Date

ADA - 1

Algorithm :→

→ An algorithm is a well defined computational procedure that take some value as input & produces some value or set of values as output.

It is a sequence of computational steps that transform your input & output. It is a step by step procedure to achieve the required results.

Input → sequence of n no.'s.
Output → Certain reordering of input sequence.

Pseudo Code :→

It is a compact & informal high-level description of a computer programming algo. that uses structural convention of some programming language. No syntax (particular) is followed. (Machine independent)

A flow chart is systematically representation of algo using figures.

Performance Analysis of an algo :→

The goal of performance analysis is to determine which part of a prog. to be optimized for speed & memory usage.

(i) Space Complexity :→

Space complexity of an algo. is the no. of elementary obj that this algo. needs to store during its execution. The no. is computed in w.r.t. size of

Date

input data. It is a concept that measures amount of memory space required for its execution. It is measured by Asymptotic notation.

(ii) Time complexity :-
The no. of machine instructions which a prog. executes during its running time is called time complexity. The no. depends upon the size of prog. input. The time taken by the prog. is sum of compile time & run time. In time complexity we consider run-time only

Q. Write an algo. to find out the maximal element in size n.

⇒
```
max = a[1]
for (i = 2, i ≤ n, i++)
if a[i] > 2 then
  max = a[i]
end if
end for
Return max.
```

Q. largest of 3 no. is
Q. Considering the sum of first n natural no.'s.
Q. Linear search.
Q. Binary search.

Q1/R ADA-2 Date

Asymptotic Notations :→

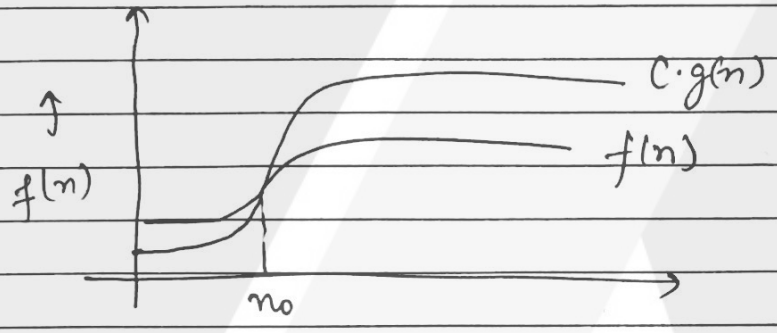
It is a mean of comparing relative performance. In time complexity of an algo. concerns determining an expression of the no. primitive operations needed as a function of problem size.

Big-oh :→

$f(n) = Og(n)$

$c > 0$ & $n_0 \geq 1$ such that

$|f(n)| \leq c|g(n)| \quad \forall n = n_0$



Q: $4n^2 + 7n + 12 \qquad g(n) = n^2$

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

$|4n^2 + 7n + 12| \leq |4n^2| + |7n| + |12|$

$\leq 4|n^2| + 7|n^2| + 12|n^2|$

$\leq 23|n^2|$

$f(x) = 4x^2 + 7x + 12$

$g(x) = x^2$

$f(x) \leq c \cdot g(x)$

for $x > 8$

$g(x) = 64$

$f(x) = 4|9| + 7(9) + 12$

$394 \leq 405$

Big oh notation gives an upper Bound on the growth rate of fun. The statement $f(n)$ is $O g(n)$ means that growth rate of $f(n)$ is no more than growth rate of $g(n)$.

Ex. $2n+10$ is $O(n)$

$c = 12 \quad n = 1$

$(n+1)^2$ is $O n^2$

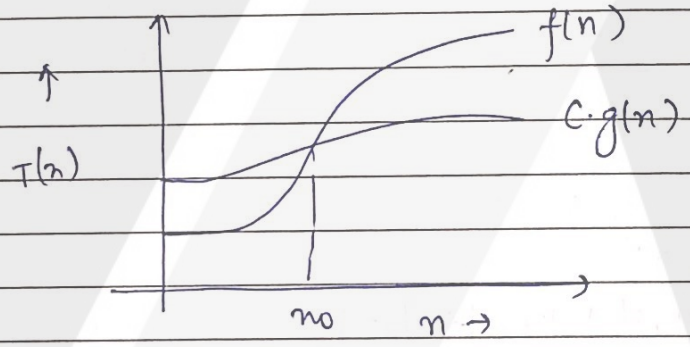
Big Omega (Ω) (Provides lower bound of function)

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

if there are positive constants $c > 0$ & $n_0 \geq 1$ such that

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

$\forall n \geq n_0$



·) Used for best case.

·) for insertion sort the array is in sorted form.

Big Thetha (Θ) :→

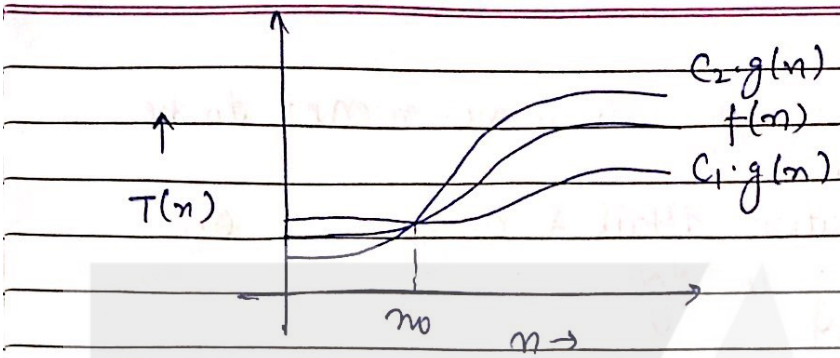
$f(n)$ is $\Theta g(n)$

if there is an integer no. & the true real constants c_1 & c_2 such that

$$c_1 \cdot g(x) \leq f(x) \leq c_2 \cdot g(x)$$

$\forall x \geq x_0$

Date



#

1. Design of algo
   i) Incremental Approach.
   ii) Divide & Conquer Approach.
   iii) Dynamic Programming. (computation results which can be used in futures).
2. Algorithm validation.
3. Analysis of algo.
4. Algo. testing (debugging).

Time Complexity :→ The amount of time needed by an algo (Program) is referred to as time complexity. Time complexity depends on the size of input thus it is a fun. of input size n. $\boxed{f(n)}$

Different time can arise for the same algo. we deal with best case, avg. case & worse case for an algo. The minimum amount of time that an algo. requires for an input of size n is referred to as best case complexity.

Space Complexity :→
The amount of memory needed by the program to run to completion is known as space complexity.

Constant = 1

Big-Oh

$\log n = \log_2 n$

$2^n = 2^n$

Date

Pseudo Code:-

Pseudo is meant to give info. code is not meant to be compiled on computer.

- Hides the implementation detail & only focus on computational aspect of an algo.
- Machine independent.

← (Replacement operator) ↔ (Exchange operator)

Assignment statement, conditional statement.....

Q) $f(n) = 3n+5$

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

$3n+5 \geq 3n+n$

$n_0 \geq 5 \quad C=4$

Exponential fun.

$2^n + 6n^2 + 3n$

$n^2 \geq 3n$

$2^n \geq 7n^2$

$n_0 = 4$

$C = 8$

Q) $f(n) = 2n^3 + n^2 + 2n$

$= 5n^3$

$C = 5 \qquad n_0 \geq 1$

$2n^3 + n^3 + 2n^2$

for $n^3 \geq 2n^2$

$2n^3 + n^3$

$3n^3 \quad (C=3 \text{ for } n_0 \geq 2)$

$4 \times 2^n + 3n$

$2^n \geq n$

$n_0 = 1$

$C = 7$

Q) $27n^2 + 16$

$C_1 \cdot g(n) \leq f(n) \leq C_2 \cdot g(n)$

$27n^2 \leq 27n^2 + 16$

$C_1 = 16$

for $\forall n \qquad 27n^2 + 16 \leq C_2 \cdot g(n)$

$C_2 = 28$

$n_0 = 8$

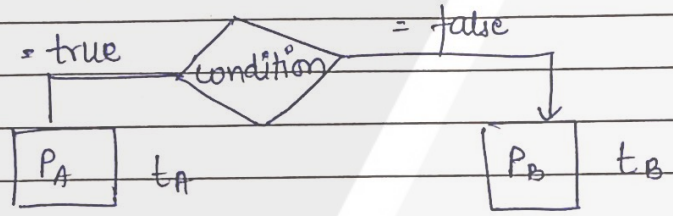
4/01 ADA - III Date

Analysis algorithm Control structures :-

Sequencing → P_A P_B
t_A t_B
$\max(t_A, t_B)$

if else
for
while
Recursion

if then else



$\max(t_A, t_B)$

for

```
for i<- 1 to m
{
   P(i)  → t
}
```

⇒ (mt)

for this loop P_i is computed for each iteration for i=1 to m & suppose if P_i take constant time t for computation then for m iteration computation time will be (mt)

If computational time t_i for P(i) varies as a fun. of i then total computation time for the loop is given not by multiplication but by submission.

$$\sum_{i=1}^{m} t_i$$

Date

```
for(i←1 to m)
{
    Sum ← Sum + A[i]  → ti
}
```

$$\Rightarrow \sum_{i=1}^{m} t_i$$

$$\sum_{i=1}^{m} \Theta(1) \quad \text{for constant}$$

$$\Theta(m)$$

```
for(i← 1 tom)
{
  for( j ← 1 to m)
  {
    Sum ← Sum + A[i][j]
  }
}
```

$$\Rightarrow \sum_{i=1}^{m} \sum_{j=1}^{m} t_{ij}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{m} \Theta(1)$$

$$\sum_{i=1}^{m} \Theta(m)$$

$$\Theta \sum_{i=1}^{m} (m)$$

$$(m^2)$$

Date _____

```
for (i ← 1 to m)
{
  for j ← 1 to i
  {
    sum ← sum + A[i][j]
  }
}
```

$$\Rightarrow \sum_{i=1}^{m} \sum_{j=1}^{i} t_{ij}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{i} \theta(1)$$

$$\sum_{i=1}^{m} \theta(m)$$

$$\frac{m(m+1)}{2} \Rightarrow \theta(m^2)$$

highest power.

```
for (i ← 2 to m-1)
{
  for (j ← 3 to i)
  {
  }
}
```

$$\sum_{i=2}^{m} \sum_{j=3}^{i} t_{ij}$$

$$\sum_{i=2}^{m} \sum_{j=3}^{i} \theta(1)$$

$$\sum_{i=2}^{m-1} \theta(i)$$

$2+3+\cdots+(m-1)$

$$\sum_{i=2}^{m-1} \sum_{j=3}^{i} t_{ij}$$

$(2+3+4+5+\cdots+(m-1)) - (3+4+5+\cdots+i)$

$$\left[\frac{(m-1)(m)}{2} - 1\right] - \left[\frac{i(i+1)}{2} - 1 - 2\right]$$

$\sum m = \frac{m(m+1)}{2}$

$$\frac{(m-1)(m-1+1)}{2} = \theta(m^2)$$

highest power.

Date

To solve recursion :-
i) Substitution
ii) Iteration
iii) Changing variable method
iv) Recursion tree
v) Master Method.

Q) Solve recurrence by iteration method :-

$t(n) = t(n-1)+1$ & $t_1 = \theta(1)$. You have to show that above recurrence is asymptotically bound by $\theta(n)$.

Iteration Method :-
The basic idea is to expand the recurrence & express it as a summation of terms dependent only on n & the initial conditions.

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

$t(n) = [t(n-2)+1] + 1$
$= t(n-2) + 2$
$t(n-3) + 3$
⋮

$t(n) = t(n-k) + k$
here if $k = n-1$

$t(n) = t(1) + (n-1)$
$= \theta(1) + (n-1)$

Date

Q) $T(n) = T(n/2) + n$ & $T(1) = \Theta(1)$

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

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

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

$T(n) = T(n/8) + n/4 + n/2 + n$

$T(n/8) = T(n/16) + n/8$

$T(n) = T(n/16) + n/8 + n/4 + n/2 + n$

$= T\left(\frac{n}{2^k}\right) + \sum_{j=0}^{k-1} n/2^{k-1}$

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

$n = 2^k$

$\log n = k \log 2$

$k = \log_2 n$

$= T(1) + \sum_{j=0}^{k-1} \frac{n}{2^{k-1}}$

$= \Theta(1) + \sum_{j=0}^{\log_2 n - 1} \frac{n}{2} \log_2 n - 1$

$= \Theta(1) + \Theta(n)$

$= \Theta(n)$

Q) $T(n) = 3T\left\lfloor\frac{n}{4}\right\rfloor + n$ $\quad O(n)$

Date _____

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

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

$2\left(2T\left\lfloor \frac{n}{4} \right\rfloor + \frac{n}{2}\right) + n$

$2^2 T\left\lfloor \frac{n}{4} \right\rfloor + n + n$

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

2^k

$n + n + n$ - - -

$T\left(\frac{n}{2}\right) = 2^3 T\left\lfloor \frac{n}{8} \right\rfloor + n + \frac{n}{2} + \frac{n}{4}$

kn

$\vdots$

$2^k T\left\lfloor \frac{n}{2^k} \right\rfloor + n + \frac{n}{2} + \frac{n}{4} - - - \frac{n}{2^{k-1}}$

$n + n + n$ - - -

$T(n) \Rightarrow 2^k T\left\lfloor \frac{n}{2^k} \right\rfloor + \left\lfloor \frac{1}{2^k} - 1 \right\rfloor$ kn

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

$n = 2^k$

$\log n = k \log 2$

$k = \frac{\log n}{\log 2} \Rightarrow \log_2 n$

(kn)

2) $n \log_2 n$

(2-3) ques
(Substitution Method)
(do on your own)

Date

Iteration

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

$T(n/2) \Rightarrow 2 \cdot T(n/4) + n/2$

$T(n) \Rightarrow 2[2T(n/4) + n/2] + n$

$T(n) \Rightarrow 4Tn/4 + n + n$

$T(n/4) \Rightarrow 2T(n/8) + n/4$

$T(n) = 4[2T(n/8) + n/4] + n + n$

$8Tn/8 + \frac{4n}{4} + n + n$

$2^3 \cdot T \cdot \frac{n}{2^3} + n + n + n$

$2^K \cdot T \cdot \frac{n}{2^K} + nk$

$2^{\log_2 n} \cdot T \cdot \frac{n}{2^{\log_2 n}} + n \cdot \log_2 n$

$\Rightarrow n \log_2 n$

$\frac{n}{2^K} \leq 1$

$n \leq 2^K$

$K \leq 2^n$

$K \leq \log_2 n$.

Master's Method :→

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

$T(n) = aT\left(\frac{n}{b}\right) + f(n)$ $\quad a \geq 1 \quad b \geq 1$

Date _____

↳ cost function: how much

Masters Method:-

1). $f(n) = O(n^{E-e})$ for some $e > 0$ then
$T(n) \in \Theta(n^E)$

2). $f(n) = \Theta(n^{E-e})$ ~~for some~~ then $T(n) \in \Theta(f(n) \cdot \log n)$

→ mid cond^n (check from book) corman

3) $f(n) = \Omega(n^{E+e})$ for some $e > 0$ and
$f(n) \in O(n^{E+\delta})$ for some $\delta \geq e$ then
$T(n) \in \Theta[f(n)]$

Q1. $T(n) = 4T\left(\frac{n}{2}\right) + n$

Q2

Q3.

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

$a = 4, \quad b = 2, \quad f(n) = n$

$\log_b^a = \log_2^4$

$n^{\log_a^b} \Rightarrow n^{\log_2^4} \Rightarrow n^2$

$e = 1 \qquad E = 2$

$T(n) \in \Theta(n^2)$ case (i)

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

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

$\Rightarrow 4^2 T\left(\frac{n}{4}\right) + 2n + n$

Q4. $T(n) = 2T\left(\frac{n}{2}\right) + n - 1$ (second case applied)

Date _____

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

$\Rightarrow 4^2\left[4T\left(\frac{n}{8}\right) + \frac{n}{4}\right] + 2n + n$

$\Rightarrow 4^3 T\left(\frac{n}{8}\right) + 4n + 2n + n$

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

$a = 4, b = 2, f(n) = n^3$

$n^{\log_2^4} \Rightarrow n^2$

$< f(n)$ so add

$T(n) \in \theta(n^3)$

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

$a = 4, b = 2 \quad f(n) = n^2$

$n^{\log_2^4} \Rightarrow n^2$

$f(n) = n^2$ equal so $\Rightarrow n^2 \log n$

cost fun :→ the cost to split the problem or combine the soln of sub-problems is given by $f(n)$.

Q4 $T(n) = 2T\left(\frac{n}{4}\right) + 1$ and $T(1) = \theta(1)$

$E = \frac{\log a}{\log b}$; $f(n) = 1$; $a = 1$; $b = \frac{4}{3}$

$n^E \Rightarrow n^{\log_b^a} \Rightarrow n^{\log_{4/3}^1} \Rightarrow n^0 \Rightarrow 1$

$T(n) \in \theta(1 \cdot \log n)$

$T(n) = \theta(\log n)$

Date _____

Changing variable method for solving recurrences :-

Q4. $T(n) = 2T(\lfloor\sqrt{n}\rfloor) + \log n$

$f(n)$

$m = \log n$

$n = 2^m$

$n^{1/2} = 2^{m/2}$

$T(2^m) = 2T(2^{m/2}) + m$

Assume $S(m) = T(2^m)$

$S(m) = 2S(\frac{m}{2}) + m$

$O(m \log m)$

$O(\log n (\log(\log n)))$

Q. $T(n) = 2T(\sqrt{n}) + 1$

1) calculate asyntotic bounding using change in variable

Q4 2. $T(n) = 3T(n^{1/3}) + \log_3 n$ } by any method.

** Q4. a $T(n) = 3T(n^{1/3}) + \log_3 n$

$a = 3$

let $\log_3 n = m$

$n = 3^m$

$\sqrt[3]{n} = \sqrt[3]{3^m}$

$n^{1/3} = \sqrt[3]{3^m} = 3^{m/3}$

$T(3^m) = 3T(3^{m/3}) + m$

$S(m) = 3S(m/3) + m$ $\quad S(m) = T(3^m)$

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

$= O(\log_2 n) \log(\log_2 n)$

Date

Q. $T(n) = 2T(\sqrt{n}) + 1$

$m = \log n$

$n = 2^m$

$n^{1/2} = 2^{m/2}$

$T(2^m) = 2T(2^{m/2}) + 1$

let $S(m) = T(2^m)$

$S(\frac{m}{2}) = T(2^{m/2})$

$T(2^m) = 2T(2^{m/2}) + 1$

$S(m) = 2S(m/2) + 1$

$a = 2 \quad b = 2$

$n \log_2^2 = n$

$f(n) = 1$

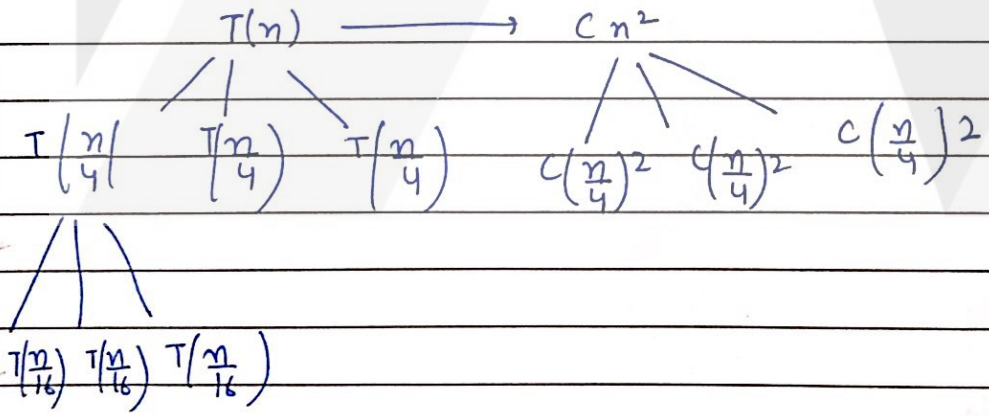
$\Theta(m)$

$\Theta(\log n)$

#

Recursion Tree :→

$T(n) = 3T(n/4) + n^2 c$



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

Date

$\frac{n}{4^i} = 1$

$n = 4^i$

$\boxed{\log_4 n = i}$

Total cost $= 3^i C \left(\frac{n}{4}\right)^2$

$$Cn^2 + \frac{3}{16}Cn^2 + \left(\frac{3}{16}\right)^2 Cn^2 - - - - - \left(\frac{3}{16}\right)^{\log_4 n - 1} + \underbrace{\Theta(n^{\log_4 3})}_{\text{Extra cost}}$$

$\propto$ G.P.

Last level at depth $\log_4 n$

$\frac{a}{1-r} = \frac{Cn^2}{1-\frac{3}{16}} \Rightarrow \frac{16}{13} Cn^2$

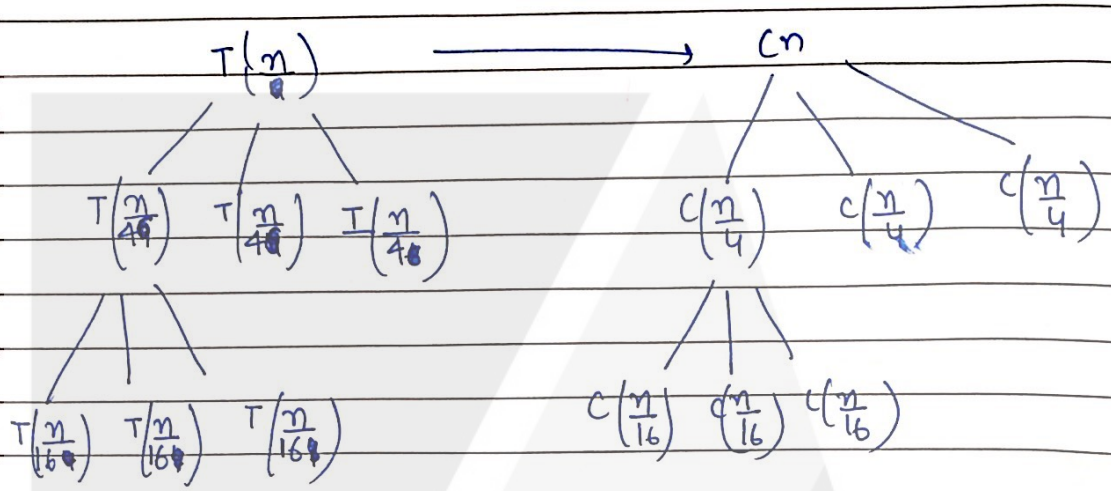
$= 3^{\log_4 n}$

$= n^{\log_4 3}$

$= \frac{16}{13} Cn^2 + n^{\log_4 3}$

$\Theta(n^2)$

Q. $T(n) = 3T\left(\frac{n}{4}\right) + cn$



$T(1)$

$\frac{n}{4^i} = 1$

$\boxed{i = \log_4 n}$

$$cn + \frac{3}{4}cn + \left(\frac{3}{4}\right)^2 cn + \cdots + \left(\frac{3}{4}\right)^{\log_4 n - 1} + \theta\left(n^{\log_4 3}\right)$$

depth $\log_4 n$

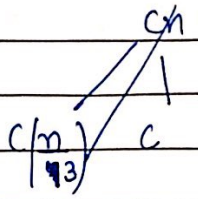
$\frac{a}{1-r} = \frac{cn}{1-\frac{3}{4}} \Rightarrow 4cn$

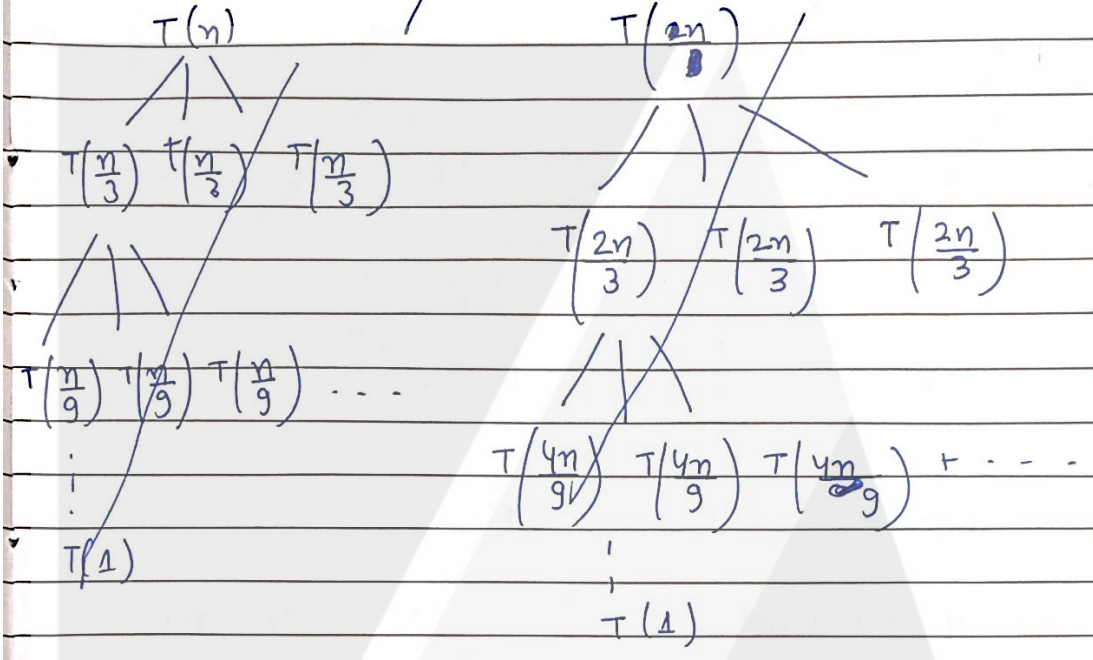
$\Rightarrow 4cn + n^{\log_4 3}$

$\theta(n)$

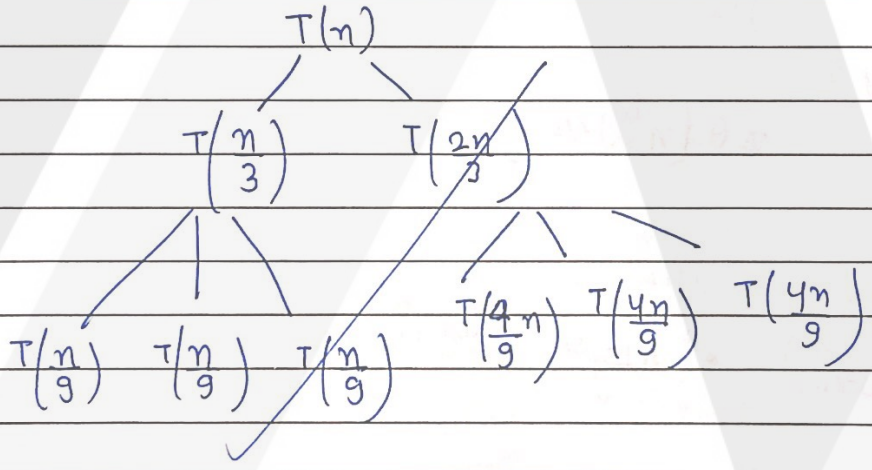
Date _____

Q4. $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$



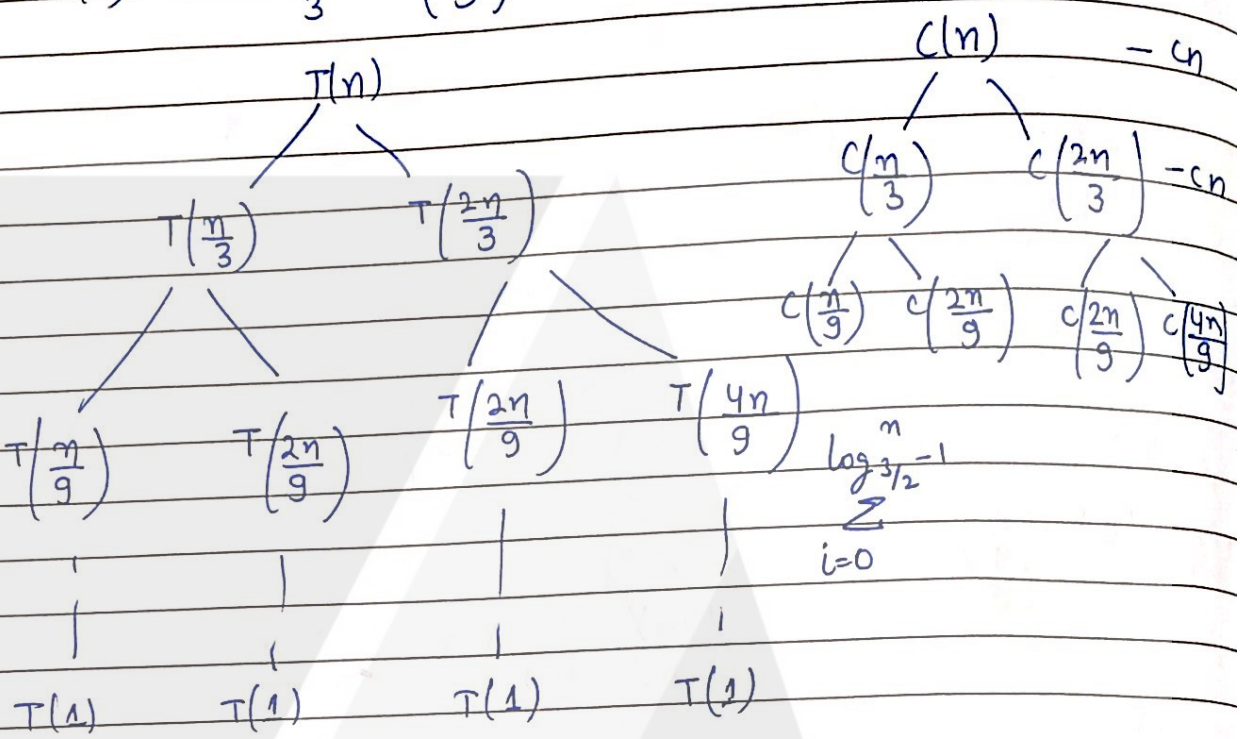


$\frac{n}{3^i} = 1$ $\quad$ $\frac{2n}{3^i} = 1$



Date

Q. $T(n) = T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + cn$



No. of nodes at last level = 2^i

$n \cdot \left(\frac{2}{3}\right)^i = 1$

Taking the depth of longest branch

$\log_{3/2} n = i$

$2^{\log_{3/2} n}$

$= \theta\left(n^{\log_{3/2} 2}\right)$

$cn(i) + 2^{\log_{3/2} n}$

$c(n)\left(\log_{3/2} n\right) + n^{\log_{3/2} 2}$

$O(n \log_{3/2} n)$ $O(n \log n)$

$O(n \log n)$

Q2. $T(n) = 4T\left(\frac{n}{2}\right) + Cn$

Date

Module - 2

Divide & Conquer

Divide the unsorted list into two list of about half the size.

Conquer: Sort each of these two sub-list recursively until we have list sizes of length 1 in which case the list itself is returned.

Combine the solutions of sub-problems to create a solution of the original problem.

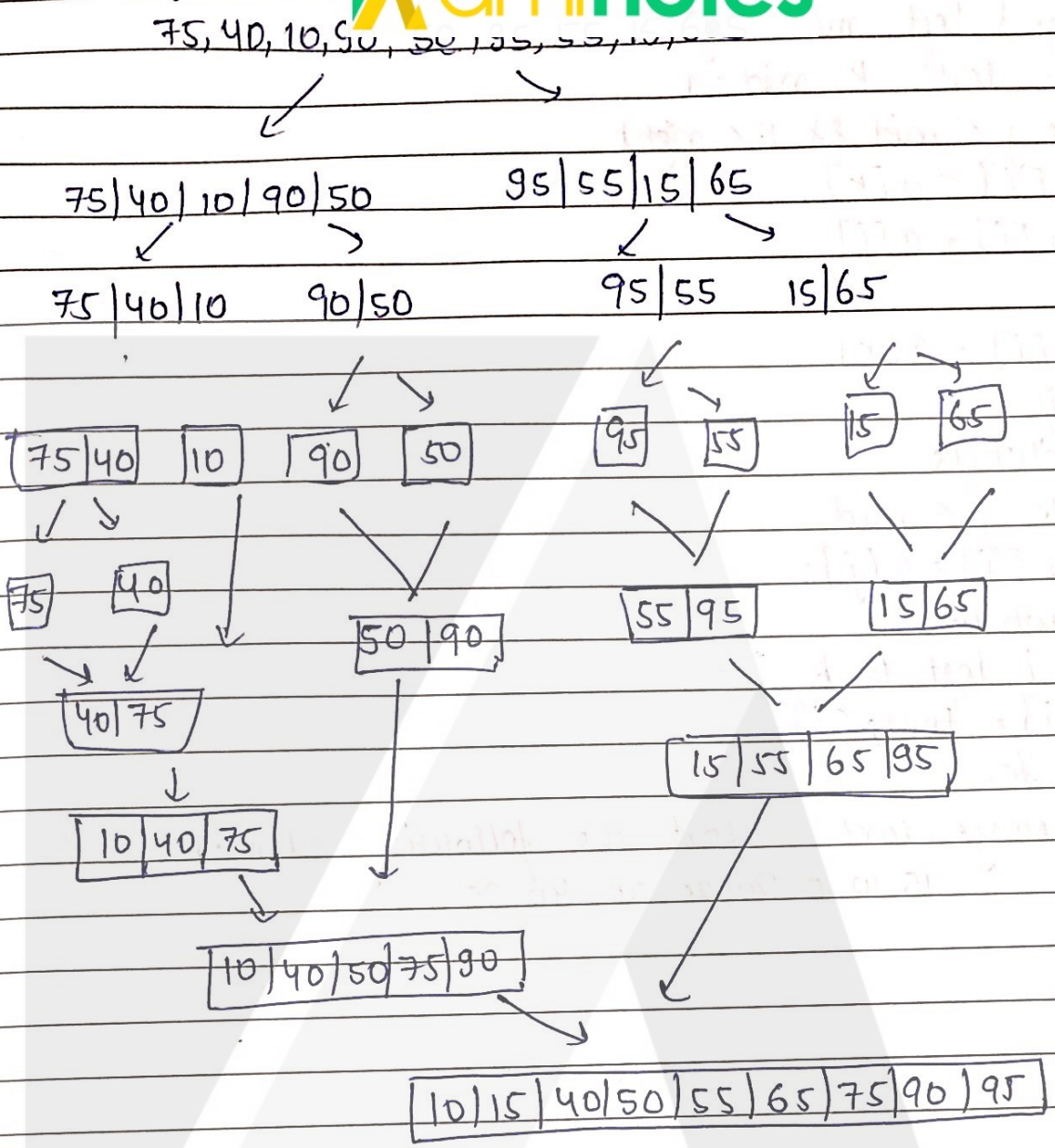
1). Binary search : is an algo. such that

```
Binary search ( arr.[1,2,---n], value, low, high)
1.  If(high < low)
2.  return -1
3.  End if
4.  mid = (low + high)/2
5.  if ( arr [mid] > value )
6.  return Binary search(arr, value, low, mid-1)
7.  Else if ( arr [mid]< value)
8.  return Binary search (arr, value, mid+1, high)
9.  Else
10. return mid
11. End if
12. End if
```

It is a logarithmic algorithm & runs in big-oh of $\log_2 n$ times.

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

2) Merge-Sort



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

Algo:-

Mergesort (left, right)

1. If left < right
2. Mid = (left + right)/2
3. Mergesort (left, mid)
4. Mergesort (mid+1, right)
5. Merge (left, mid, right)
6. End if.

Date

1. Merge (left, mid, right)
2. i=j= left, K= mid+1
3. while j ≤ mid && K ≤ right
4. If a[i] < a[K]
5. temp [i] = a[i]
6. Else
7. temp[i] = a[K]
8. End if
9. End while
10. while j ≤ mid
11. temp [i] = a[j];
12. End while
13. For i left to R
14. a[i] = temp[i]
15. End for;

Q Using Merge-sort , sort the following elements

15, 10, 5, 20, 25, 30, 40, 35

Date

04/01/18

Divide & Conquer Approach:

Strassen's Matrix Multiplication :-

$$\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$$

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$
$$C_{12} = A_{11}B_{12} + A_{12}A_{22}$$
$$C_{21} = A_{21}B_{11} + A_{22}B_{21}$$
$$C_{22} = A_{21}B_{12} + A_{22}B_{22}$$

$$A = \left[\begin{array}{cc|cc} 1 & 2 & 3 & 4 \\ 0 & 6 & 0 & 3 \\ \hline 4 & 1 & 1 & 2 \\ 0 & 3 & 5 & 0 \end{array}\right] \qquad B = \left[\begin{array}{cc|cc} 1 & 4 & 2 & 7 \\ 3 & 1 & 3 & 5 \\ \hline 2 & 0 & 1 & 3 \\ 1 & 4 & 5 & 1 \end{array}\right]$$

$$A_{11} = \begin{bmatrix} 1 & 2 \\ 0 & 6 \end{bmatrix} \qquad B_{11} = \begin{bmatrix} 1 & 4 \\ 3 & 1 \end{bmatrix}$$

$$A_{12} = \begin{bmatrix} 3 & 4 \\ 0 & 3 \end{bmatrix} \qquad B_{12} = \begin{bmatrix} 2 & 7 \\ 3 & 5 \end{bmatrix}$$

$$A_{21} = \begin{bmatrix} 4 & 1 \\ 0 & 3 \end{bmatrix} \qquad B_{21} = \begin{bmatrix} 2 & 0 \\ 1 & 4 \end{bmatrix}$$

$$A_{22} = \begin{bmatrix} 1 & 2 \\ 5 & 0 \end{bmatrix} \qquad B_{22} = \begin{bmatrix} 1 & 3 \\ 5 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} 17 & 32 & 31 & 30 \\ 21 & 18 & 33 & 33 \\ 11 & 25 & 22 & 38 \\ 19 & 3 & 14 & 30 \end{bmatrix}$$

Each of these four eqn specifies two multiplication of $n/2 \times n/2$ matrices and addn of these $n/2 * n/2$ products

Date

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

$\Rightarrow \theta(n^3)$ complexity

So, in order to reduce stresser discover a diff. recursive approach that requires only 7 recursive multiplication of $\left(\frac{n}{2} \times \frac{n}{2}\right)$ matrices & big θ plus $\theta(n^2)$ scaler addition & substraction

$$T(n) = 7T\left(\frac{n}{2}\right) + \theta(n^2)$$

$(n^{\log_2^7})$

$\Rightarrow (n^{2.81})$

In strassen's formula there are 7 multiplication & 18 add^n or substraction.

$$P_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22})$$

$$P_2 = (A_{21} + A_{22}) \times B_{11}$$

$$P_3 = A_{11} \times (B_{12} - B_{22})$$

$$P_4 = A_{22} \times (B_{21} - B_{11})$$

$$P_5 = (A_{11} + A_{12}) \times B_{22}$$

$$P_6 = (A_{21} - A_{22}) \times (B_{11} + B_{12})$$

$$P_7 = (A_{12} - A_{22}) \times (B_{21} + B_{22})$$

$$C_{11} = P_1 + P_4 - P_5 + P_7$$

$$C_{12} = P_3 + P_5$$

$$C_{21} = P_2 + P_4$$

$$C_{22} = P_1 + P_3 - P_2 + P_6$$

Date

$P_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22})$

$$= \begin{bmatrix} 2 & 4 \\ 5 & 6 \end{bmatrix} \times \begin{bmatrix} 2 & 7 \\ 8 & 2 \end{bmatrix}$$

$$\begin{bmatrix} 4+4 & 14+8 \\ 10+40 & 35+12 \end{bmatrix} \Rightarrow \begin{bmatrix} 36 & 22 \\ 58 & 47 \end{bmatrix}$$

$P_2 = (A_{21} + A_{22}) \times B_{11}$

$$= \begin{bmatrix} 5 & 3 \\ 5 & 3 \end{bmatrix} \begin{bmatrix} 1 & 4 \\ 3 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 14 & 23 \\ 14 & 23 \end{bmatrix}$$

$$P_3 = \begin{bmatrix} -3 & 12 \\ -12 & -24 \end{bmatrix}$$

$P_4 = A_{22} \times (B_{21} - B_{11})$

$$= \begin{bmatrix} 1 & 2 \\ 5 & 0 \end{bmatrix} \begin{bmatrix} 1 & -4 \\ -2 & 3 \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} 1-4 & -4+6 \\ 5 & -20 \end{bmatrix} \Rightarrow \begin{bmatrix} -3 & 2 \\ 5 & -20 \end{bmatrix}$$

$$P_5 = \begin{bmatrix} 34 & 18 \\ 45 & 9 \end{bmatrix}$$

$$P_6 = \begin{bmatrix} 3 & 27 \\ -18 & -18 \end{bmatrix}$$

$$P_7 = \begin{bmatrix} 18 & 16 \\ 3 & 0 \end{bmatrix}$$

prove for → odd power $[n^{\log 7}]$

Date pg. ~~734~~ 735 }.

$C_{11} = P_1 + P_4 - P_5 + P_7$

$= \begin{bmatrix} -1 & 9 \\ 13 & -13 \end{bmatrix} - \begin{bmatrix} 82 & 34 \\ 48 & 9 \end{bmatrix}$

$48 - 13 = 35$.

$= \begin{bmatrix} -53 & 25 \\ -35 & 27 \end{bmatrix}$

$= \begin{bmatrix} 17 & 22 \\ 21 & 18 \end{bmatrix}$

Q4. $\begin{bmatrix} 1 & 3 \\ 5 & 7 \end{bmatrix} \quad \begin{bmatrix} 8 & 4 \\ 6 & 2 \end{bmatrix}$

$P_1 = 8 \times 10 \Rightarrow 80$;

C: ~~[5 x 10]~~

9×3 mul + 2 add.

27 mul + 18 add.

~~(n3) mult +~~

24/02 Date

Greedy Algo. → diff. from dynamic programming

A greedy algo. obtains optimal soln to a problem by making a sequence of choices. At decision point the algo. make choice that seems best at that moment. This approach will not always give (always) the optimal soln. But in some case it will give you the optimal solution.

Knapsack Problem (0-1) :→ (leave / Pick)

There is a thief robbing a store & find n items there

Items	P_i [Profit]	Weight (w_i)	P_i/w_i
1	5	1	5
2	9	3	3
3	4	2	2
4	8	2	4

Bag capacity = 4 kg

Greedy does not give optimal for knapsack (0-1) but give optimal soln for knapsack (fractional).

$6 per pound | $5 per p | $4 per pound

Item 1	Item 2	$120
10 pounds	20 pounds	Item 3
$60	$100	30 pounds

W = 50 pounds

10 pou + 20 pou + 20 po

60 + 100 + 80

⇒ $240

(greedy)

20 + 30

100 + 120

⇒ $220

Date

Items	Pi	Wt.
1	5	1
4	8	2
2	9	3
3	4	2

(5) 4 3 (16)

→ 1
→ 2
3 kg.

Can not pick next weight item

because it is (0-1)

without = 14 (5+9)

with fractional = 16

Item	Wt.	Value	V/wt.
1	5	30	6
2	10	20	2
3	20	100	5
4	30	90	3
5	40	160	4

Max => 60 kg.

fractional

Item	Wt.	Value	V/wt.	
1	5	30	6	→ 5
3	20	100	5	→ 20
5	40	160	4	→ 40 35
4	30	90	3	→ 30 (60)
2	10	20	2	→ 5

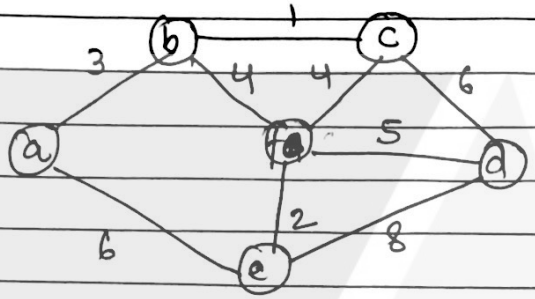
5+20+30
(1-0) 55 kg
=> 220

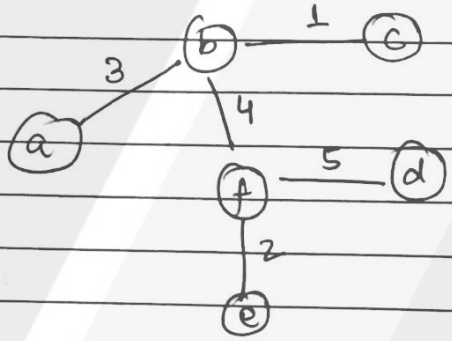
30+100+(4×35)
=> 130+140 => 270

Date

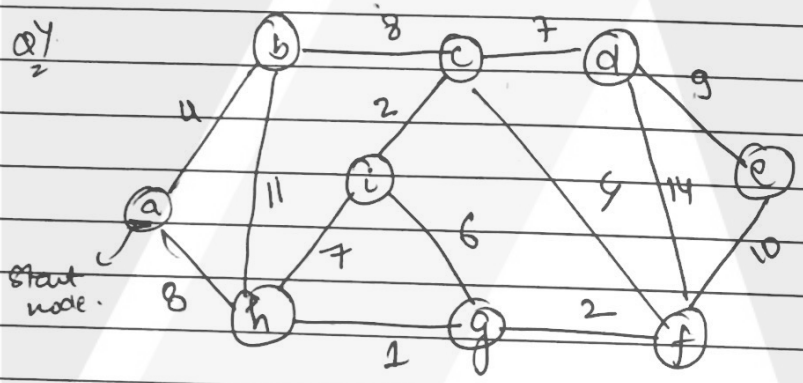
Minimum Spanning Tree :-

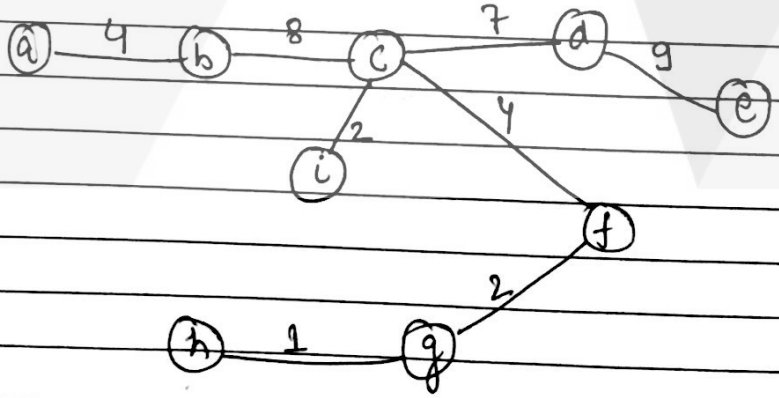
~~Prim's~~ Kruskal's ~~Algo~~ :- Prim's algo :-





Q)





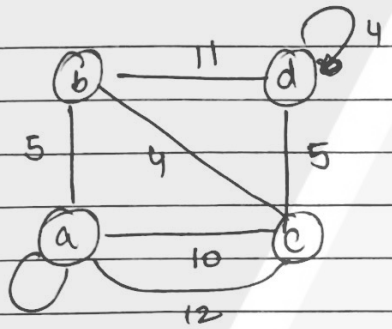
aminotes

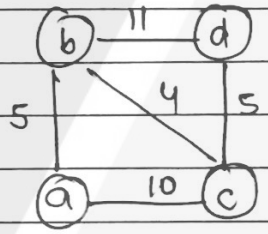
Date

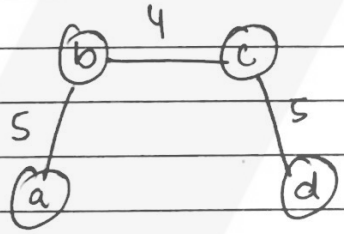
Prims algo work similary Dijkstra for finding shortest path.

Kruskal's Algo :-

loops - discard
parallel edge → minimum weight taken (path)

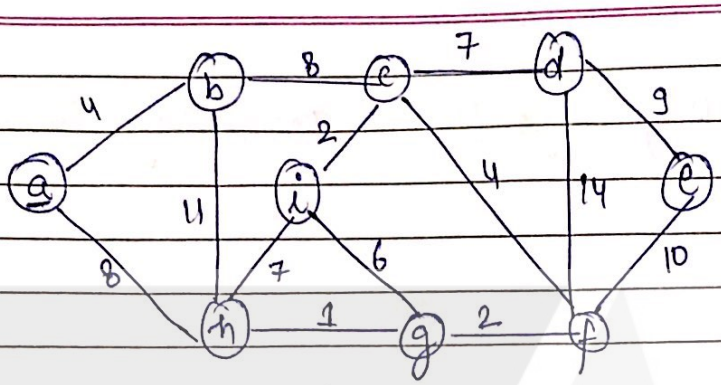


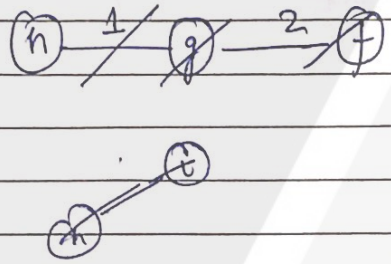


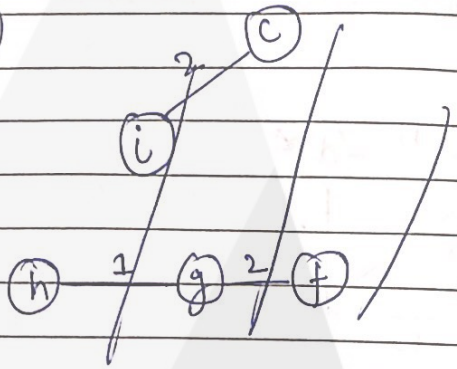


Date

Cormen
632





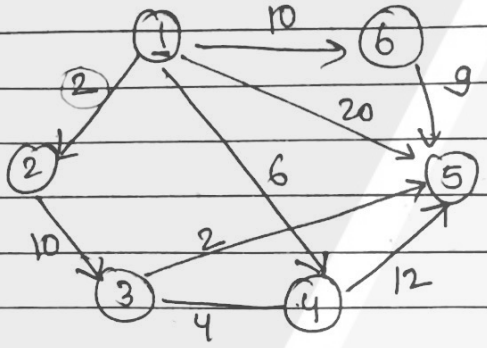


h ↔ g 1
g ↔ f 2
i ↔ c 2
a ↔ b 4
c ↔ f 4, i ↔ g 6
c ↔ d 7
h ↔ i 7
b ↔ c 8
a ↔ h 8
d ↔ e 9
c ↔ f 10
b ↔ h 11
d ↔ f 14

Date

Corman
658

Dijkstra algo:→



	1	2	3	4	5	6
1	—	2	∞	∞	20	10
{1,2}	2	2	12 (10+2)	6	20	10
start from one	↑ new added node					
{1,2,4}	4	2	10	6	18	10
{1,2,3,4}	3	2	10	6	12	10
{1,2,3,4,3,6}	6	2	10	6	12	10
{1,2,4,3,6,5}	5	2				

Date _____

05/03

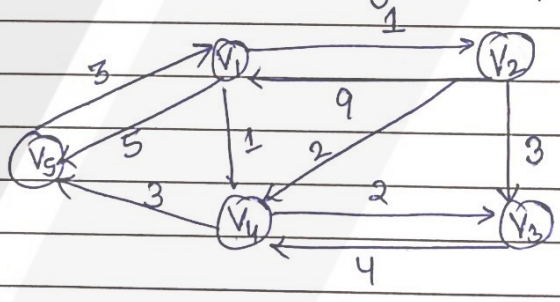
Dynamic Programming :-

Dynamic programming is a technique for solving problems with overlapping subproblems. It suggests solving each sub-problem once & storing the result in a table from which a soln to the original problem can be obtained.

It satisfies the principle of optimality. An optimal soln to any of its instances must be up of optimal soln to its sub-instances.

Problem solved with this
1). Matrix chain multiplication.
2). Floyd Warshal algo. (All pair shortest path algo.)
3). Travelling-salesman problem.
4). Knapsack problem.

2) Floyed Warshal algo. (All pair shortest path) :-



	1	2	3	4	5
1	0	1	∞	1	5
2	9	0	3	2	∞
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	∞	∞	∞	0

$= D^{(0)}$

All pair - Shortest Path [W[1---n], n)

① Initalize 'Matrix D with weight matrix W'.
② For k = 1 to n
③ For i = 1 to n
④ For j = 1 to n
⑤ D[i][j] = min {D[i,j], D[i][k] + D[k][j]}
⑥ end for
⑦ end for
⑧ end for.

$D^{(1)}$ =

	1	2	3	4	5
1	0	2	∞	1	5
2	9	0	3	2	14
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	4	∞	4	0

$D^{(2)}$ =>

	1	2	3	4	5
1	0	1	4	1	5
2	9	0	3	2	14
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	~~∞~~ 4	7	4	0

$D^{(3)}$ =>

	1	2	3	4	5
1	0	1	4	1	5
2	9	0	3	2	14
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	4	7	4	0

$D^{(4)}$ =

	1	2	3	4	5
1	0	1	3	1	4
2	4	0	3	2	5
3	∞	∞	0	4	7
4	∞	∞	2	0	3
5	3	4	6	4	0

$D^{(5)}$ =

	1	2	3	4	5
1	0	1	3	1	4
2	8	0	3	2	5
3	10	11	0	4	7
4	6	7	2	0	3
5	3	4	6	4	0

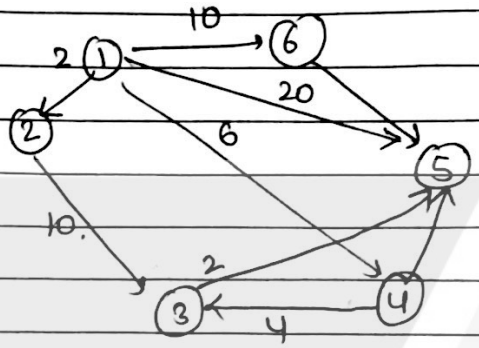
Complexity = $\theta(n^3)$

SHOE - BHANG - EE

HEMAN - SHOE

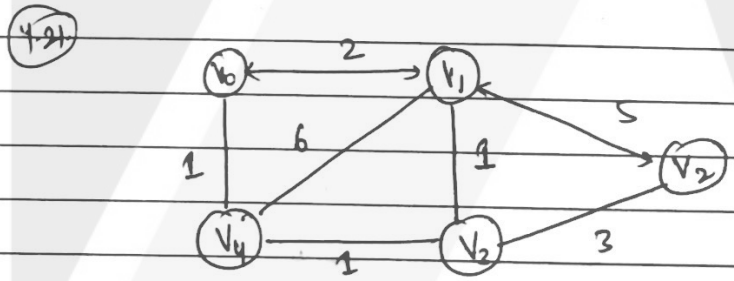
06/03

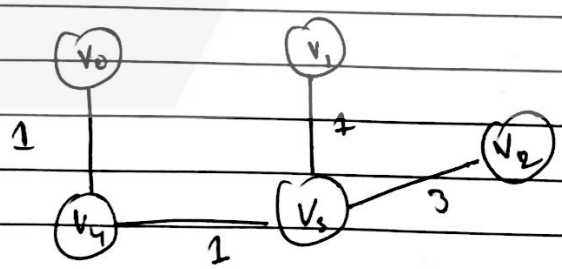
Date



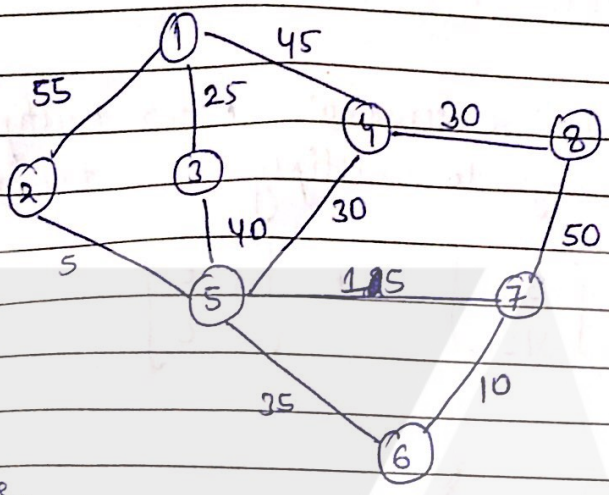
	1	2	3	4	5	6
1	0	2				
(X) 2		0				
3			0			
4				0		
5					0	
6						0

Dijkstra → done already.

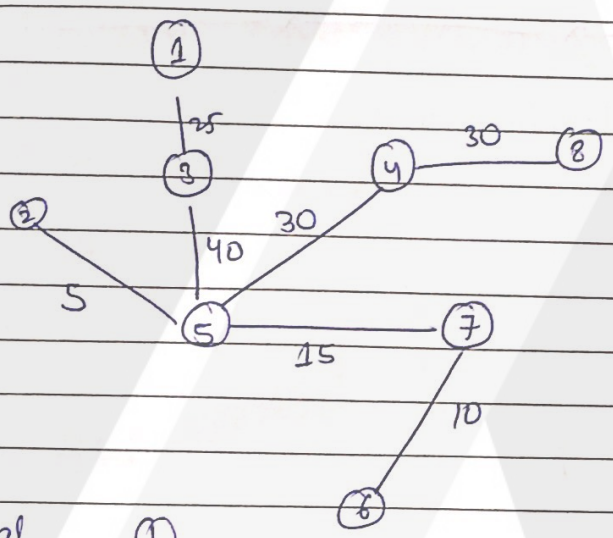




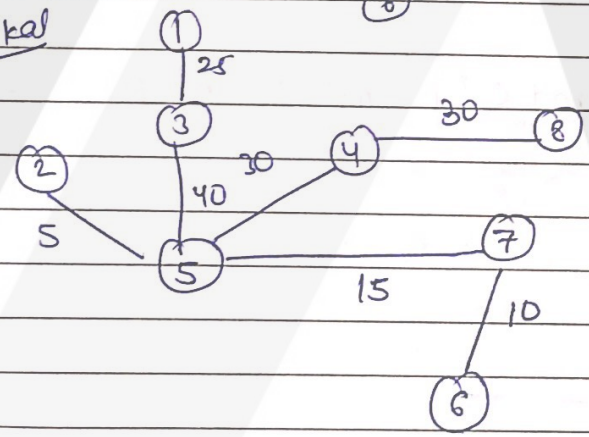
Date



Prims



Kruskal



① Find an optimal soln of Knapsack instance (7) ($m = 15$ wt.)
$(P_1 \dots P_7) = \{10, 5, 15, 7, 6, 18, 3\}$ and
$\{w_1 \dots w_7\} = \{2, 3, 5, 7, 1, 4, 1\}$

② Let $S = \{A, B, C, D, E, F, G\}$ be a collection of obj. with benefit values as follows A:(12,4) B(10,6) C(8,5) D(11,7) F(14,3) F(7,1) G(9,6). Optimal sol? for S assuming we have a knapsack that can hold obj. with the total wt. 18.

Q. Explain 0.1 knapsack prob. with ~~riddle~~ greedy method & show that it does not generate optimal soln.

Q. Show how quick-sort sorts the following seq. ~~in ascending order~~
22, 55, 33, 11, 99, 77, 55, 66, 54, 21, 32

p

10	2
5	3
15	5
7	7
6	1
18	4
3	1

Q4 Use Strassen's Matrix Multiplication also to multiply these two matrices

$A = \begin{bmatrix} 3 & 2 \\ 4 & 8 \end{bmatrix}$ $B = \begin{bmatrix} 1 & 5 \\ 9 & 6 \end{bmatrix}$

07/03

* Matrix - Chain Multiplication :→

$A_1 = 10 \times 100$
$A_2 \Rightarrow 100 \times 5$
$A_3 \Rightarrow 5 \times 50$

Matrix - Chain - Order (p)

1. $n = p.length - 1$
let $m[1 \ldots n, 1 \ldots n]$ and $s[1 \ldots n-1, 2 \ldots n]$ be the new table
for $i = 1$ to n
$m[i,i] = 0$
for $l = 2$ to n // l is the chain length
for $i = 1$ to $n-l+1$
$j = i+l-1$
$m[i,j] = \infty$
for $k = l$ to $j-1$
$q = m[i,k] + m[k+1,j] + p_{i-1} p_k p_j$
if $q < m[i,j]$
if $q < m[i,j]$
$m[i,j] = q$
$s[i,j] = k$.

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

Date

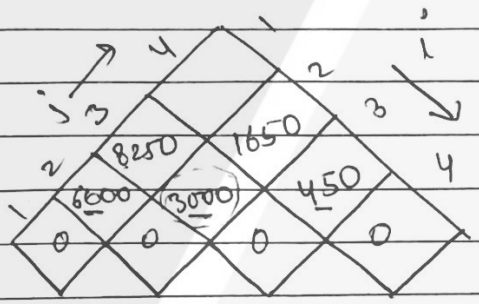
$A_0 = 30 \times 40$
$A_1 = 40 \times 5$
$A_2 = 5 \times 15$
$A_3 = 15 \times 6$

$p_0 = 30$ $p_1 = 40$ $p_2 = 5$
$p_3 = 15$
$p_4 = 6$

p_0 p_1 p_2 $0+450+$

$k = (i \text{ to } j-1)$
$(2 \text{ to } 2)$ (2)

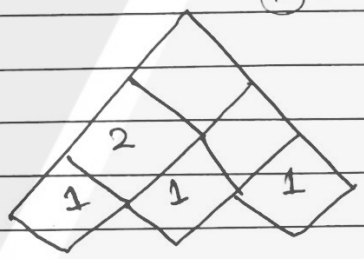
2
15
75
75
1
75
×3
2250



2, 3
$i=2, j=3, k=2$
$m[2,2] + m[3,3] + p_1 \cdot p_2 \cdot p_3$
$0 + 0 + 40 \times 5 \times 15$
$\Rightarrow 3000$

$0+3000+$

(m)



$k = 1, 2$ (1 to 2)
$i=1, j=3, k=1,2$
$m[1,1] + m[2,3] + p_0 p_1 p_3$
$\Rightarrow 0 + 3000 + 30 \times 40 \times 15$
$\Rightarrow 3000 + (1200 \times 15)$
$\Rightarrow 4,000.$

1200
×15
00

$k=2, i=1, j=3$
$m[1,2] + m[3,3] + p_0 p_2 p_3$
$\Rightarrow$ 8250

(1) min L·D + Max [R·D] + $p_{i-1} p_k p_j$
(2) max L·D + Min [R·D] + $p_{i-1} p_k p_j$
=

$6000 + 0 + 2250$
$\Rightarrow$ 8250

```
P.O.P.(S,1,6)
{ Print"("
  POP(S,1,3)
  POP(S,4,6)
  Print")"
}
```

$A_1 = 30 \times 35$
$A_2 = 35 \times 15$
$A_3 = 15 \times 5$
$A_4 = 5 \times 10$
$A_5 = 10 \times 20$
$A_6 = 20 \times 25$

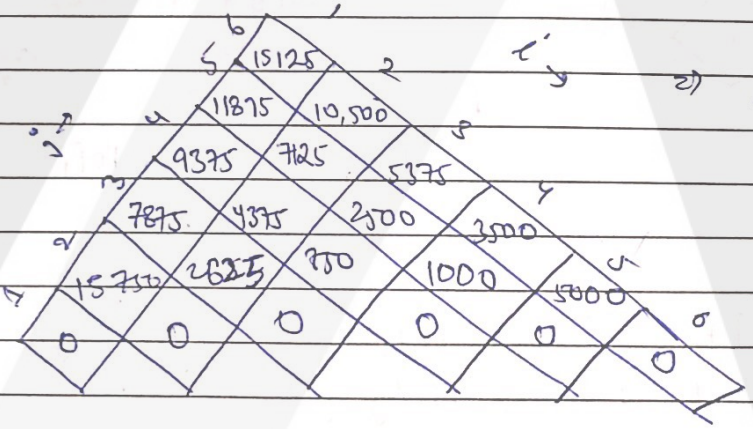
PRINT - OPTIMAL - PARENS (s, i, j)

if i == j
print "A"
else print "("
PRINT - OPTIMAL - PARENS (s, i, s[i, j])
(s, s[i, j] + 1, j)
print ")"

Q4.

$A_1 = 30 \times 35$
$A_2 = 35 \times 15$
$A_3 = 15 \times 5$
$A_4 = 5 \times 10$
$A_5 = 10 \times 20$
$A_6 = 20 \times 25$

P_0	P_1	P_2	P_3	P_4	P_5	P_6
30	35	15	5	10	20	25



⇒ (1,2)
i=1, j=2 k=(i to j-1)
(1 to 1)
k=1

m[1,1] + m[2,2] + $P_0 P_1 P_2$
⇒ 30 × 35 × 15
⇒ 15750

⇒ i=2, j=3, k=(2 to 2) (i to j-1)
m[2,2] + m[3,3] + $P_1 P_2 P_3$
0 + 0 + 35 × 15 × 5
⇒ 2625

$P_2 P_3 P_4$
⇒ 15 × 5 × 10
⇒ 750

⇒ $P_3 P_4 P_5$
⇒ 5 × 10 × 20
⇒ 1000

$P_4 P_5 P_6$
10 × 20 × 25
⇒ 5000

Date _____

~~0+2652+ 157~~
~~2625~~

$0 + 2625 + P_0 P_1 P_3$

~~2652~~ + 30 × 35 × 5

$\Rightarrow$ ~~2652~~ + 2625 + 5250

$\Rightarrow$ 7875

$\Rightarrow$ $0 + 750 + P_1 P_2 P_4$ (X)

$750 + 5250 \Rightarrow 6000$

$\Rightarrow$ 2625 +

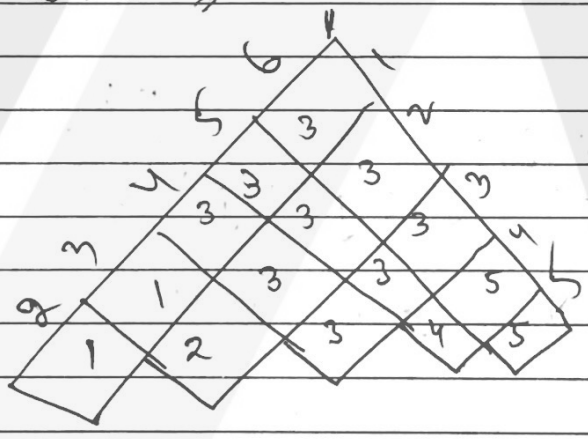
$(k=3)\ (i=1), (j=4)$

$m[1,3] + m[4,4] + P_0 P_3 P_4$

$\Rightarrow$ 7875 + 0 + 30 × 5 × 10

$\Rightarrow$ 7875 + 1500

$\Rightarrow$ 9375



2

35
~~25~~
~~175~~
× 3
525

35
× 35
1750

750

~~175~~ 525

$i=2, j=4 \quad k=(2,3)$

$k=3$

$m[2,3] + m[4,4] + P_1 P_3 P_4$

$2625 + 1750 + 0$

$\Rightarrow$ 4375 , 300

$\Rightarrow$

$k=2, i=2 \quad j=4$

$m[2,2] + m[3,4] + P_1 P_2 P_4$

$0 + 750 + 35 \times 15 \times 10$

$\Rightarrow$ 750 + 5250

$\Rightarrow$ 6000

P(1,4)

i: 1 K = (1 to 3)

j: 4

Longest common sub-sequence using dynamic programming :-

1) Characterize the optimal soln.

$X = \{x_1, x_2 \dots x_m\}$ & $Y = \{y_1, y_2 \dots y_n\}$ be sequences & $Z = (z_1, z_2 \dots z_k)$ be any LCS of X & Y.

a) If $x_m = y_n$, then $z_k = x_m = y_n$ and z_{k-1} is an LCS of x_{m-1} and y_{n-1}.

b). If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is an LCS of x_{m-1} and Y.

c). If $x_m \neq y_m$, then $z_k \neq y_m$ implies that Z is an LCS of X and y_{n-1}

2) Calculate the recursive soln :- (Recursive definition of optimal soln).

$$c[i,j] = \begin{cases} 0 & \text{if } i=0 \text{ or } j=0 \\ c[i-1,j-1]+1 & \text{if } i,j>0 \text{ and } x_i = y_j \\ \max(c[i,j-1], c[i-1,j]) & \text{if } i,j>0 \text{ \& } x_i \neq y_j \end{cases}$$

3). (Computing the value of optimal soln) :-

Algo. in book — In this case

for i = 1 to m (1 to 7)
for & for j = 1 to n
if $x_i == y_i$
$c[i,j] = c[i-1, j-1] + 1$
$b[i,j] =$ "↖"
else if $c[i-1,j] \geq c[i,j-1]$
$c[i,j] = c[i-1,j]$
$b[i,j] =$ "↑"
else $c[i,j] = c[i,j-1]$ $b[i,j] =$ "←"
return c and b

i ↓ / j →			B 0	B 1	D 2	C 3	A 4	B 5	A 6
	A	0	0	0	0	0	0	0	0
A	B	1	0	0↑	0↑	0↑	↖1	←1	↖1
(B)	C	2	0	↖1	←1	←1	1↑	↖2	←2
(C)	B	3	0	1↑	1↑	↖2	←2	2↑	2↑
(B)	D	4	0	↖1	1↑	(2↑)	2↑	↖3	←3
D	A	5	0	↑1	↖2	(↑2)	↑2	↑3	3↑
(A)		6	0	↑1	↑2	↑2	↖3	↑3	↖4
B		7	0	↑1	↑2	↑2	↑3	↖4	↑4

ABCB

BACB

X = (A, B, C, B, D, A, B) Y = (B, D, C, A, B, A)

Call Print- LCS (b, X, i, j)

Q4. X = {1, 0, 0, 1, 0, 1, 0, 1} Y = {0, 1, 0, 1, 1, 0, 1, 1, 0}

i ↓ / j →			0 0	1 1	2 0	3 1	4 1	5 0	6 1	7 1	8 0
	0	0	0	0	0	0	0	0	0	0	0
(1)	1	0	0↑	↖1	←1	↖1	↖1	←1	↖1	↖1	←1
0	2	0	↖1	1↑	↖2	←2	←2	↖2	←2	←2	↖2
(0)	3	0	↖1	1↑	↖2	2↑	2↑	↖3	←3	←3	↖3
(1)	4	0	↖1	↖2	2↑	↖3	↖3	3↑	↖4	↖4	←4
0	5	0	↖1	2↑	↖3	↖3↑	↖3↑	↖4	↑4	4↑	↖5
(1)	6	0	↖1↑	↖2	3↑	↖4	↖4	4↑	↖5	↖5	5↑
(0)	7	0	↖1	2↑	↖3	4↑	4↑	↖5	5↑	5↑	↖6
1	8	0	1↑	↖2	3↑	↖4	↖5	5↑	↖6	↖6	6↑

0 1 0 1 0 1

10

1 0 1 0 1 1

1 0 1 1 0 1

Date

TSP using Dynamic Programming (To decrease complexity)

G(V, E) be directed graph with edge cost c_{ij}

A tour of G is a directed simple cycle that includes every vertex in G.

→ Minimal cost tour = $n \to (n-1)!$

$4 \to 3! = 6$

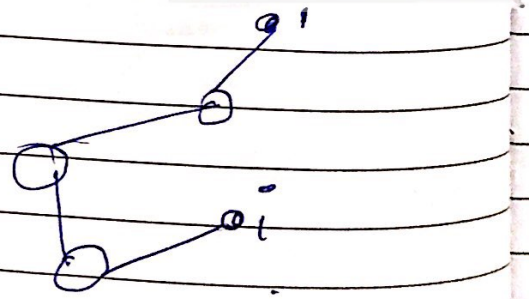
1-2-3-4-1 →
1-2-4-3-1 →
1-3-2-4-1 →
1-3-4-2-1 →

each tour consists of edge $\langle 1, k \rangle$ for some $k \in V-\{1\}$ & a path from vertex k to vertex 1

If tour is optimal then path from k to 1 through all the vertices $V-\{1,k\}$ should be shortest.

Principal of optimality

$g(i, S)$ = length of shortest path from node i & going through all vertices in S



$g(1, V-\{1\})$ is optimal length tour.

$$g(1, V-\{1\}) = \min_{2 \le k \le n} \{ c_{1k} + g(k, V-\{1,k\}) \}$$

for $i \notin S$.

$$g(i, S) = \min_{j \in S} \{ c_{ij} + g(j, S-\{j\}) \}$$

To solve

	1	2	3	4
1	0	10	15	20
2	5	0	9	10
3	6	13	0	12
4	8	8	9	0

$g(i, \phi) \Rightarrow c_{i1}$ $\quad 1 \le i \le n$

$g(i, S)$ is nodes are $i \neq 1$, $1 \notin S$, $i \notin S$

$g(2, \phi) \Rightarrow c_{21} = 5$

$g(3, \phi)$ $\quad c_{31} = 6$

$g(4, \phi) = c_{41} \Rightarrow 8$.

$g(4, 2) = 13$

$g(4, 3) = 15$

Now i have set size 1.

$g(2, \{3\}) \Rightarrow c_{23} + g(3, \phi) \Rightarrow 9 + 6$

$g(2, \{4\}) \Rightarrow c_{24} + g(4, \phi) \Rightarrow 10 + 8$

$g(3, \{2\}) = c_{32} + g(2, \phi) \Rightarrow 13 + 5 = 18$

$g(3, \{4\}) \Rightarrow c_{34} + g(4, \phi) = 20$

Date

Now i have set size of 2.

$g(2, \{3,4\}) = \min \{ c_{23} + g(3, \{4\}),\ c_{24} + g(4, \{3\}) \}$

$9 + 20 \Rightarrow 29$

$10 + 15 \Rightarrow 25$

$\Rightarrow 25$

$g(3, \{2,4\}) \Rightarrow 25$

$g(4, \{2,3\}) \Rightarrow 23$

$g(1, \{2,3,4\}) \Rightarrow \min \{ c_{12} + g(2, \{3,4\}),\ c_{13} + g(3, \{2,4\}),\ c_{14} + g(4, \{2,3\}) \}$

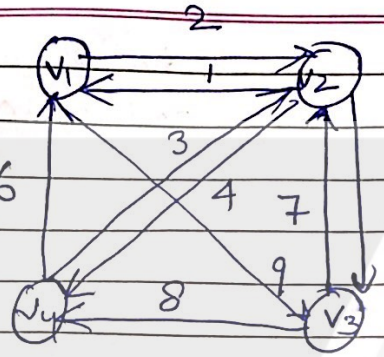
$\Rightarrow 35$

Recovery

$1 - 2 - 4 - 3 - 1$

TSP (V. V. Muniswamy) Pg - 137 - 141

Date



	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$n^2 2^n$

Time Complexity $n(2^n)$ Space

Knapsack 0-1 (using Dynamic)

Backtracking & Branch & Bound

↳ N Queen's

TSP

0-1 Knapsack.

= 35

= 40

= 43.

P, NP, NP complete

NP Hard

Branch & Bound :-
(Knapsack 0-1 problem)

- Breadth first search.
- depth first search.

$W = 12$

	v_i	w_i	v_i/w_i
1.	50	5	10
2.	48	8	6
3.	30	6	5
4.	12	4	3

Total wt :- $\sum_{i=1}^{k-1} w_i$

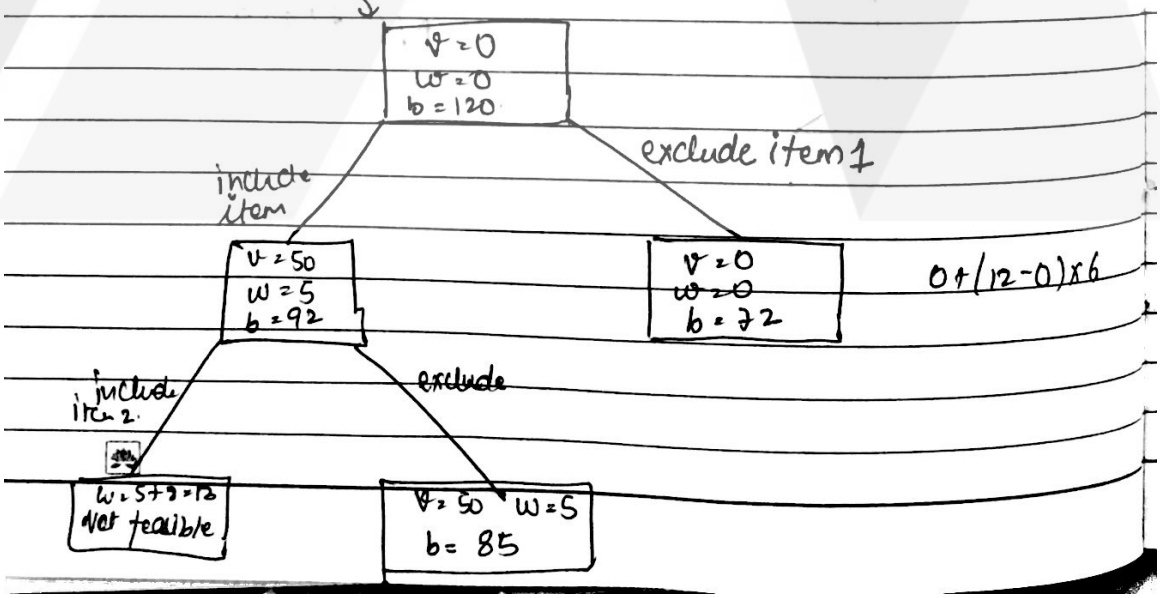
total value $v = \sum_{i=1}^{k-1} v_i$

Bound at i^{th} level.

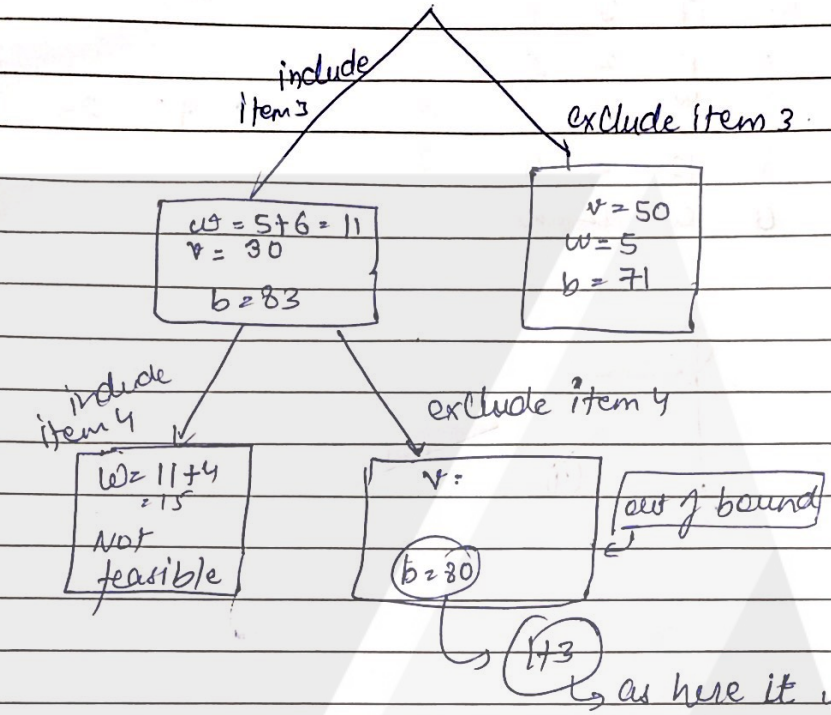
bound = $v + (w - w) \times (v_k / w_k)$ when $k = i+1$

for Root Node, $w = 0$ $v = 0$
For $i = 0$ (level at root)

$b = 0 + (12-0) \times 10 = Rs\ 120$



Date _____



pg-175 from V.V. Kunswami

TSP (using branch & bound method) :-

	1	2	3	4	5
1	∞	20	30	10	11
2	15	∞	16	4	2
3	3	5	∞	2	4
4	19	6	18	∞	3
5	16	4	7	16	∞

		1	2	3	4	5
10	1	∞	10	20	0	1
+2	2	13	∞	14	2	0
+2	3	1	2	∞	0	2
+3	4	16	3	15	∞	0
+4	5	12	0	3	12	∞
21						

∞	10	20	0	1
13	∞	14	2	0
1	3	∞	0	2
16	3	15	∞	0
12	0	3	12	∞

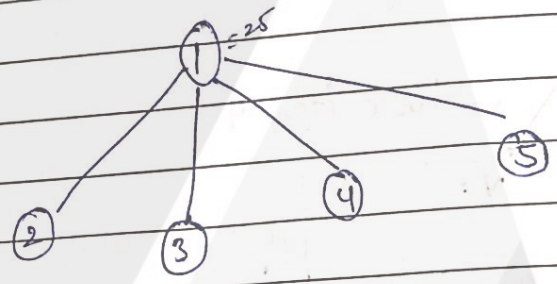
∞
12
0
15
11

Subtract minimal entry.

	∞	10	17	0	1
	12	∞	11	2	0
	0	3	∞	0	2
	15	3	12	∞	0
	11	0	0	12	∞

(A)

21 + 1 + 3 = 25



Let A be the reduced cost matrix for node R. Let S be the child of R such that the tree edge (R,S) corresponds to including edge (i,j) in the tour. If S is not a leaf then the reduced cost matrix for S may be obtained as follows:→

i) Change all the entries in Row i & column j of A to ∞. This prevents the use of any more edges leaving vertex i or entering vertex j.

ii) set A(j,1) to ∞, this prevents the use of edge(j,1).

iii) Reduce all the rows & columns in the resulting matrix except for rows & column containing ∞. If R is the total amount to be subtracted in step 3 then $\hat{c}(s) = \hat{c}(R) + A(i,j) + r$

Date

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	11	2	0
3	0	∞	∞	0	2
4	15	∞	12	∞	0
5	11	∞	0	12	∞

∞	∞	∞	∞	∞
∞	∞	11	2	0

cost of reduced matrix

$\hat{c}(R)$

(1→2)

$25 + 10 + 0$

$\Rightarrow 35$ (cost)

(1→3)

$25 + 17 + 0$

$\Rightarrow$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	∞	2	0
3	∞	3	∞	0	2
4	15	3	∞	∞	0
5	11	0	∞	12	∞

2) ~~$25 + 17 + 0 \Rightarrow 42$~~ $25 + 17$

Ist col.

∞	∞	∞	∞	∞
1	∞	∞	2	0
∞	3	∞	0	2
4	3	∞	∞	0
0	0	∞	12	∞

$25 + 17 + 11$

$\Rightarrow 53$

$(1 \to 4) \Rightarrow 25$

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	12	∞	11	∞	0
3	0	3	∞	∞	2
4	∞	3	12	∞	0
5	11	0	0	∞	∞

$\Rightarrow 25+0$

$\Rightarrow 25$

$(2 \to 1)$

∞	∞	17	0	1
∞	∞	∞	∞	∞
∞	3	∞	0	2
∞	3	12	∞	0
∞	0	0	12	∞

$(1 \to 5) \Rightarrow 31$

$(1 \to 4 \to 2)$

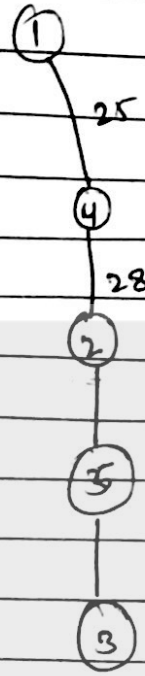
∞	∞	17	0	1
∞	∞	11	2	0
0	∞	∞	0	2
∞	∞	∞	∞	∞
11	∞	0	12	∞

	1	2	3	4	5
1	∞	∞	∞	∞	∞
2	∞	∞	11	∞	0
3	0	∞	∞	∞	2
4	∞	∞	∞	∞	∞
5	11	∞	0	∞	∞

$25 + 3 \Rightarrow 28$

$25 + 3$

$\Rightarrow 28.$



∞	10	∞	0	1
∞	1	∞	∞	∞
∞	2	∞	0	2
∞	3	∞	∞	0
11	0	∞	12	∞

(2-5)

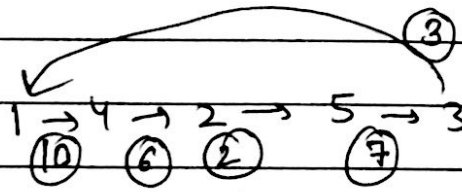
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
0	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	0	∞	∞

28+0+0 ⇒ 28

(5-3)

∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞
∞	∞	∞	∞	∞

~~28+2 ⇒ 30~~ 28+0 ⇒ 28.



⇒ (28)

Date

Assignment

Q4.

	1	2	3	4	5
1	∞	17	7	35	18
2	9	∞	5	14	19
3	29	24	∞	30	12
4	27	21	25	∞	48
5	15	16	28	18	∞

Knapsack (0-1) by dynamic programming :→

Item	Wt.	Values	Vi/Wi
1	1	1	1
2	3	4	1.3
3	4	5	1.25
4	5	7	1.4

Total weight = 7 kg.
so, no. of columns = 7+1 = 8.

Values	Wt.	0	1	2	3	4	5	6	7
1	1	0	1	1	1	1	1	1	1
					max(4 to, 1)				
4	3	0	1	1	4x	5 max(4+1, 1)	5	5	5
5	4	0	1	1	4	5	6	6	9
									↑
7	5	0	1	1	4	5	7	8	[9]

```
if (j < wt[i])
{
T[i][j] = T[i-1][j]
Else
{
```

Date

$$T[i][j] = \max \begin{cases} val[i] + T[i-1][j - wt[i]] \\ T[i-1][j] \end{cases}$$

bag: 4 3

value: 5 4

Q)

item	wt.	value	v_i/w_i
1	5	50	10
2	8	48	6
3	6	30	5
4	4	12	3

Tot. wt. = 12.

value	wt.	0	1	2	3	4	5	6	7	8	9	10	11	12
50	5	0	0	0	0	0	50	50	50	50	50	50	50	50
48	8	0	0	0	0	0	50	50	50 (48)	50	50	50	50	50
30	6	0	0	0	0	0	50	50	50	50	50	50	80	80
12	4	0	0	0	0	12	50 (12)	50	50	50	62	62	80	80

6 5

30 30