode using Huffman Tree → B A

01

Show Decrypted message

→ Show Decrypted message from encrypted version

→ Find number of bits for constant length &

variable length encoding . .

bangladesh →

i:

baad

[0] →

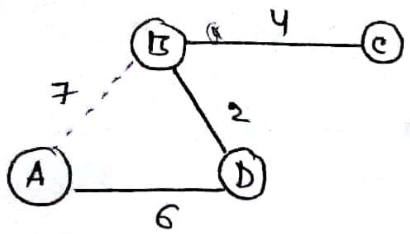
[1] = b

[i] = a

b a a d

10 1 6

b[0] = b



1. store all edges into PQ (Priority Queue)

2. for ($i = 1 \rightarrow |E|$) // $E \rightarrow$ no. of Edges

Extract edge (v_1, v_2) from PQ // for 1st case

$B \rightarrow v_1$

$D \rightarrow v_2$

if v_1, v_2 don't create any loop

$MST = MST \cup (v_1, v_2)$

The nodes that have been visited should be marked as

mark

B	D	A	C
1	1	1	
1	2	3	4

if (mark [v_1] != mark [v_2])

$\rightarrow$ take

else

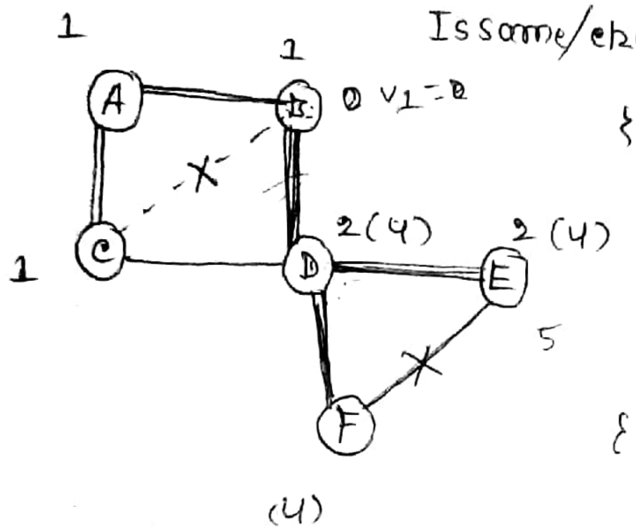
Discard

Krus Disjoint set :

(a, b) = get min edge; (dequeue)

if (a & b is not in MST)
or a-b not forming cycle
MST = MST ∪ (a, b);

0	1	2	3	4	5
A	B	C	D	E	F
1	1	1	2	2	2
			4	4	4



Issame/check (v1, v2)
{ if (mark [v1] != Mark [v2])
//take(); return false;
else
discard. // return true;
}

O(n²)

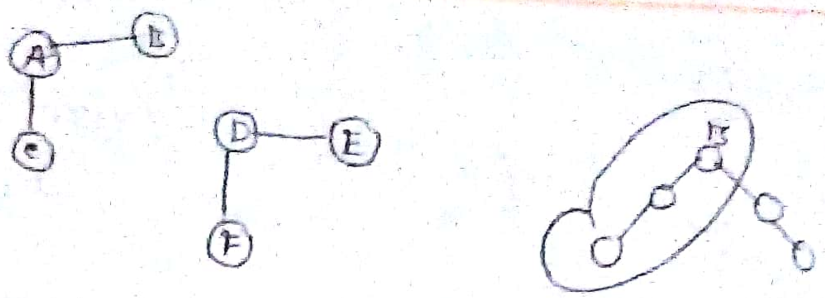
Union (v1, v2) {

if (!IsSame (v1, v2)){

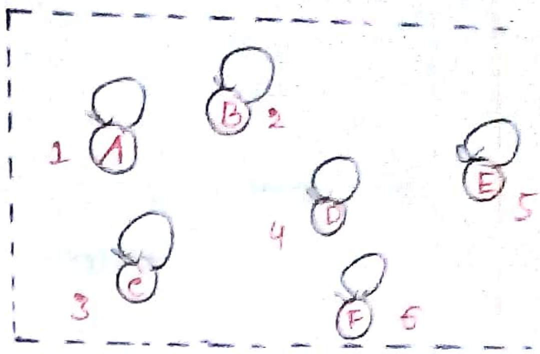
mark[v1] = mark [v2];
for (i = 1 → |V|) ⟶ O(n)
if (mark [i] == mark[v1])
mark [i] = mark [v2];

~~B/D~~ |V| → O(n)
|E| → O(n)

যার parent সে নিজেই হবে root



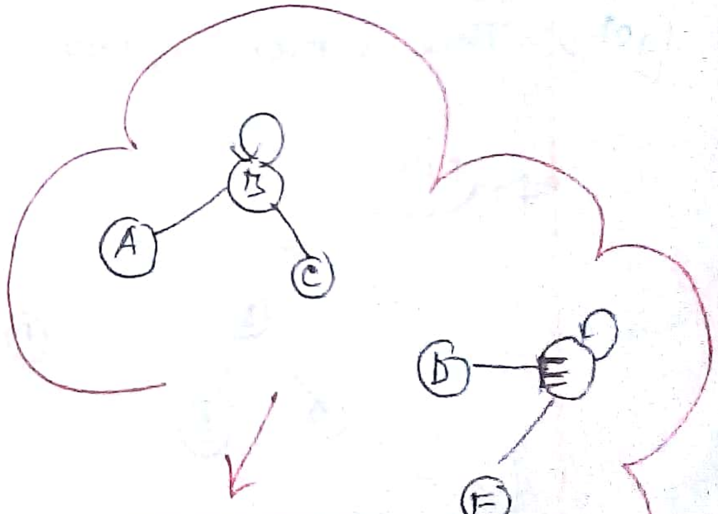
	A	B	C	D	E	F
Mark par	1	2	3	4	5	6



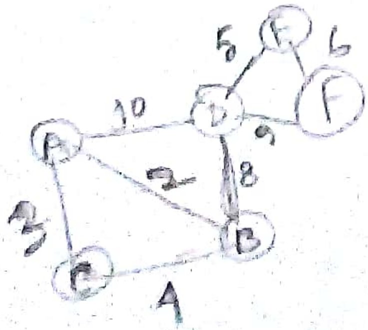
2	2	3	4	5	6

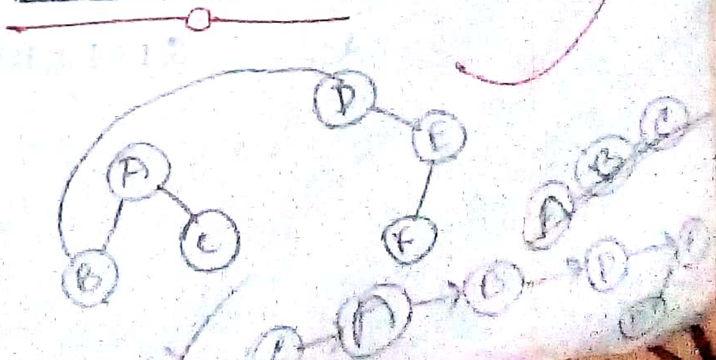
```
root (x) {
    while (par[x] != x) {
        x = par[x];
    } return x;
}
```

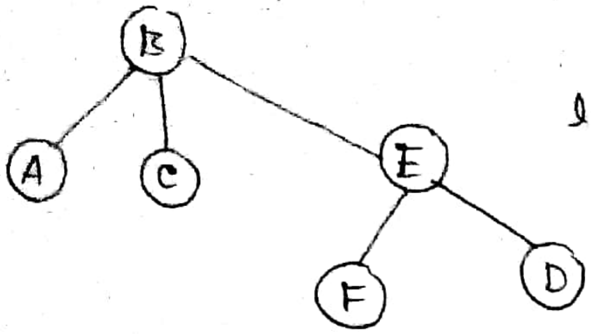
2	2	2	4	5	6



2	2	2	5	5	5

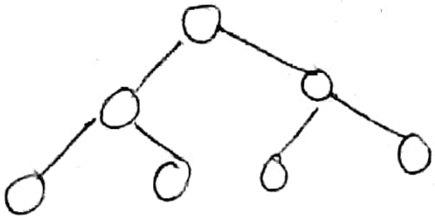






$\log_2 n$

balance tree (log n)



if (root (v_1) = root (v_2))

return false;

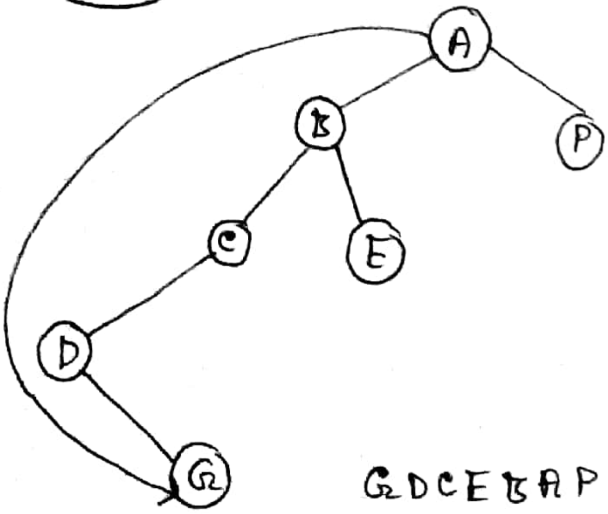
else

return true.

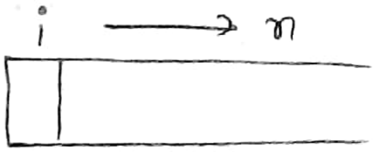
greedy aproach :.

Tree vertex spliting [sahani]

DFS



GDCEBAP

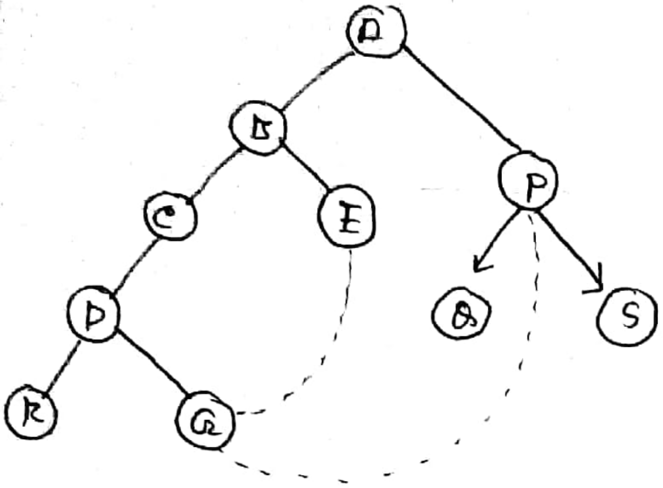


insert

traverse

delete

BFS



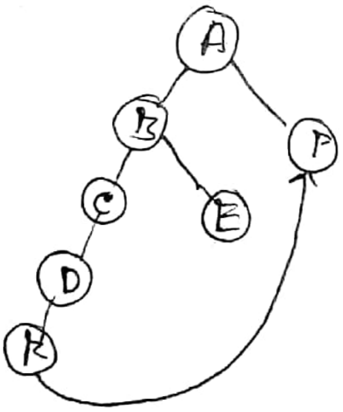
ABPCEDRG

DFS (নামার সময় traverse)

A B C D R G E P

BFS

A → B → P → C → E → Q → S → D → R → G



A → B → C → D → R → P → E

① | B | P |
A

② | P | B |
A

③ | P | E | C |
A B

④ | P | E | D |
A B C

⑤ | P | E | R |
A B C D

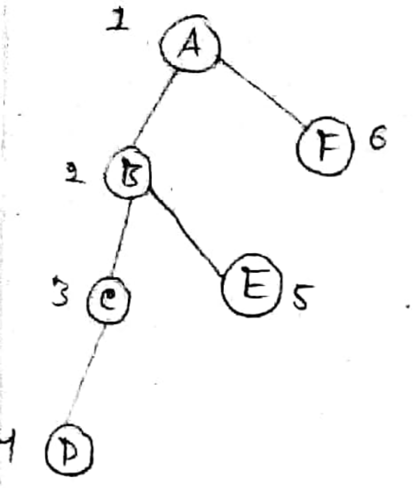
⑥ | P | E | P |
A B C D R

P নিতে পারব

⑦ | P | E |
A B C D R

⑧ | P |
A B C D R E P

* Adjacent গুলো push...



Adj

	1	2	3	4	5	6
1	0	1	0	0	0	1
2			1			
3				1		
4						
5		1				
6	1	0	0	0	0	0

stack

S	2	6	

process/visited

1	2	3	4	5	6
0	0	0	0	0	

not proces

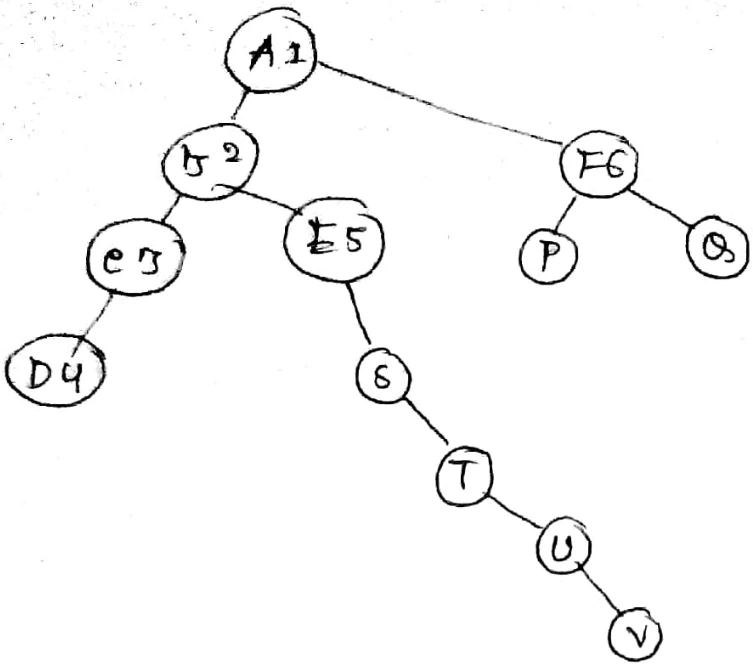
6, 1 process হলো,

1	0	0	0	0	1

```
u=1
{
 for ( v=1 ; v≤ 6 , v++)
 { if (Adj [u] [v] == 1 && processed [v] == 0)

       s.push (v);

 }
   u= s.pop (); // print
   visited [u] =1 ;
}
```



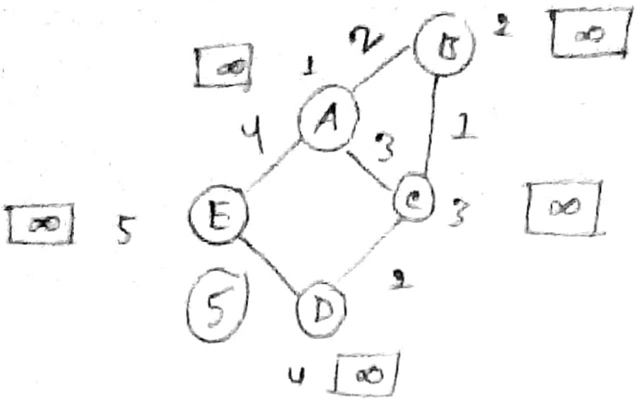
126

Q

Q. Enqueue

Q. Dequeue.

MST : Prim's Algorithm (node by node)



যাদের cost infinity তাদের নিয়েছে।

O

A

```
for (u ∈ |V|)
{ cost [u] = ∞ ;
  par [u] = NULL;
  enqueue ( PQ , u ); {
  s <<            // source node;
  cost [s] = 0; // heapify (logn)
}
u = Dequeue (PQ);
if (Adj [u][v] > 0 && visited [ ] == 0
  && cost [v] > Adj [u][v]
  cost [v] = Adj [u][v];   ← heapify
  par [v] = u ;  ← 1
}
```

Dijsk

Prims

Tree Vertrix

Dijsktra

Coin change

Activity subset selection

1) create random edges and corresponding weights in a file.

2) select n(user input) nodes and corres

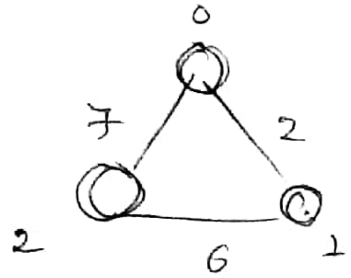
w[1

0	3	1	6	0	0
3	0	5	0	3	0
1	5	0	5	6	4
6	0	5	0	0	2
0	3	6	0	0	6
0	0	4	2	6	0

0

(rand() % 100

)%1(



	0	1	2
0	0	2	7
1	2	0	~~4~~6
2	7	6	0

W [1][1] = 999   W[1][2] = 2   W[1][3] = 7

W [2][1] = 2   W[2][2] = 999   W[2][3] = 6

W [3][1] = 7   W[3][2] = 6   W[3][3] = 999

min = 2

a = u = 1

b = v = 2

[1][2]

Dynamic Algorithm :

greedy → subproblem . local solution এ trap হয়ে যেতে পারে

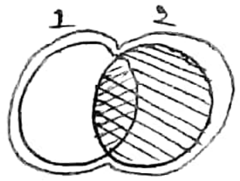
Dynamic → subproblem (overlapping)

কোনোকিছুর ব্যাপারে sorting, তাহলে সেই property এর ব্যাপারে greedy.

আগের step নিয়ে ভাবে না।

but $\boxed{\text{Dynamic}}$ এ পিছনের step গুলোও consider করে।

তাই overlapping



Dynamic

0/1 knapsack : (Dynamic)

	0	1	2	3	4	5	6
P	1	2	3	4	5		

w	1	3	4	5	6	

$\boxed{BW = 10kg}$

P	1	12	3	4	5

Every sqr denotes profit

solution

	0	1	2	3	4 → object
0	0	0	0	0	0
1	0	0	0	0	0
2	0	0	0	0	0
3	0	2	2	2	2
4	0	2	3	3	3
5	0	2	3	4	4
6	0				
7	0				
8	0				
9	0				
10	0				5 → 5 + 3

“কোনো কিছু না নিলে”

6 কেজির জন্য global optimum

4 নাম্বারকে নিলাম

10 kg − 6 kg = 4 kg

যদি 4 kg এর জন্য profit জানা থাকে,

suppose, profit = 12tk

then total profit = 5+12
= 17

Dynamic এ আগে থেকে বের করা profit কাজে লাগানো যায়।

```
if ( w [total object (i তম obj)] <= BW )
        // take | S[BW][total_obj] = P[~~tot_obj~~];
                  = max( S[BW][tot_obj-1], P[tot_obj] + R )
else // Discard
      S[BW][total_obj] = S[BW][tot_obj-1];
```

(R (Remaining) = S[BW − w[tot_obj]][tot_obj − 1];

complexity O(BW × tot_obj) /

O(n²) / O(mn)

Fibonacci → Dynamic

F[i] = f[i-1] + f[i-2]; (আগের দুইটা subproblem এর solution)

one dimension

Longest common subsequence:

(দুইটা string)

✻ solve 0/1 Knapsack Problem using dynamic algorithm for given data.

solve logest common subsequence (LCS) using dynamic algorithm for following 2 strings:

str1 : ABCDEF

str2 : BRECFA

Kruskal : Store All edges into PQ

(i=0 1 → |E|)

{ (a, b) = got min edge ; (dequeue)

IsSame (v1, v2)

{ if (mark [v1] != mark [v2])

return false;

else

return true;

if (! Is same)

{ Union (v1, v2)

Mark v1

~~Mark [v1]~~ = Mark [v1];

for (i = 1 → n |V|)

{ if (mark [i] == mark v1)

mark [i] = mark [v2];

}

}

}

37 page : 2·5·1 – 2·5·3

Theory of induction : $0+0=0$

96 page (A language theoria)

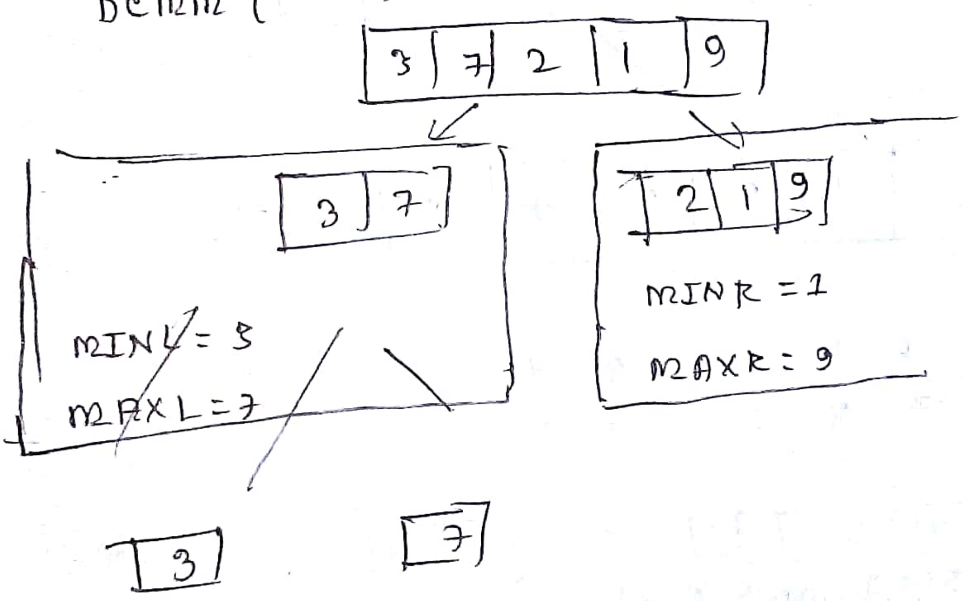
তুলনা করে prove

* If (low high)

MAX = MIN = A[l];

MAX = MIN = A[h];

else, DCMM (



MINL = 7

MAX = 7

MAXL = 3,

MIN = 3

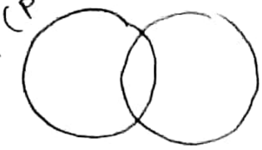
11E

$2^n \rightarrow$ exponential

$x^2 + 2x + 5 \rightarrow$ polynomial.

Theoratical computer science

P(polynomial)

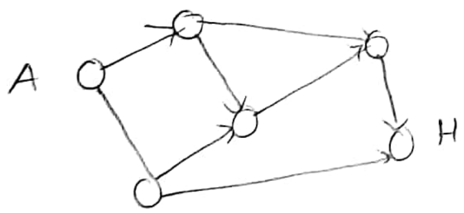


NP (Nonpolynomial / exponential)

Tractable → P

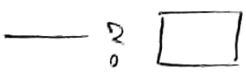
Intractable → NP

NP problem কে polynomial time a solution করা যায়।



A → H যাওয়ার অনেকগুলো path আছে।

যদি একটা system ~~এ~~ path গুলো বলে দেয়, তাহলে solution polynom টাইমেই করা সম্ভব।



Deterministic → complexity অনেক বাড়তে পারে।

Non:

ans + verify → n সংখ্যক

| risky

সর্বনিম্ন 1 (random)

(12E)

P → Polynomial time এ solution

NP → Nonpolynomial

$\boxed{\text{shortest path:}}$ normally deterministic preferable but solution না থাকলে,

nondeterministic এ solution পাওয়ার সম্ভাবনা থাকে।

polynomial এ solution চলে আসতে পারে।

Completeness:

π (problem) is C (is a set of problem) complete if

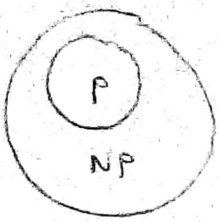
i) $\pi \in C$

ii) All problems in C reduce to π

All problem can be solved in nonpolynomial

deterministic কেউ ঠিক ভাবে করতে পারে।

Scanned by CamScanner



P = NP ?

NP - complete গুলোকে solve

Madian Find:

1 , 10 , 5 , 7 , 9

sort 1 , 5 , 7 , 9 , 10

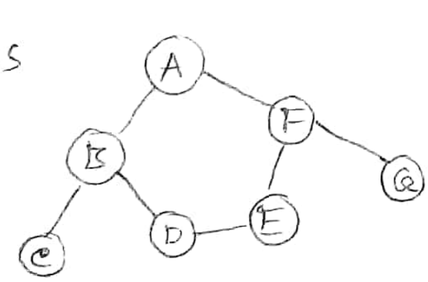
middle 7

madian ~~find~~ reduces to sorting

Graph or Tree : (cycle check)

traverse করে একই node এ ফিরে আসলে cycle.

DFS



cycle checking reduces to DFS

clique / Independent set / vertex cover / max flow

2/3 SAT

boolean
satisfiability: $x_1 \vee/\wedge x_2 \vee/\wedge x_3$ result true?,

2^n

integer addition

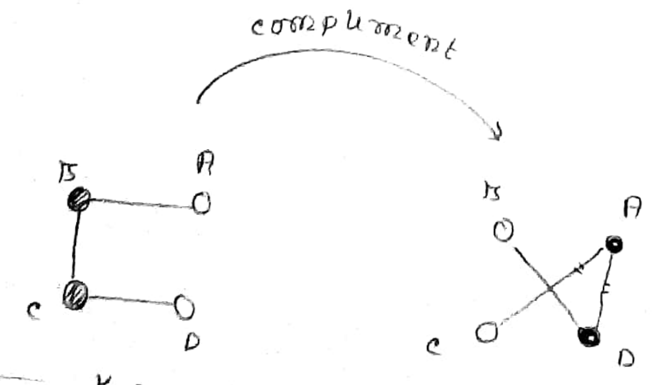
$\boxed{TSP}$ → Traveling salesman

0/1 knapsack → exp (n × m) থাকলে হলেও exp

Pseudopolynomial

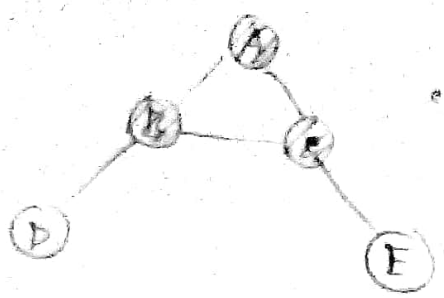
clique: Largest vertex set connected to each other.

vertex cover: Min set of vertex, so that all edges are covered.

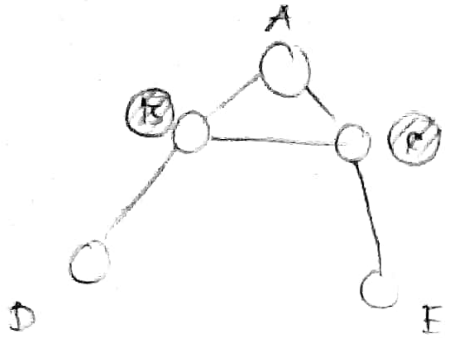


clique ← K = 2

|V| − K



clique



যেখানে দাঁড়িয়ে সবকিছু নজর রাখা যায়।

camera বসানো, tower create

Optimization Problem : What is the lowest possible

Decision Problem: Is there a sol^n of size $\leq k$ → yes/no

polynomial time এ varify করা যায় কিনা।

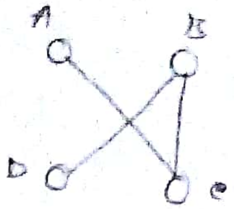
→ bruteforce এ 2^n.

decision দিবে + varify করবে। (time কম লাগবে)

প্রয়োজন অনুযায়ী k এর value change করে

nondeterministic use করে।

vertex cover.



// $c \leftarrow \phi$; [edge set empty]
// randomly edge

// c = { c ∪ select randomed edge from E }

<u>possible vertex :</u>

$c \leftarrow 0$;
while $(E \neq \phi)$
{ c = { c ∪ select randome vertex $u \in V$ };

Eliminate all edges adjacent to u,

}

then check ;

Dynamic Algorith → shortest path

i) Subset sum → { 1, 5, 2, 8, 10 }

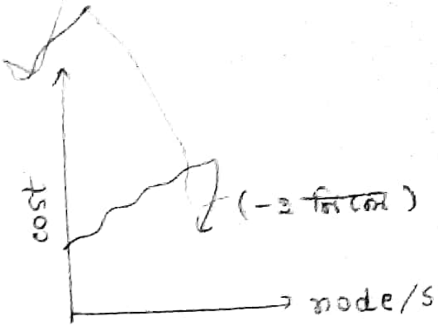
n

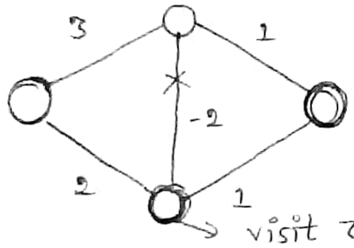
2n

15

<u>Dijkstra</u> → monotonically increasing (Greedy)

কখনও কিছু নামবে না।



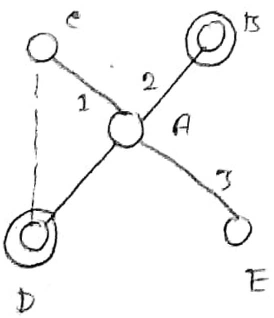


neg(−) এ risk

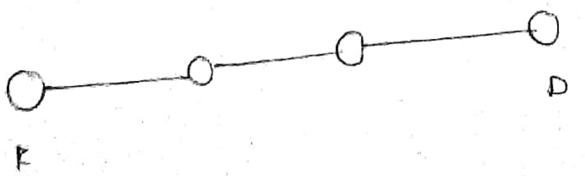
visit করে ফেলেছে

$\boxed{\text{Coin change :}}$

cycle থাকলে অসংখ্যবার

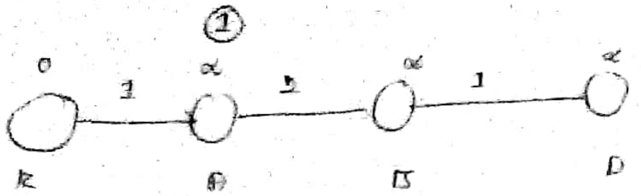


starting থেকে ending এ |V| − 1 সংখ্যক update

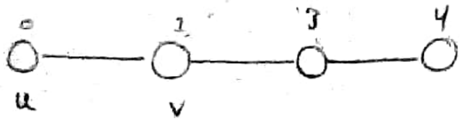


তিনবার update করলেই

Bellmanford



monotonically increasing



প্রত্যেকটা node কে update

|V| ← for |V|-1 times

|E| ← for e ∈ E

update (e);

update (u, v).

distance / dis[v] = min { dis[v], dist [u] + weight

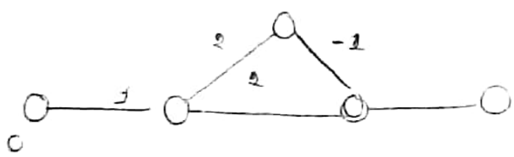
for all edges (E)

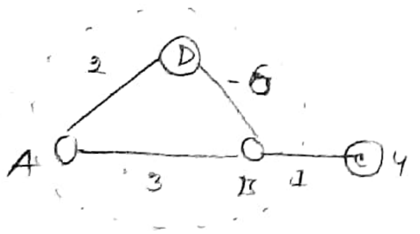
{ if (update (e) == true)

{ printf ("-ve cycle exist")

} exist

// সাইকেল থাকলেও effect করবে না।





negative cycle

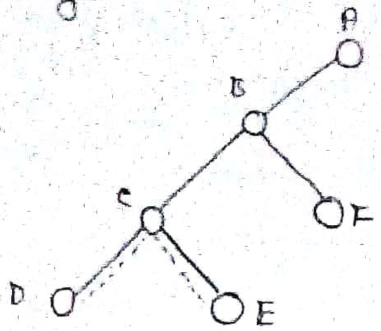
cost কমতেই থাকে।

no feasible solution.

O (VE)

~~FW~~ Floyd warshall

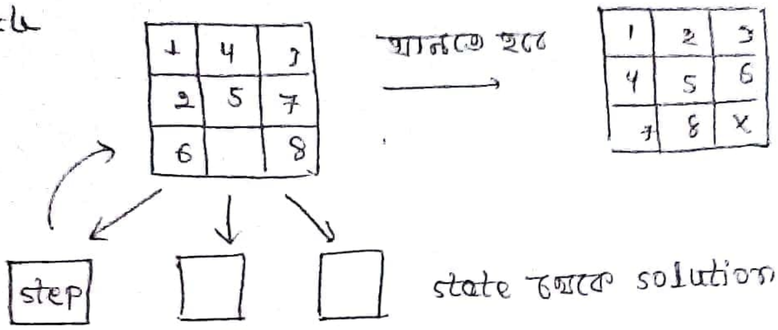
Backtracing:



DFS এ

ABCD→F

Puzzle



state থেকে solution পাওয়ার possibility কত?

node

f( ) → possibility

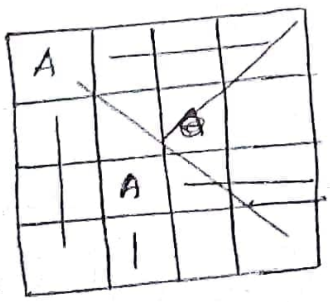
solution possible না হলে back এ চলে যায়।

4টা Queen

N-Queens problem : N = 4 8

চেস বোর্ড

4×4



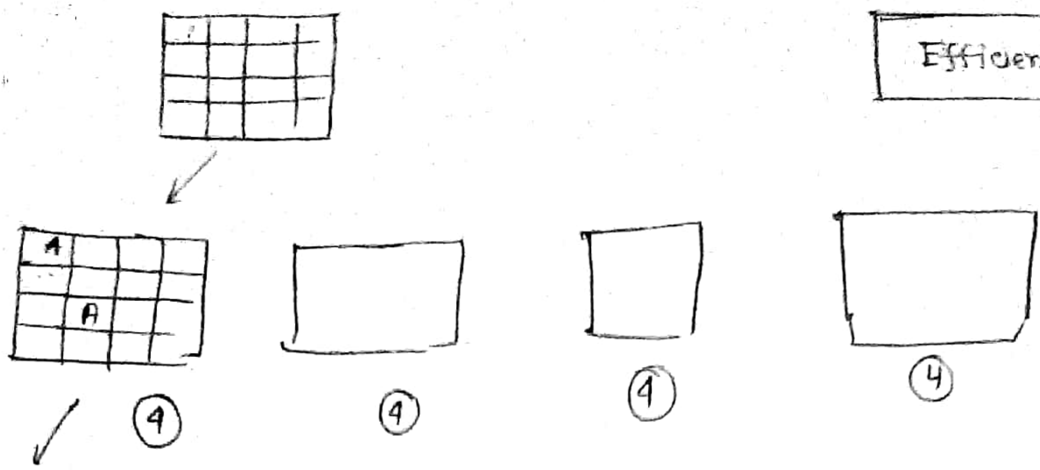
solution possible না।

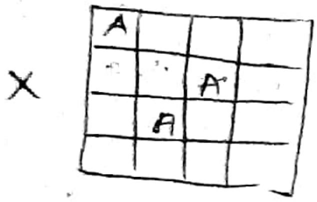
Scanned by CamScanner

total টা → state space

| naive approach এ back track করবে না।

$\boxed{\text{Efficiency } n^n}$





1, 3, 2

one dimension array

P আর যায়নে আগাবো না।

Backtracking

	row 1	2	3	4
RSA	1	~~3~~ 4		

column এর no.

$r = 1$

NBK (r, N) {

for (c = 1 → N)

if (place (r, c) হলে) true/false, true হলে place করতে পারব।

{ RSA [r] = c;

NBK (r++, N);

}

}

NBK (1, 4) c = 1

NBK (2, 4) c = 3 (backtracking c = 4)

| NBK (3, 4) এ solution নেই।

place (r, c) {

// same row (x) // same diagonal

← column ✓

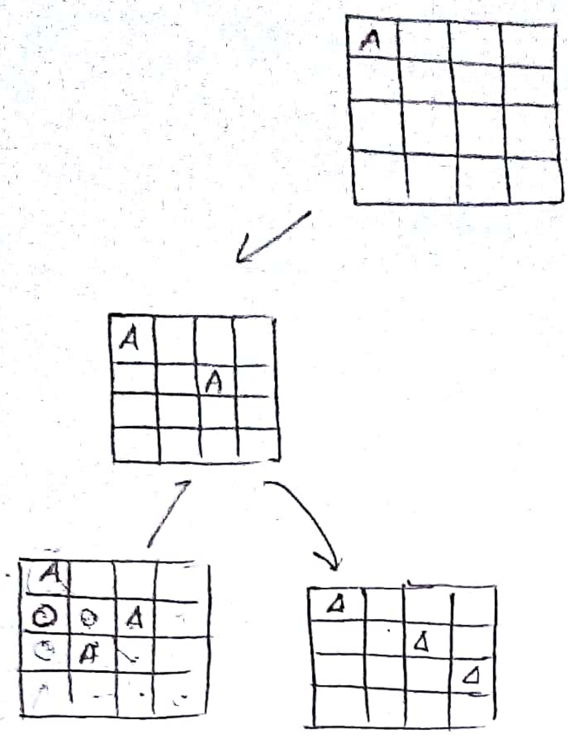
for (row = 1 → ≤ r) if (c == RSol [row])

return 0;

if (RSol [row] − c) == r − row;

}

※ Break করানোর ব্যবস্থা করতে হবে।



Backtracking ↗ 4!

⊛ NP-P

⊛ completeness

* clique , vertex cover

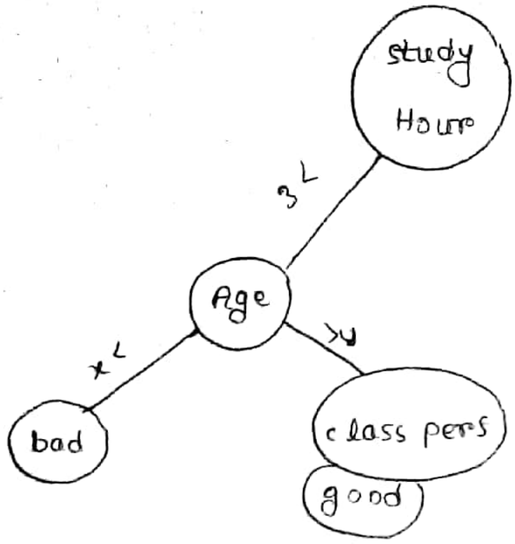
* Dynamic → shortest path (Bellmanford , floyd warshal

* Backtracking (N-Queen Problm)

Decision Tree:

ST

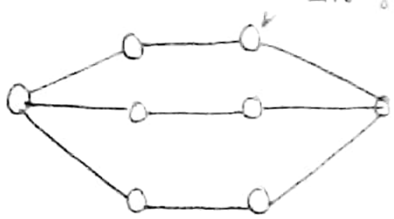
study | Age | Height | weight | class pers | ⟶ Result



$\boxed{\text{AVL}}$ tree and Hashing

multistage graph (shahani) ⟶ Dynamic

যাদের মূল্যও consider করতে হবে।



Bin Packing

Boolean SAT

Hashing: i) indexing

ii) security Hash code

Lab:

1. Sort: i) Insertion Sort
   ii) Merge sort
   iii) Bubble
   iv) Heap sort
   v) Quick sort

2. Divide and conquer:
   i) Binary search
   ii) min max
   iii)

3) Priority Queue

4) convex Hull

5) Greedy : i) Knapsack
   ii) ~~Gr~~ Huffman

6) Kruskal

7) Dynamic → 0/1 Knapsack

8) Longest common subsequent

12D

- Problem
  - solvable
    - Decidable (Algorithm & procedure)
    - Undecidable (only procedure)
  - unsolvable

bIbO

bounded input bounded output

<u>Halting Problem</u>: infinity loop machine এ create হতে পারে।

Dynamo or "

401 (error)