04.01.2023
Tuesday

Date: / / Page no:

ADA - Analysis & Design of Algorithms
= DAA - Design and Analysis of Algorithms

Algorithms : set of finite instruction to do task.

5 characteristics of Algo :-
1. i/p (0 or more i/p)
2. o/p (atleast one o/p)
3. finiteness (no. of instructions should be finite)
4. definateness (instruction should be unambiguous)
5. effectiveness (statement should be very basic/simple)

ex: Add 2 floating pt. no.
↳ ? How programmer knows this ?
→ Step by step

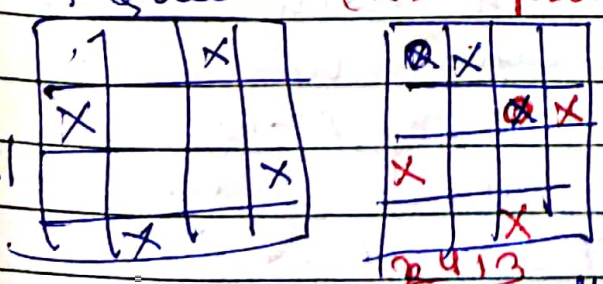
Part of Design Phase (Software life cycle)
- Analysis
- Design
- Coding (programmer)
- Testing

(No code)

→ Feasible soln (constraint solving)
→ Optimal soln (single source shortest path)

5 methods to design algorithms :-
1. Greedy (Prim's, Dijkstra's, Kruskal, Job sequencing deadline, Optimal Knapsack)
2. Dynamic Programming
3. Divide & Conquer (Quick sort, Merge Sort, Binary Search, Strassen's matrix multiplication)
4. Backtracking → (N) Queen's Problem, Puzzle (Hamiltonian Path), TSP
5. Branch & Bound 4&8 TSP → Knapsack

4 Queen (2 soln possible)



No 2 Queens on same row/ column / diagonal

Floyd Warshall algo (all pair shortest path)
of Knapsack, LCS (Longest common subsequence)
CMM (Chain Matrix Multiplication)
TSP (Travelling Salesman Problem)

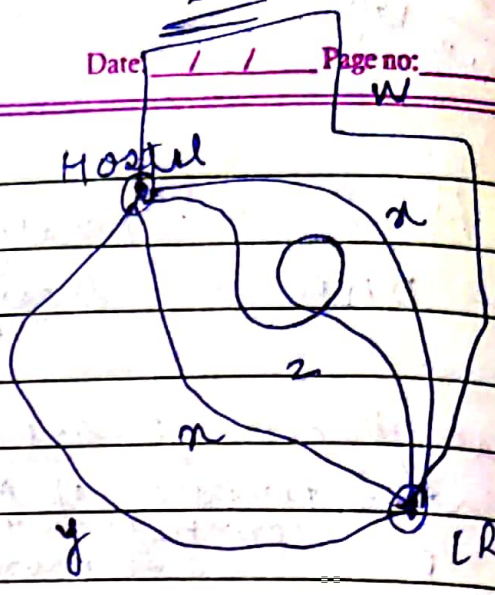
04.01.2023
Wednesday

DP

Date: / / Page no:

Greedy

Feasible Soln	Optimal Soln
"All possible Soln"	"Only one"
Spanning Tree (No cycle)	Minimum Spanning Tree.



Takes decision at runtime (multiple soln)

Branch Bound

The branch which does not leads to soln we bound that using function

Algorithms :-

Sequence of instruction unambiguous, precise (Paper & pencil)

to solve the task

Terminate in a finite amount of time.

for legitimate set of i/p set & o/p set

Select the ~~ambiguous~~ ~~conflict~~ color

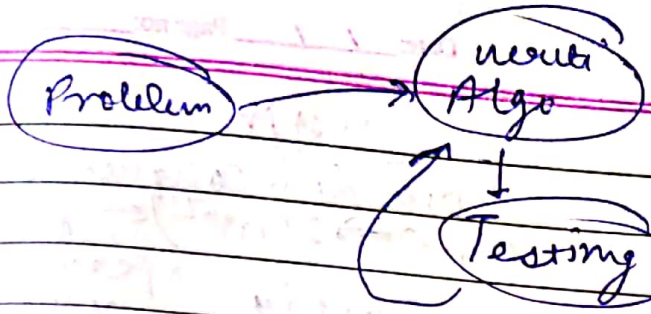
Ambiguous

Fundamental subject ; B in each & every Subject we require Algos.

Set of ~~instructions~~ Classical Algorithms related CS

Algorithm design techniques → used to model design for new

Date: / / Page no:



Process in industry.

Efficiency

Designing

Analysis → Efficiency practical feasible
Space & Time Complexity
(Main)

Time Complexity → Big O (Most standard)

push & pop operation in Stack → O(1)

Polynomial
$N, N^2, N^3, \log_2 N, N^2, N \log_2 N$

Non Polynomial
$O(2^N)$ exponential (grow very fast)
$O(N!)$
Not Practically feasible.
Hamiltonian cycle,
Combinational problems

Add^n of matrices = $O(N^2)$
Mult^n of matrices = $O(N^3)$

n^2 2^n
$n \uparrow$ This increase very fast
; Non Polynomial.

$n = 16$
16^2 $2^{16} = 1024 \times 2^6$

Solution
① Approximate solution
② Main steps of algo

Study
Problems Types → On the basis of chapter
Sorting →
Graph →

Study
Design Techniques → Design
Divide & Conquer
Greedy

09.03.2023 monday,

Date: / / Page no:

Problem types

- searching
- sorting
- Graph Algo → MST, Shortest Path
- String Processing (very less) LCS (6th sem)
- Combinatorial problems TSP (each city once & min-fare) → Generally exponential TC
- Geometric problem (Closest pair, Graph problem)
- Numerical Problem (gcd of 2 numbers) (Part Ackermann function)* complex

Hamiltonial cycle (decision making problem) → TSP (optimization problem)

Design Techniques

- Divide & Conquer — Strassen Matrix Multiplication: $O(n^3) \rightarrow O(n^{2.71})$
- Greedy → feasible → optimal
- Dynamic (dfs)
- Backtracking
- Branch & Bound (bfs) used for optimization & decision problem (maximization problem)

Dynamic: We use "principle of optimality" & subproblem must be optimal

1 — r_1 — r_2 — r_3 — K

suboptimal

Branch & Bound → discard unnecessary nodes (which will not lead to soln)

$T(n) = O(n)$ → linear

↳ function of Time complexity on i/p size as i/p

① Priori Analysis : O( ) (or Performance Analysis)
↳ Ⓟ Test (No Tension Programming language, compiler, system, Data structure, etc.)

② Posteriori Analysis : (or Performance measurement)
→ after program ~~run~~ is made & execute → actual statistics

Ex. Turnaround of program

Basic o/p
① Read
② Load
③ Write

④ Add/Subtr/ Multi/ Divi.

Date: / / Page no:

```
Algorithm Max( int a[], int n)
{   int maximum := a[1]                          1
    for (int i:=2; i ≤ n; i++)                   n
    {   if (a[i] > maximum)                      n-1
            maximum:= a[i];                      1 to n-1
    } // Reading
    return maximum;                              1
}                                                ---
                                                 3n
```

Priori Analysis : How to find

① Order of frequency of execution of each statement
② Adding the frequency of each statement to get the total no. of programe steps in the algo

$T(n) = C_n(Add) \times T_{add} + C_n(Sub) \times T_{sub} + \ldots\ldots + C_n(Load) \times T_{Load}$

$C_n(x)$ = ct of 'x' type operations
T_x = Time taken in single 'x' operation

Its tedious work

∴ We count Program step.

eg. for (int i=2; i<=n; i++)
① ② ③

Progm step = 1

Types of Statements

	No. of basic small operations	Programming steps
① Declaration stmts int i, j, k	0	0
② Simple Assignment stmts x = x + a[i]/4 - a[j];	1	1

x = x + Add(n) // Not Simple stmts
PS = 1 + PS(Add(n))

Date: / / Page no:

Types of Statements

PS (Program Steps)

③ Conditional Stmts

if (<condition>) then
    <Stmt1>
else
    <Stmt2>

true → 1 + PS(Stmt1)
false → 1 + PS(Stmt2)

* cond" is checked only once

④ Loops

(a) for (int i=1; i≤n; i++)

1st time	i=1 ; i≤n	1st PS	n+1
2nd time	i=i+1 ; i≤n	2nd PS	
⋮			
nth time	i=n ; i≤n	nth PS	
(n+1)th time	i=n+1 ; i≤n false	(n+1)th PS	

* for loop runs always one time more than value of i

(b) i=1; 1 PS
while(i≤n) { 1 PS n+1
i++ 1 PS
}

⑤ Function calls ;

x = Add(n)

// Not a Simple Assignment

1 + PS(Add(n))
↓ Assignment
↓ No of PS required to execute Add(n)

⑥ Return statement

comments

OPS

Date: / / Page no:

(a) return 1

(b) return (1 + f(n))

1 PS

1 + PS (Q)

return

No. of program steps read by the f^n f(n)

(I) Count Method

(i) Global variable count := 0;

(ii) Encounter a prog. step, increment count appropriately

(iii) Value of count at the end of algorithm → total no. of prog. steps.

ex Sum of elements of an array

```
Algorithm Sum (int a[], int n)
{
    int sum = 0;
    for (int i = 1; i <= n; i++)
        sum := sum + a[i];
    return (sum);
}
```

OPS (Declaration statement)

```
int count = 0;
count++; // Initializing sum var.
count++; // control part for loop
count++; // Simple Assignment
count++; // last increment of control part of for loop (i = n+1)
```

2n+3

count of return statement should be above return statement

∴ T(n) = 2n + 3

ex. Same in recursion

```
Algorithm RSum (int a[], int n)
{
    count++; // Cond. Evaluation
    if (n <= 0) { count++; return 0; }   // return stmt
    else { count++;                      // return stmt
           return [RSum(a, n-1) + a[n]); }
}
```

Date: / / Page no:

$$T(n) = \begin{cases} 2 & \text{if } n \le 0 \\ 2 + T(n-1) & \text{if } n > 0 \end{cases}$$

Recurrence Relation solving

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

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

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

$= 2 * i + T(n-i)$ (In general)

on the basis of $T(0) = 2$ $\quad i = n$ $\quad n <= 0$

$T(n) = 2 * n + T(0)$

$\Rightarrow \boxed{T(n) = 2n + 2}$

S/e Method (Steps per Execution Method)

```
Algebraic sum (int a[], int n)
1.{
2.      int sum = 0;
3.      for (int i=1; i<=n; i++)
4.  {a[i]++; sum := sum + a[i];}
5.      return sum;
6. }
```

S/e	frequency	Total no. of freq. prog. step contributed by each stmt
0	0	
1	1	$1 \times 1 = 1$
1	$n+1$	$1 \times (n+1) = n+1$
1 2	n	$1 \times n = n$
1	1	$1 \times 1 = 1$
		$2n+3$

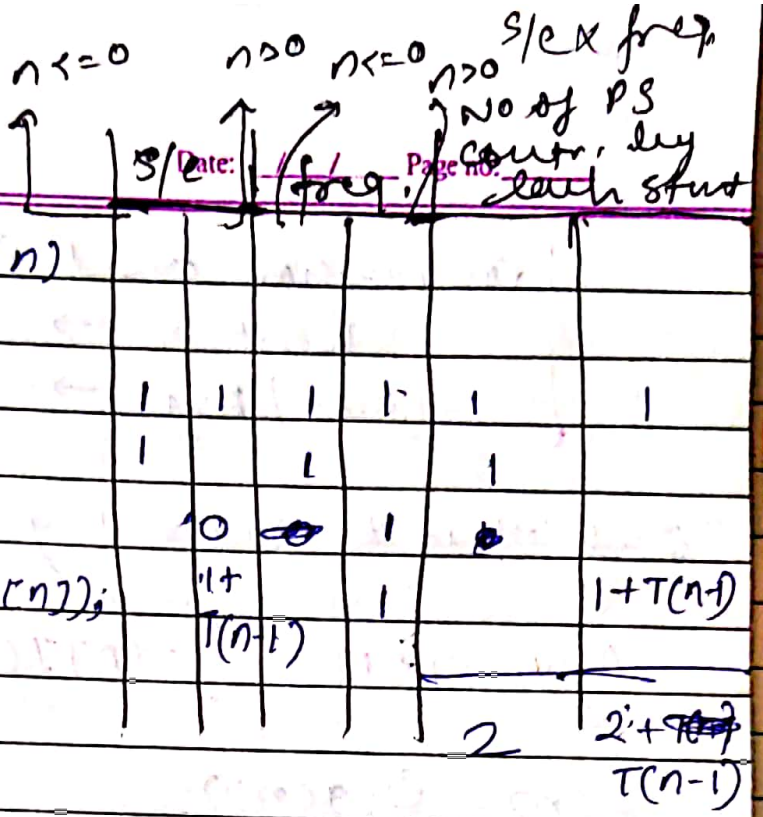
Total no. of PS contr. by each stmt = S/e × freq

Recursive

Algorithm Rsum (int a[], int n)

1.{

if (n <= 0)




```
1. Algorithm RSum(int a[], int n)
2. {
3.     if (n <= 0)
4.         return 0;
5.     else
6.         return(RSum(a,n-1) + a[n]);
7. }
```

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

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

$= 2*i + T(n-i)$ $\Rightarrow T(n) = 2 \times n + 2$

Quick Sort

Also for average

Best Case Time Complexity = $O(n \log n)$

Best Case Input = Random Array

↳ I/P for which algo takes least amt of time among all i/ps

Worst Case Time complexity = $O(n^2)$

Worst Case Input = Sorted / Reverse sorted array

↳ I/P ——— max. amt of time

'Any random i/p

Merge Sort

Best == Average == worst = $O(n \log n)$

Lower bound — upper bound of time taken (Time)

Helps to determine

Lower bound algorithm

Date: / / Page no:

Comparision Based Sorting Algo
Worst → $O(n^2)$
Best / Avg. → $O(n \log n)$

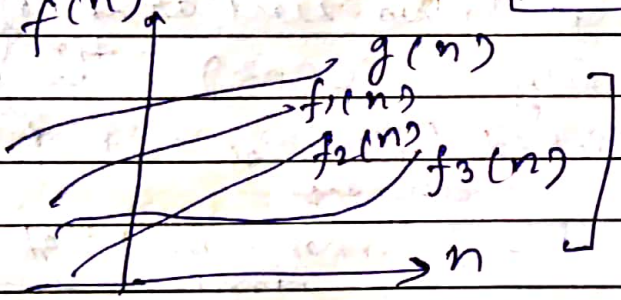
* Quick sort is the best sorting algo

ASYMPTOTIC NOTATIONS

$f(n) = O(g(n))$
or $f(n) \in O(g(n))$

↳ all functions $f_i(n)$ such that its growth rate is less than equal to constant times of $g(n)$ for $n \geq n_0$ (infinite set)

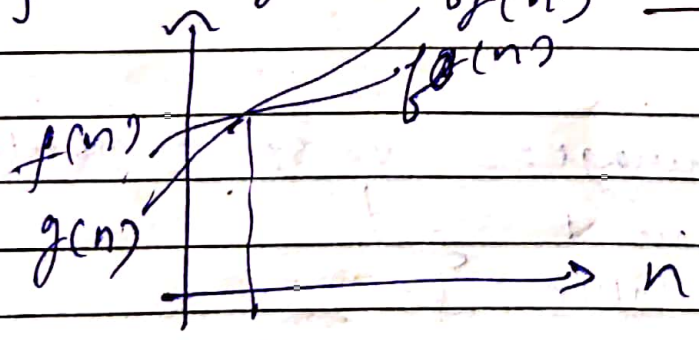
$T_1(n) = 100n + 20$
$T_2(n) = 2n^2 + 3$
$n = 1 \ldots k, \ldots n$
$T_1(n) > T_2(n)$ $T_1(n) < T_2(n)$



set.

for smaller n. (<no)
$f_i(n) \geq g(n)$

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



$g(n)$ → upper bound of function $f(n)$
(There can be infinite)
But we only need smallest among them

$O(g(n)) = \{ f(n) \mid f(n) \leq c \cdot g(n)$ for all $n \geq n_0$ and $c > 0 \}$

n_0 → can't be all fraction
c → can be fraction

Date: / / Page no:

O(g(n)), O-Notation (upper bound)

A function f(n) is said to be in O(g(n)) denoted as $f(n) \in O(g(n))$ if f(n) is bounded above by some constant multiple of g(n) for all large n ~~that~~ i.e. if there exists some true constant c and ~~non~~ some non negative integer n_0 such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$

(Worst case TC)

eg) $f(n) = 100n + 5$

$f(n) = O(n)$

$f(n) \leq 105n$ for all $n \geq 1$

$n=1$ $f(n) = 105 \leq 105$ ✓

$n=2$ ~~210~~ $205 \leq 210$ ✓

more meaningful $f(n) = O(n^2)$ ↓ But it is correct as per definition

Ω - Notation (lower bound) Ω(g(n))

A function f(n) is said to be in O(g(n)) denoted as $f(n) \in \Omega(g(n))$ if f(n) is bounded ~~above~~ below by some constant multiple of g(n) for all large n i.e., if there exists ~~some~~ a true constant c and a non negative integer n_0 such that $f(n) \geq c \cdot g(n)$ for all $n \geq n_0$

(Best case TC)

Ω(g(n)) = set of all functions f(n) that have a higher order of growth than g(n)

eg

$T(n) = 100n + 5$

$f(n) \geq 10n$ for all $n \geq 1$

$n=1$ $f(1) = 100 \cdot 1 + 5 \geq 10 \cdot 1$ ✓

$n=2$ $f(2) = 100 \cdot 2 + 5 \geq 10 \cdot 2$ ✓

→ constant in definition (g(n))

NOTE: Do not write constant in O(n), Ω(n) or Θ(n)

11.01.2023 wednesday

Date: / / Page no:

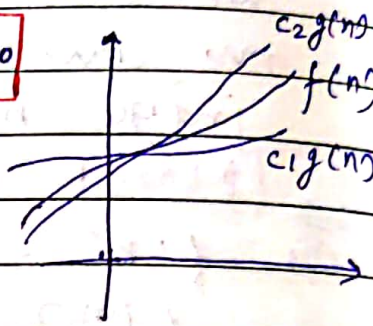
$\Theta(g(n))$, Θ-Notation

↳ $f(n)$ has same order of growth as $g(n)$ $\quad f(n) \in \Theta(g(n))$

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

iff there exists a two constants c_1 and a non negative constant c_2 integer no. such that

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \quad \text{for all } n \geq n_0$$



ex:

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

$c_1 = \frac{1}{4}$, $c_2 = 2$, $g(n) = n^2$, $n_0 = 1$

$$\frac{1}{4}(n^2) \leq f(n) \leq 2n^2$$

→ Implies ⇒ $\Omega(g(n))$

Implies $O(g(n))$

ex:

$f(n) = a_1 n^n + a_2 n^{n-1} + \cdots + a_1 n + a_0$

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

directly $\Omega(n^n)$ and $O(n^n)$ (proof in books)

Small oh - o (loose upper bound, not tight as O (Bigoh))

used to describe upper bound on function $f(n)$ that is not asymptotically tight.

$f(n) = 100n + 5 = O(n)$ or $O(n^2)$ ✓

$= o(n^2)$ $\quad f(n) \notin o(n)$ $\quad f(n) \in o(n^2)$

$f(n) =$

$o(g(n)) =$ ~~set of~~ iff ∃ there exist the integer c and non negative no. n_0

such that $f(n) < c \cdot g(n)$ [No equality]

Small ω (loose lower-bound)

$O : o :: \Omega : \omega$

Date: / / Page no:

Complexity Classes

Proof

① O(1) $f(n) = c$ $f(n) \le c \cdot g(n)$ O(1)
push pop in stack.

② O(log n) Binary search, search in Tree

③ O(n) ~~Insert~~ Search in list

④ O(n log n) Quick sort, Mergesort

⑤ $O(n^2)$ Matrix add^n,

⑥ $O(n^3)$ Matrix multiplication

⑦ $O(2^n)$ Combinatorial

⑧ O(n!)

Time Complexity increases ↓

Using limits to compare and Rank Time Complexity

$t(n)$ $g(n)$

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \begin{cases} 0 & t(n) \text{ has a lower order of growth as compared to } g(n) \\ c & t(n) \text{ has same order} \\ \infty & t(n) \text{ has higher order} \end{cases}$$

$t(n) = O(g(n))$
$t(n) = O(g(n))$ ← e
$t(n) = \Omega(g(n))$

L'Hospital's rule if we can't find directly.

$$\lim_{n\to\infty} \frac{t(n)}{g(n)} = \lim_{n\to\infty} \frac{t'(n)}{g'(n)}$$ → first diff wrt → ∞

Stirling's formula

$$n! \simeq \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$$

ex $t(n) = \frac{1}{2} n(n-1)$ $g(n) = n^2$

$$\frac{t(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{1}{2} n(n-1)}{n^2} = \frac{1}{2}\frac{(n-1)}{(n)} = \frac{1}{2}\lim_{n\to\infty}\left(1-\frac{1}{n}\right) = \frac{1}{2} = c$$

$\therefore t(n) = \Theta(g(n))$

$\Rightarrow t(n) = \Theta(n^2)$

GR(t(n)) = GR(Θ(g(n)) g(n))

order of Growth Rate

Fundamentals of Algorithms Horowitz and Sahni
↳ Common (Problem type) — Rajasekharan

Date: / / Page no:

Q) $t(n) = \log_2 n$ $\quad g(n) = \sqrt{n}$

Sol: $\lim_{n\to\infty} \frac{t(n)}{g(n)} = \lim_{n\to\infty} \frac{\log_2 n}{\sqrt{n}}$

L'Hospital's Rule:

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

$\lim_{n\to\infty} \frac{1}{\sqrt{n}} = 0$

$t(n) = O(g(n))$

$= O(\sqrt{n})$

Q) $t(n) = n!$ $\quad g(n) = 2^n$

Sol: $\lim_{n\to\infty} \frac{n!}{2^n} = \lim_{n\to\infty} \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{2^n}$ (Stirling's Formula)

$= \lim_{n\to\infty} \left(\frac{\sqrt{2\pi n}}{\infty}\left(\frac{n}{2e}\right)^{\infty}\right)^{\infty} = \infty \times \infty$

$= \infty$

$GR(t(n)) > = GR(g(n))$

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

Q) $(n-2)!$, $5\log(n+100)^{10}$, 2^{2^n}, $0.001n^4 + 3n^3 + 2$, $(\ln n)^2$, $\sqrt[3]{n}$, 3^n

↳ Natural log — Compare

Sol: $(n-2)! = O(n!)$

$5\log(n+100)^{10} = 50\log(n+100) = O(\log n)$

$2^{2^n} = 4^n = O(4^n)$

$0.001n^4 + 3n^3 + 2 = O(n^4)$

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

$\sqrt[3]{n} = O(n^{1/3})$

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

n	$\log n$	$n^{1/3}$	$\log^2 n$
1	0	1	0
8	2		4

$O(\log n) < O(n^{1/3}) < O(\log^2 n) < O(n^4) < O(3^n) < O(4^n) < O(n!)$

highest (const)^n

16/01/2023 Monday

$\frac{d}{dn}(a^n) = a^n \log a$ $\quad \frac{d(x^n)}{dn} = x^n(1+\ln x)$ $\quad \frac{d}{dn}(\log_a x) = \frac{1}{x} \times \frac{1}{\ln a}$

Revise AP, GP & HP

Date: / / Page no:

Series Θ, O Notation, Ω Notation

① $f(n) = 1+2+3+\ldots+n = \frac{n(n+1)}{2} = O(n^2)$

$1+2+3+\ldots+n \leq n+n+n^2+n\ldots+n$ times

$\leq n \cdot n \leq n^2 \quad g(n) = n^2$

$= \Theta O(n^2)$

$1+2+\ldots+\frac{n}{2}+\ldots n \geq \frac{n}{2}+\ldots+n$

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

$\frac{n(n+1)}{2} - \frac{n(n+1)}{8} \geq \frac{n^2}{4} + \frac{n}{4}$

$\frac{3n(n+1)}{8} \geq \frac{n^2}{4}$

$\geq \left(\frac{1}{4}\right) n^2 \hookrightarrow g(n)$

c

② $f(n) = \log(n!)$ → Heap Sort

~~→ Insertion of n elements in Heap~~

$= \log(n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1)$

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

$\leq \log n + \log n + \ldots \log n$ + .. n times

$\Rightarrow f(n) \leq n \log n$

$\Rightarrow \log(n!) \leq n \log n \longrightarrow g(n)$

$O(n \log n)$

$f(n) = \log(n!)$

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

$\geq \log n + \log(n-1) + \ldots + \log(\frac{n}{2})$

$\geq \log\frac{n}{2} + \log\frac{n}{2} + \ldots + \log(\frac{n}{2}) + \ldots \frac{n}{2}$ terms

$c_1 g(n) \le f(n) \le c_2 g(n)$

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

$\Omega(g(n))$ $O(g(n))$

O → n से बड़ा

Ω → Half series $\frac{n}{2}$ से छोटा

Date: / / Page no:

$\geq \frac{n}{2} \log \frac{n}{2}$ $\quad c = \frac{1}{2}$

$\geq \frac{n}{2} \log n - \frac{n}{2} \log 2$

$\geq \frac{n}{2} \log n$ $\quad c = \frac{1}{2}$ $\quad g(n) = n \log n$

$\Omega(n \log n)$

and also

$f(n) = O(n \log n)$ and $f(n) = \Omega(n \log n)$

$\therefore f(n) = \Theta(n \log n)$

HP

$$\sum_{i=1}^{n} \frac{1}{i} = O(\log n)$$

Result

$$\frac{1}{n} \le \log(n+1) - \log(n) \le \frac{2}{n}$$

Proof:

$$\sum_{i=1}^{n} \frac{1}{i} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}$$

$$\le \sum_{i=1}^{n} [\log(i+1) - \log i]$$

$\le$ $\log 2 - \log 1 +$ $\quad i=1$

$\log 3 - \log 2 +$ $\quad i=2$

$\log 4 - \log 3 +$ $\quad i=3$

$\vdots$

$\cdots +$

$\log(n+1) - \log n$ $\quad i=n$

$\le \log(n+1) - \log(1)$ $\quad \log(1) \to 0$

$\Rightarrow f(n) \le \log(n)$

$O(\log n)$

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

$\sum n^2 = \frac{1}{6} n(n+1)(2n+1)$

$\sum n^3 = \left[\frac{1}{2} n(n+1)\right]^2$

Date: / / Page no:

① for $i = 1$ to n

  for $j = 1$ to i

    $x = x + 1$

$O(n^2)$

$i=1$ $j=1$ times

$i=2$ $j=2$

$i=3$ $j=3$

$i=n$ $j=n$

$j = 1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$

② for $i = 1$ to n

  for $j = 1$ to i

    for $k = 1$ to j

      $x = x + 1$

$i=1$ $j=1$ $k=1$

$i=2$ $j=2$ $k=1+2$

$i=3$ $j=3$ $k=1+2+3$

$i=n$ $j=n$ $k=1+2+3+\cdots+n$

(a) $i_q = q$ $j_q = q$ $k_q = 1+2+\cdots+q$

$\frac{q(q+1)}{2}$

M.1.

Series Sum $= \sum_{q=1}^{n} k_q = \sum_{q=1}^{n} \frac{q^2}{2} + \sum_{q=1}^{n} \frac{q}{2}$

$= \frac{1}{2} \left[\frac{n(n+1)(2n+1)}{6}\right] + \frac{n(n+1)}{4}$ — neglect as $n^2 > n$

$= \frac{1}{12} (n^2+1)(2n+1)$

$= \frac{1}{12} (2n^3 + 2n + n^2 + 1)$

$= \frac{2n^3}{12} = \frac{n^3}{6}$ $\therefore O(n^3)$

M.2.

$f(n) = \sum_{i=1}^{n} \sum_{j=1}^{i} j = \sum_{i=1}^{n} \frac{i(i+1)}{2} = \sum_{i=1}^{n} \left(\frac{i^2}{2} + \frac{i}{2}\right)$

$= \frac{1}{2} \sum_{i=1}^{n} i^2 + \frac{1}{2} \sum_{i=1}^{n} i$

$= \frac{1}{2} \left\{\frac{n(n+1)(2n+1)}{6}\right\} + \frac{1}{2} \frac{n(n+1)}{2}$

$= O(n^3)$

Assume $n = 2^k$

Date: / / Page no:

③
```
for (i = n to 1
{ j = 1
  while (j > 1)
  { x = x + 1;
    j = j - 1; }
  i = i/2; }
```

	i	j
1st step	$i = 2^k$	$j = 2^k$
2nd step	$i = 2^{k-1}$	$j = 2^{k-1}$
3rd step	$i = 2^{k-2}$	$j = 2^{k-2}$
⋮	⋮	⋮
nth step	$i = 2^{k-k} = 2^0 = 1$	$j = 1$

Eq ②

$1 + 2 + \dots + 2^{k-1} + 2^k$

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

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

$= 2\, 2^k$

$= O(n)$

Outer loop

$n = 2^k$

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

$\Rightarrow k = \log_2 n$ times run

④
```
for i = n to 1
{ j = 1
  while (j > 1)
  { x = x + 1;
    j = j - 1; }
  i = i/3; }
```

Assume $n = 3^k$

⇒ outer loop run

series

$1 + 3 + \dots + 3^{k-1} + 3^k$

$= \frac{3}{2} \cdot 3^k - \frac{1}{2} = O(3^k)$

$= O(n)$

⑤ Prove $\sum_{i=0}^{k} \log \left(\frac{n}{2^i} \right) = \Theta(\log^2 n)$ assume $n = 2^k$

Sol: $\frac{n}{2^0} + \frac{n}{2^1} + \frac{n}{2^2} + \dots \frac{n}{2^n}$

$= n \left(\frac{1}{2^0} + \frac{1}{2} + \dots \frac{1}{2^n} \right) = n \left(\frac{1 - \frac{1}{2^{n+1}}}{1 - \frac{1}{2}} \right)$

Date: / / Page no:

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

$= \Theta n$

$= \log \frac{n}{2^0} + \log \frac{n}{2^1} + \log \frac{n}{2^2} + \cdots \log \frac{n}{2^k}$

$= \log \frac{2^k}{2^0} + \log \frac{2^k}{2^1} + \log \frac{2^k}{2^2} + \cdots \log \frac{2^k}{2^k}$

$= \log 2^k + \log 2^{k-1} + \log 2^{k-2} + \cdots \log 2^{k-k}$

$= k \log 2 + (k-1)\log 2 + (k-2)\log 2 + \cdots 0$

$= k + (k-1) + (k-2) + \cdots 1 + 0 \quad [\because \log_2 2 = 1]$

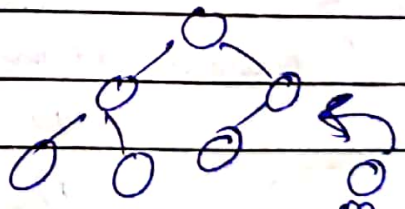
$= \frac{k(k+1)}{2} = O(k^2)$

$= O(\log^2 n) \quad [\because k = \log n \text{ as } n = 2^k]$

~~HeapSort~~

Insert in a Heap.

```
for (i = 1 to n)
        Heapify (A, item)
```



Max Heapify for $\log n$ is Height

$= \log 1 + \log 2 + \log 3 + \cdots + \log(n-1) + \log n$

$= \log(1 \times 2 \times 3 \times \cdots \times (n-1)\, n)$

$= \log(n!)$

$= \Theta(n \log n)$

17-01-2023 Tuesday.

Date: / / Page no: 10

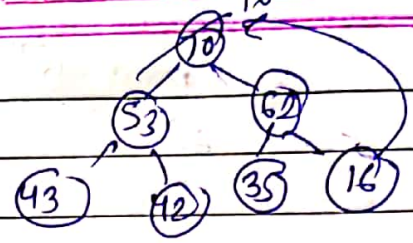
• Delete the element from Heap

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

$= \log(n \cdot (n-1) \cdot (n-2) \cdots 2 \cdot 1)$

$= \log(n!)$

$= O(n \log n)$



Always delete Root & replace with Leaf (from right.) & then heapify

Design Technique

1. Brute force — Sort 'n' numbers : $n! \rightarrow N$ (All possible soln) → Y
2. Divide & conquer — Divide, solve, merge (Independent subproblems)
3. Dynamic Programming (Memoization) → Overlapping of problems
   (i) Overlapping subproblems
   (ii) Optimal substructure
4. Greedy Approach
5. Backtracking
6. Branch & Bound
7. Parallel Algorithms
8. Approximation Algo.
9. Quantum Algo.
10. Transform Algos. (Linear programming)

Date: / / Page no:

Divide & Conquer Algo

Algo DAndC(P) → solve

if small(P) then return S(P);

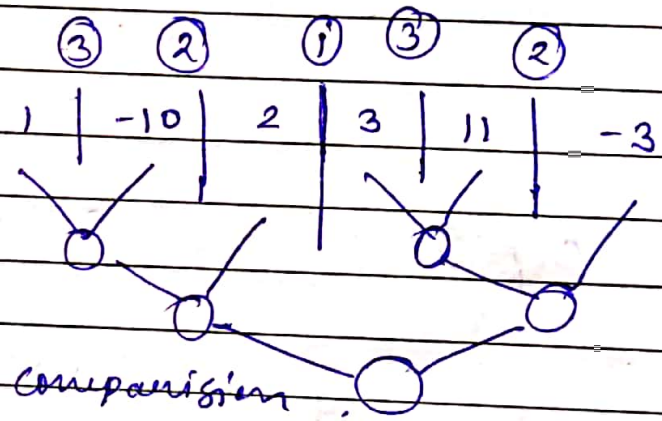
else

Divide P into smaller subproblems $P_1, P_2, \ldots P_k$ $k \geq 1$

return combine(DAndC(P_1), DAndC(P_2), ..., DAndC(P_k));

MERGE SORT

Significant ~~operations~~ comparision



last ~~to~~ divide is the first merge

Iterative Method

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

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

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

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

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

General

$$= 2^k T(\tfrac{n}{2^k}) + kn$$

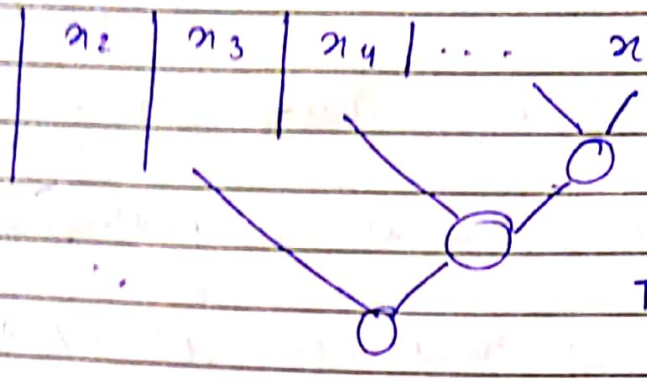
$n = 2^k$ $\quad \frac{n}{2^k} = 1$

$$= 0 + kn$$

$$= n \log_2 n$$

(5) Date: / / Page no:

- little change in Divide & merge



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

$O(n^2)$

$$T(n) = \begin{cases} 0, & n=1 \\ T\left(\frac{n}{3}\right)+T\left(\frac{2}{3}n\right)+n, & n>1 \end{cases}$$

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

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

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

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

$O(n \log_3 n)$

Page no:

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

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

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

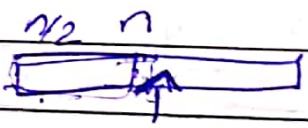
$T(\frac{n}{3})$

$T(n)$

$T(\frac{2n}{3}) + \frac{2n}{3}$ → Ano 1

$T(\frac{n}{3}) + (\frac{n}{3})$ → Ano 2.

$$T(n) = \begin{cases} 0, & n=1 \\ T(\frac{n}{4}) + T(\frac{2n}{4}) + n, & n>1 \end{cases}$$



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

$O(1+1)+1)$

n^2

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

Date: / / Page no:

comp.

Q) $T(n) \begin{cases} 1, & n=1 \\ T\left(\frac{n}{2}\right)+1, & n>1 \end{cases}$

] Binary search (Can be similar to other algo

Sol:

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

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

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

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

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

General

$T\left(\frac{n}{2^k}\right) + k$ — (1)

$= 1 + k = 1 + \log_2 n$

Binary search $O(\log_2 n)$

Q) $T(n) \begin{cases} 1, & n=1 \\ 1 + T\left(\frac{2}{3}n\right), & n>1 \end{cases}$

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

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

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

$= T(1) + k$

$= 1 + \log_{\frac{3}{2}} n$

$\frac{2^k}{3^k} n = 1$

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

$\log_{\frac{3}{2}} n = k$

$O(n) = \log_{\frac{3}{2}} n$

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

18.01.2023 Wednesday

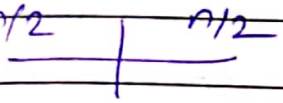
Date: / / Page no:

RECURRENCE RELATION

$T(n) = a\,T(n/b) + f(n)$

- a → No. of subproblems
- n/b → Size of subproblems
- $f(n)$ → time to divide + time to conquer

① Divide & Conquer → Merge sort



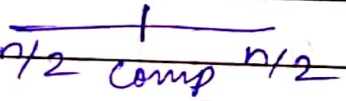
divide = $O(1)$

merge (conquer) = $O(n)$

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

$\therefore f(n) = cn$

② Binary Search.



Divide = $O(1)$

Compare (conquer) = $O(1)$

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

$f(n) = O(1) = c$

* Iterative substitution Method. (Brute Force)

Remember to look for pattern, don't add terms

$T(n) = T(n-1) + 2$, $T(0) = 2$

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

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

In General, $T(n) = T(n-i) + 2 \cdot i$

$[n - i = 0,\ i = n]$ ↑ $T(n) = T(0) + 2 \cdot n$

$= 2 + 2n = O(n)$

formula

② Substitution method (Practise / Experience)

Guess a soln for a recurrence rel^n & we try to prove our guess by using Mathematical Induction

Date: / / Page no:

$T(n) = 2T(n/2) + an$ → (1)

Guess: $T(n) = O(n \log n)$

$\Rightarrow T(n) \leq c\, n \log n$

Let us assume that this holds true for $n/2$

$T(\frac{n}{2}) \leq c\frac{n}{2}\log\frac{n}{2}$

put in RHS of (1)

$T(n) \leq 2 \times c\frac{n}{2}\log\frac{n}{2} + an$

$\leq cn\log\frac{n}{2} + an$ (2)

$\leq cn\log_2 n - cn\log 2 + an$

$\Rightarrow T(n) \leq cn\log_2 n - cn + an$

$\leq cn\log_2 n + (a-c)n$ → $a-c < a$

$\leq cn\log_2 n + c'n$

n_0

c

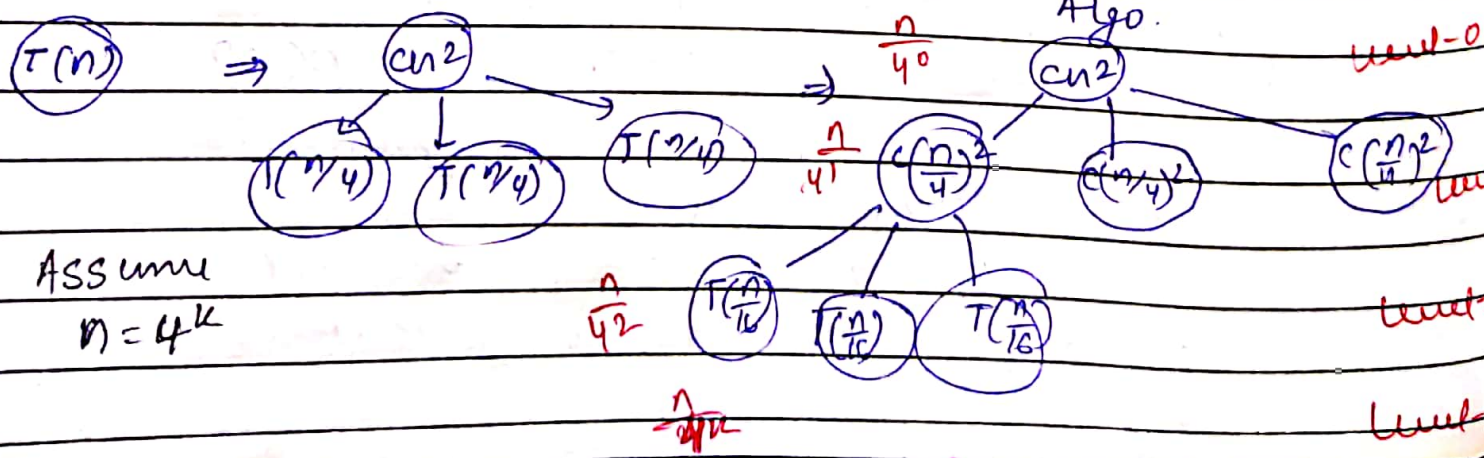
$[a-c<a]$ (guess)

$\therefore$ Our assumption is correct

$T(n) \leq c\, n\log n$

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

③ Recursion Tree Method :- Mainly used to solve $T(n) = 3T(\lfloor n/4 \rfloor) + cn^2$, $T(1) = a$ for Divide & Conquer Algo.



Assume $n = 4^k$

Scanned with CamScanner

Date: / / Page no:

per level cost = No. of nodes × cost of each node at each level. (in level)

Level-0	$3^0 \times c\left(\frac{n}{4^0}\right)^2$	
Level-1	$3^1 \times c\left(\frac{n}{4^1}\right)^2$	
Level 2	$3^2 \times c\left(\frac{n}{4^2}\right)^2$	
⋮		
Level i	$3^i \times c\left(\frac{n}{4^i}\right)^2$	[General level]
⋮		
Analysis upto $(k-1)^{th}$ level	$3^{k-1} \times c\left(\frac{n}{4^{k-1}}\right)^2$	Internal nodes
k^{th} level	$3^k \times T(1) = 3^k \times a$	

↳ No. of nodes per level.

Total cost = Σ per level cost

$$T(n) = 3^0 \times c\left(\frac{n}{4^0}\right)^2 + 3^1 c\left(\frac{n}{4}\right)^2 + 3^2 c\left(\frac{n}{4^2}\right)^2 + \cdots 3^{k-1} c\left(\frac{n}{4^{k-1}}\right)^2 + 3^k T(1)$$

$$= cn^2 \left[\frac{3^0}{(4^0)^2} + \frac{3^1}{(4^1)^2} + \frac{3^2}{(4^2)^2} + \cdots + \frac{3^{k-1}}{(4^{k-1})^2}\right] + 3^k T(1)$$

$$= cn^2 \left[\frac{3^0}{(4^2)^0} + \frac{3^1}{(4^2)^1} + \frac{3^2}{(4^2)^2} + \cdots \frac{3^{k-1}}{(4^2)^{k-1}}\right] + 3^k T(1)$$

$$= cn^2 \left[\frac{1\left(1-\left(\frac{3}{16}\right)^k\right)}{\left(1-\frac{3}{16}\right)}\right] + 3^k T(1) \quad (|r|<1)$$

$$= \frac{cn^2}{\left(\frac{13}{16}\right)}\left(1-\frac{3^k}{16^k}\right) + 3^k T(1)$$

23.01.2023 Monday.

Date: / / Page no:

$$\Rightarrow T(n) = \frac{cn^2}{\frac{13}{16}}\left(1 - \frac{n^{\log_4 3}}{n^2}\right) + n^{\log_4 3} T(1)$$

$$= c'n^2 - c'n^2 n^{\log_4 3} + n^{\log_4 a}$$

$$= c'n^2 \cdot 0 - c' n^{\log_4 3} + n^{\log_4 a}$$

$$= c'n^2 + (a - c') n^{\log_4 3}$$

$$= c'n^2 + c'' n^{\log_4 3}$$

$$T(n) \leq c'n^2$$

$\therefore T(n) = O(n^2)$

$k = \log_4 n$

$16^k = 16^{\log_4 n}$

$= n^2$

$n^{\log_b a} = a^{\log_b n}$

$3^k = 3^{\log_4 n}$

$= n^{\log_4 3}$

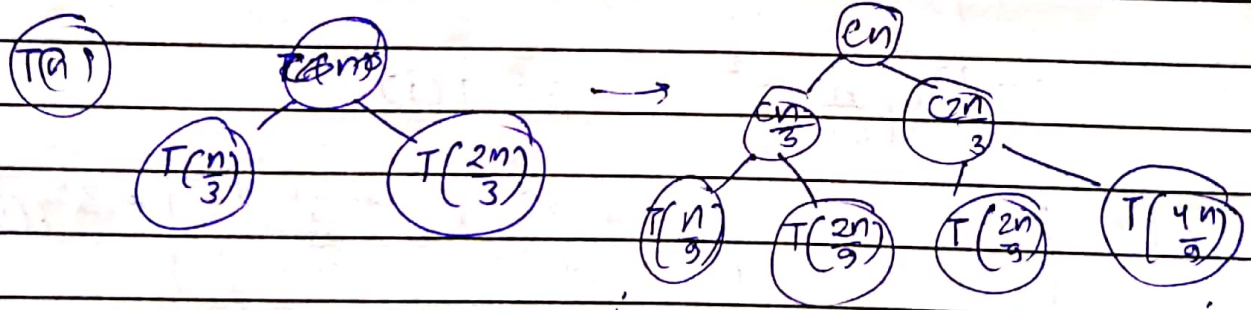
$[\log_4 3 < 1]$

$[n^2 > n^{\log_4 3}]$

Slopiness (Error) in answer as $\lfloor \frac{n}{4} \rfloor$ → have not considered.

It's easy as there is complete tree.

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



would end soon

Appox. in soln

$cn \cdot \log_{3/2} n$ (although not tight)

$\Rightarrow T(n) = O(n \log_{3/2} n)$

Main height this

n

$n/(3/2)$

$n/(3/2)^2$

$n/(3/2)^k$

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

Then we can use Substitution method to verify

Date: / / Page no:

MASTER'S THEOREM

Prove it using Recursion tree method

$T(n) = a\,T(n/b) + f(n)$

(When we draw Recursion tree we get 2 terms)

Compare $f(n)$ & $n^{\log_b a}$

Case 1: $f(n)$ has a lower order of growth than $n^{\log_b a}$ by a factor n^{ϵ} $\epsilon > 0$ (polynomially)

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

Case 2: $f(n)$ has same order of growth than $n^{\log_b a}$

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

Case 3: $f(n)$ has higher order of growth than $n^{\log_b a}$ by a factor of n^{ϵ} $\epsilon > 0$ (polynomially greater)

$T(n) = O(f(n))$

check only in Case 3 → regularity cond^n i.e., $a f(n/b) \le c \cdot f(n)$ for some constant $c > 0$

ex. (1) $T(n) = 4T(n/2) + n$

$a = 4$, $b = 2$ $\quad f(n) = n$ $\quad n^{\log_b a} = n^{\log_2 4} = n^2$

(polynomially larger)

Clearly $f(n) < n^{\log_b a}$ $\quad \frac{n}{n^2} = \frac{1}{n}$

$\therefore T(n) = O(n^2)$

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

$f(n) = n^{\log_b a} = n^2$ $\quad T(n) = \Theta(n^2 \log_2 a)$

$= \Theta(n^2 \log_2 n)$

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

$a = 4$, $b = 2$ $\quad f(n) = n^3$ $\quad n^{\log_b a} = n^{\log_2 4} = n^2$

$n^3 > n^2$ $\quad T(n) = \Theta(f(n)) = O(n^3)$

Regularity cond^n $\quad a f(n/b) \le c f(n) \Rightarrow 4\left(\frac{n}{2}\right)^3 \le c n^3 = c \ge \frac{1}{2}$

Date: / / Page no:

Q① $T(n) = 4T(n/2) + n^2 \log n$

$a = 4, \quad b = 2$

$f(n) = n^2 \log n$ $\quad n^{\log_b a} = n^{\log_2 4} = n^2$

$f(n) > n^{\log_b a}$ by factor $\log n$ which is not polynomially greater (Logarithmically greater)

∴ master's theorem can't be applied.

Q② $T(n) = 3T(n/4) + n \log n$

$f(n) = n \log n$ $\quad a = 3 \quad b = 4$ $\quad n^{\log_b a} = n^{\log_4 3} = n^{0.793}$

$f(n) > n^{\log_b a}$ by the factor $(n^{1-0.793}) \log n$

polynomial term. $\quad$ even $\log n$

first guarantees term

Regularity condⁿ

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

$\Rightarrow 3(\frac{n}{4} \log(\frac{n}{4})) \leq c n \log n$ $\quad n^{\epsilon}$ $\epsilon > 0$

$\Rightarrow \frac{3}{4} n \log n - \frac{3}{4} n \log 4 \leq c \cdot n \log n$

$\Rightarrow \boxed{c \geq \frac{3}{4}}$

$T(n) = O(f(n))$

$= O(n \log n)$

Q③ $T(n) = 2T(\sqrt{n}) + \log n$ ~~$T(n) = \sqrt{n} T(\sqrt{n}) + n$~~

Sol: Assume $n = 2^m \Rightarrow m = \log_2 n$

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

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

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

$a = 2, b = 2$ $\quad f(m) = m$ $\quad m^{\log_b a} = m$ $\quad m \log m$

Same order $\quad S(m) = O(f(m) \cdot \frac{\log m}{\log b}) = O(m \log m)$

Scanned with CamScanner

Date: / / Page no:

If $S(m) = \Theta(m \log_2 m)$ $\quad n = 2^m$

If $T(2^m) = \Theta(m \log_2 m)$ $\quad m = \log_2 n$

$\Rightarrow T(n) = \Theta(\log_2 n \log_2 \log_2 n)$

(Q4) $T(n) = 2T(\sqrt{n}) + n$

Sol: Assume $n = 2^m \Rightarrow m = \log_2 n$

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

Let $T(2^m) = S(m)$ $\quad S(m) = 2S(m/2) + 2^m$

$S(m) = 2S(m/2) + 2^m$ → further Subst^n not can calcul err

By Brut Fru:

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

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

In General

$S(m) = 2^i S\left(\frac{m}{2^i}\right) + 2^{m/i-1} + \cdots + 2^{m/2} + 2^{2^0}$

$= 2^i S\left(\frac{m}{2^i}\right) + \left(K^{1/2^0} + K^{1/2} + \cdots K^{1/2^i}\right)$ Let $2^m = K$

GP

$= 2^i S\left(\frac{m}{2^i}\right) + K\left(K\right)$

Assume $m = 2^i$

(22) $S(m) = m\left(S\left(\frac{2^i}{2^i}\right)\right) + 2^{2^i/(2^{i-1})} + 2^{2^i/2^{i-2}} + \cdots 2^{2^i/2^0}$

so that base case →

$\Rightarrow S(m) = mS(1) + (2^{2^1} + 2^{2^2} + \cdots 2^{2^{i-1}} + 2^{2^i})$

$= mS(1) + (4^1 + 4^2 + \cdots 4^{i-1} + 4^i)$

Date: / / Page no:

Q(5) $T(n) = 2T\left(\frac{n}{2}\right) + bn\log n$

Sol: $A = 1 \quad B = 2$

$f(n) = bn\log n$

$n^{\log_B A} = n^{\log_2 1} = n^0 = 1$

$f(n)$ has higher growth rate. $n^{\log_B A}$ (Polynomial)

$T(n) = O(n\log n)$ Master's theorem

Regularity condn

$af\left(\frac{n}{B}\right) \le cf(n)$

$\Rightarrow \frac{b}{2} n\log n \le cn\log n$

$\Rightarrow c > \frac{b}{2}$

$\frac{b}{2}\frac{n}{2}\log\frac{n}{2} \le cn\log n$

$\frac{b}{2}(n\log n - n\log 2) \le bcn\log n$

$\Rightarrow \boxed{c \ge \frac{1}{2}}$

Iteration Substn

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

$= \left[T\left(\frac{n}{2^2}\right) + b\frac{n}{2}\log\frac{n}{2} + bn\log n\right)$

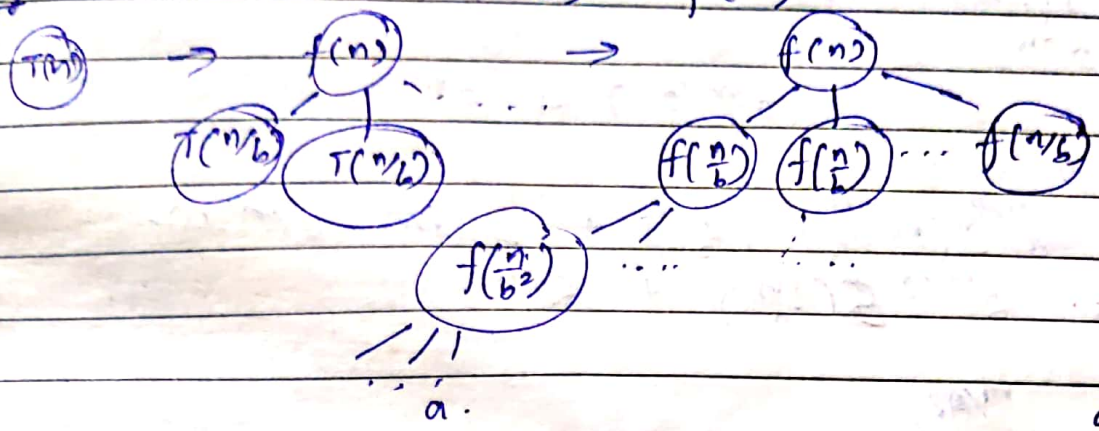
$= \left(T\left(\frac{n}{2^3}\right) + b\frac{n}{2^3}\log\frac{n}{2^3} + b\frac{n}{2}\log\frac{n}{2} + bn\log n\right)$

General $T\left(\frac{n}{2^i}\right) + b\frac{n}{2^i}\log\frac{n}{2^i} + b\frac{n}{2}$

$n = 2^k$ $\quad T(1) + b\frac{2^k}{2^k}$

Date: / / Page no:

Proof: $T(n) = a\,T(n/b) + f(n)$



$n = b^k$

$f(n) + a \cdot f(n/b) + a^2 f(n/b^2) + a^3 f(n/b^3) + \cdots + f(1)$ — k levels ending

$f(b^k) + a\,f(b^{k-1}) + a^2 f(b^{k-2}) + \cdots + f(1)$

1 2 3 4 5 6 6

6 15 21 27

25.01.2023 Wednesday.

Date: / / Page no:

Q) $T(n) = \sqrt{n}\, T(\sqrt{n}) + n$

Sol: Let $n = 2^m$

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

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

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

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

$S(m) = 2^{m/2}$

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

$n^{\log_b a} = n^{\log_2 1} = n^0$

$O(n)$ 2

Date: / / Page no:

Divide & Conquer

Control Abstraction → Flow of Algo.
(earlier written by sir) -

$T(n) = T(n_1) + T(n_2) + T(n_3) + \dots T(n_k) + f(n)$

Time to solve subproblems | Time to Divide + Time to Combine

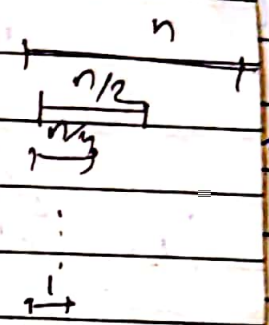
Binary Search

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

$a = 1 \quad b = 2 \quad f(n) = c$

$n^{\log_b a} = n^0 = 1$ ↙ Same Growth Rate

$\therefore T(n) = O(\log_2 n \cdot 1)$



Best case :- $O(1)$
Worst Case & Avg. Case: $O(\log n)$

Mergesort

310	285	179	652	351	423	861	254	450	520
low = 0				mid = $\lfloor \frac{1+10}{2} \rfloor = \lfloor \frac{11}{2} \rfloor = 5$					high = 10
1	2	3	4	5	6	7	8	9	10

310	285	179	652	351	423	861	254	450	520
l=1	m = $\lfloor \frac{1+5}{2} \rfloor = 3$			h=5					
	m = $\frac{1+3}{2} = 2$								
m = $\frac{1+2}{2} = 1$									
285 310									
179 285 310									

m = $\frac{4+5}{2} = 4$

Date: / / Page no:

a (Main Array) [ ] | [ ]

b (Auxillary Array) O(n) compare & copy & remaining elements

b → a. O(n)
copy

∴ O(2n)

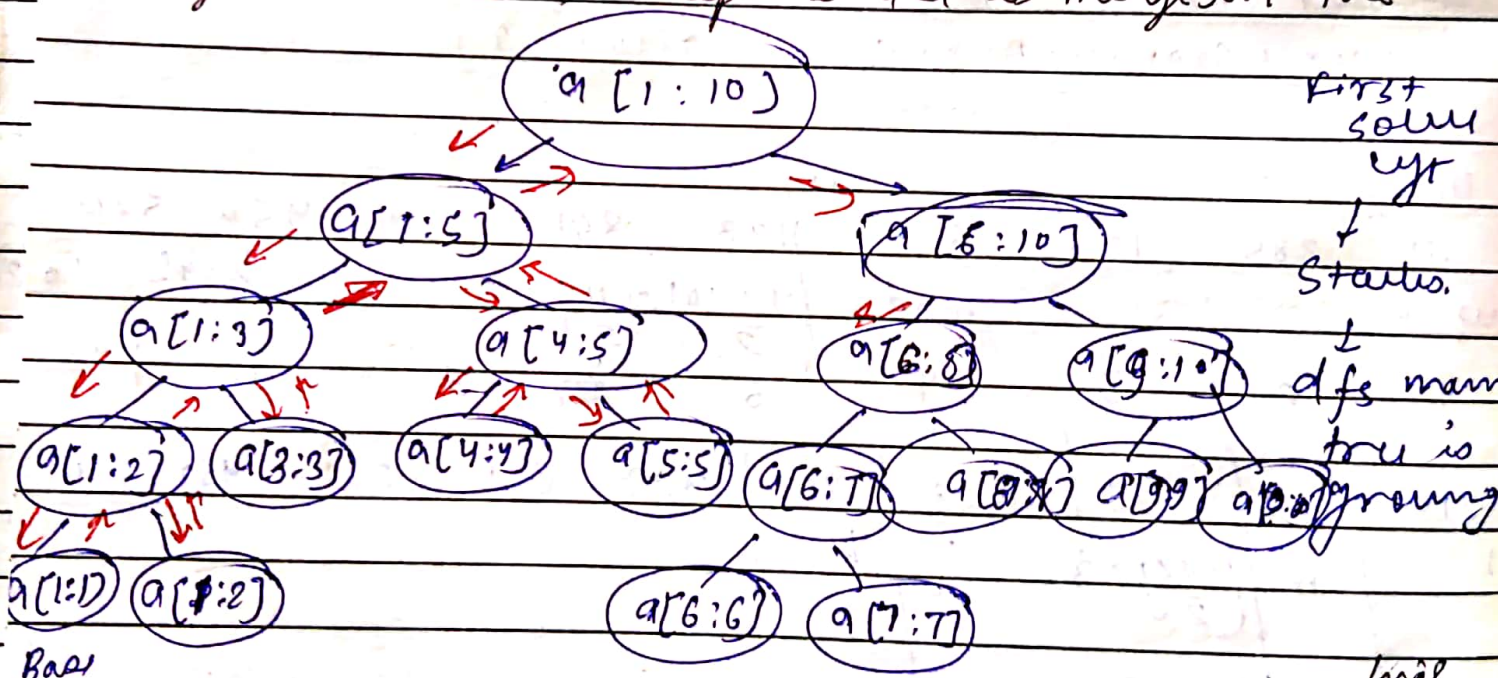
- $T(n) = 2T(n/2) + cn$ ⟶ Splitting + Merging
  $T(n) = aT(n/b) + f(n)$ Const. O(n)

$n^{\log_b a} = n$ == $f(n) = cn$

∴ $T(n) = O(n^{\log_b a} \cdot \log_b n) = O(n \log_2 n)$

BC, WC, AC = $n \log n$

Merge Sort can be represented as Merge Sort Tree



First solve left & Stacks. & dfs manner tree is growing

Base case

high

a[low: ~~mid~~]

a[low: mid] a[mid+1: high]

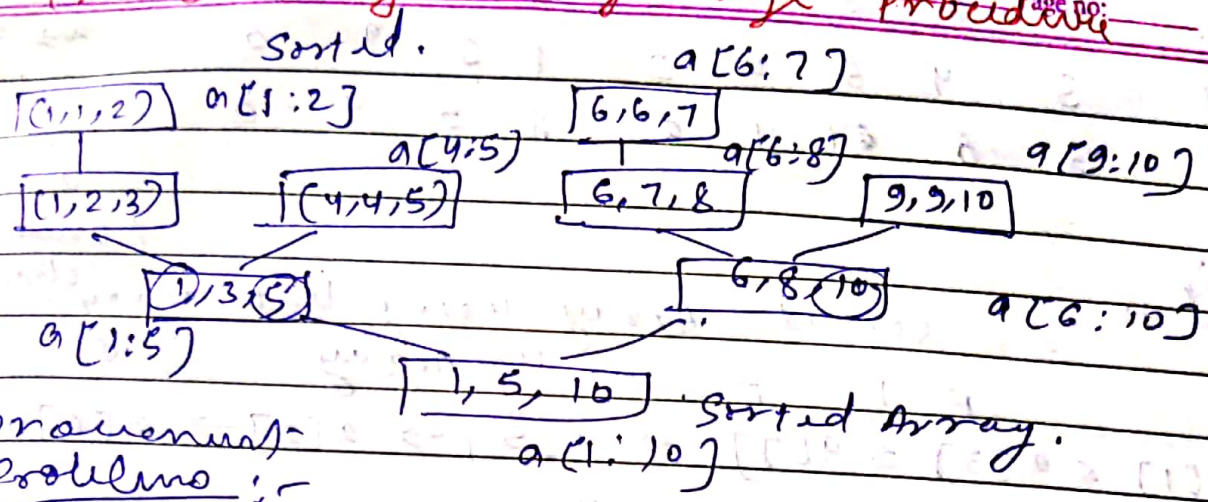
Merge(a, low, mid, high)

30.01.202- Monday.

<FDAS> Lab </FDAS>

Date: Page no:

Tree showing Calls of Merge Procedure



Improvements:-

(1) Problem:-
To reduce recursive stack overhead we can fix min i/p of mergesort from where we will

(2) Auxiliary array to store

(3) Data change (Records can be large) → Merging process

~~LLAA~~ LINK ARRAY → To reduce large records swap during merge process.

Three pointers q, r, p

q → starting index of 1st sorted sublist

r → " " 2nd sorted sublist

p → " " of merged sorted list

LinkArray	1	2	3	4	5	6	7	8	9
	10	30	15	45	25	35	20	40	50
Initialize → q r p	0	0	0	0	0	0	0	0	0
1 2 1	2	0	0	0	0	0	0	0	0
1 3 1	3	0	2	0	0	0	0	0	0
4 5 5	3	0	2	0	4	0	0	0	0
4 5 1	3	4	5	0	2	0	0	0	0
6 7 7	3	4	5	0	2	0	6	0	0
8 9 8	3	4	5	0	2	0	6	9	0

Date: / / Page no:

	1	2	3	4	5	6	7	8
	7	97	3	4	5	0	2	8	6	9	0
	17	1	3	6	67	9	02	8	5	4	0

Stored in sorted order

If any array ends then rem. link array is correct automatically (only any change)

Sorted Array using link array

Link Array

$p = 1$

$\therefore a[1] \le a[3] \le a[7] \rightarrow a[5] \rightarrow a[2] \rightarrow a[6] \rightarrow a[8]$

Link array $1 \rightarrow 3 \rightarrow 7 \rightarrow 5$ link[7]

$a[4]$
$a[2]$
end $= 0$

$10 < 15 < 20 < 25 < 30 < 35 < 40 < 45 < 50$

Advantages :-

1. ~~waste need to~~ Extra space is reduced (Records is replaced by pointers)
2. Some time is reduced (as copying into original array procedure is removed)

Q) k sorted subsets

$x_1 \quad x_2 \quad x_3 \quad \dots \quad x_k \quad \rightarrow$ Not sorted

$\sum_{i=1}^{k} x_i = n$

Merge these subsets in time $O(n \log k)$

Sol. M 1:

$x_1 \quad x_2 \quad x_3 \quad \dots \quad x_k$

$(x_1 + x_2) \quad (x_3 + x_4) \quad \dots \quad (x_{k-1} + x_k)$

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

$= n \quad \frac{n}{2^2}$

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

$n \log_2 k$

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

① 100 → Sort
n=8 Stop Merge Sort.
else Insertion Sort

② Find Array

Date: / / Page no:

M.2: MinHeap. Construct a (x_{1min}, x_{2min}, ... x_{lmin}) l elements

x_1 (x_{1min} ...)
x_2 (x_{2min} ...)
⋮
x_l ——

Bottom up / TopDown
O(l) / O(l log l)

① ~~Construct~~ Remove the root / rehapify (O(log l))

② Suppose X_3 min is the root remove

Fake coin Problem (Binary search)
↳ n is odd (find)
↳ n is even

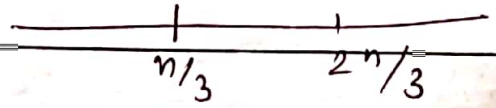
-When we do know coin is heavier or lighter
~~Solve fake~~

Date: / / Page no:

Lighter & heavier not known

Explore the use of Ternary search for fake coin problem.

Ternary Search



2 comparison

```
if (n/3) {

}
else {
    if(2n/3) {

    }
    else {

    }
}
```

$T(n) = T(\frac{n}{3}) + c_2$

$T(n) = O(\log_3 n)$

$T(n) = T(\frac{n}{2}) + c_1$

(Binary Search)

$\Rightarrow c_1 < c_2$

Assignment

What is the total no. of comp. made in binary search. Comp. it with Ternary Search

What is the speedup obtained if we use Ternary Search

left ptr just greater circle

Date: / / Page no:

Quick Sort

pivot, p

<p (1) >p

QS(left) QS(right)

p ∈ any element.

↳ Inplace sorting algo → (Not takes extra space)

p → pivot first element (Hoarde's method)

	1	2	3	4	5	6	7	8	9	10
	p	i++								p--
	65	70	75	80	85	60	55	50	45	+∞
		i=2	70>65						p=9	45<65
swap	45	75	80	85	60	55	50	70	+∞	
		i=3					p=8			
	45	50	80	85	60	55	75	70	+∞	
			i=4			p=7				
	45	50	55	85	60	80	75	70	+∞	
				i=5	p=6					
pivot	65	45	50	55	60	85	80	75	70	+∞
				p=5	i=6					
	60	45	50	55	65	85	80	75	70	+∞

Right ptr. with pivot when i crosses p swap p with pivot

To stop the i ptr to go beyond

<65 >65

(n+1) Partitioning

Partitioning = O(n)

Partioning element is placed at k=5

A[1:-4) A[6...9]

Use Stacks to store

6
1

Upper

Lower

9
4

Upper

8, 17, 25, 29, 33, 48, 64, 75 — pivot & p crosses → All are scanned.

Date: / / Page no:

64 45 50 55

i ↑ p ↑

Sorted Array

	1	2	3	4	5	6	7	8	9	10	+∞
n	p	i	i								i & p
	i	←						←		P	crosses
	p										

n-1	2	3	4	5	6	7	8	9	10

A[1...n]

A[2...n]

A[3...n]

A[n-1...n]

Depth

O(n²)

O(n²)

Reverse Sorted Array

	10	9	8	7	6	5	4	3	2	1	+∞
n	←									→	
	Interchange									i	p

1	9	8	7	6	5	4	3	2	1	+∞
p	i									

01.02.2023 Wednesday

Date: / / Page no:

A[1..10]

A[1..09]

A[2..9]

A[2..8]

depth = O(n)

:-

$O(n^2)$

```
int Partition(Type a[], int m, int p)
//

{ Type - v = a[m], int i = m, j = p
  do {
      do i++;
      while (a[i] < v),        until we find bigger
      do j--;
      while (a[j] > v),        smaller
      if(i<j)
        Interchange (a[i] & a[j]),
  }
  while (i<j);

// Interchange v and a[j]
   a[m] = a[j] ;   a[j] = v;
  return j;

}
```

Date: / / Page no:

```
void Quicksort (int p, int q)
    // a[1 .. n]   a[n+1] = ∞
{
    if (p<q)   // if there are more than 1 element in the array
// Partition the array
int j = Partition (a, p, q+1);
// j is the position of partitioning element.
// Solve the subproblems
Quicksort (p, j-1);
Quicksort (j+1, q);
}}
```

$\underbrace{\quad}_{k-1} \; k \; \underbrace{\quad}_{n-k}$

Avg. Case Analysis

$$C_{Avg}(n) = \underbrace{(n+1)}_{\text{Time to Partition}} + \sum_{k=1}^{n} \overbrace{P(k)}^{\text{equally likely for all}} [C_{Avg}(k-1) + C_{Avg}(n-k)]$$

$$\Rightarrow C_{Avg}(n) = n+1 + \frac{1}{n} \sum_{k=1}^{n} [C_A(k-1) + C_A(n-k)]$$

$$\Rightarrow C_A(n) = n+1+\frac{1}{n} \{ C_A(0) + C_A(n-1) + C_A(1) + C_A(n-2) + \dots$$
$$C_A(n-2) + C_A(1) + C_A(n-1) + C_A(0) \}$$

$$\Rightarrow C_A(n) = n+1 + \frac{2}{n} \{ C_A(0) + C_A(1) + C_A(2) + \dots + C_A(n-1) \}$$

Multiply both sides by n:

$$n C_A(n) = n(n+1) + 2\{ C_A(0) + C_A(1) + C_A(2) + \dots + C_A(n-1) \} \quad \hookrightarrow (I)$$

Replacing (I) for $C_A(n-1)$

Date: / / Page no:

$(n-1)\,C_A(n-1) = (n-1)\,n + 2[C_A(1) + \cdots C_A(n-2)]$ → (II)

(II) − (I):

$C_A(n) - (n-1)\,C_A(n-1) = n(n+1) - n(n-1) + 2C_A(n-1)$

$\Rightarrow nC_A(n) = n\{n+1-(n-1)\} + (2+n-1)C_A(n-1)$

$\Rightarrow nC_A(n) = 2n + (n+1)C_A(n-1)$ → (III)

Divide (III) by $n(n+1)$

$\frac{C_A(n)}{n+1} = \frac{2}{n+1} + \frac{C_A(n-1)}{n}$ ← No of comp.

$= \frac{C_A(n-2)}{n-1} + \frac{2}{n} + \frac{2}{n+1}$ [+1, +2]

$= \frac{C_A(n-3)}{n-2} + \frac{2}{n-1} + \frac{2}{n} + \frac{2}{n+1}$

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

$[C_A(0) = C_A(1) = 0 \text{ (Base case)}]$

$= \frac{C_A(n-(n-1))}{2} + \frac{2}{3} + \cdots + \frac{2}{n-1} + \frac{2}{n} + \frac{2}{n+1}$

$= \frac{C_A(1)}{2} + \frac{2}{3} + \frac{2}{4} + \cdots \frac{2}{n} + \frac{2}{n+1}$

$= 0 + 2\left(\frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n} + \frac{1}{n+1}\right)$

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

$\sum_{i=2}^{n} \frac{1}{i} = \int_{x=2}^{x=n} \frac{1}{x}\,dx = [\log x]_2^n$

$= [\log_e n - \log_e 2]$

$\frac{C_n(n)}{n+1} = \frac{0}{2} + 2[\log n - \log_e 2]$

$\Rightarrow C_A(n) = O(n \log_e n)$ or $C_A(n) = O(n \log_2 n)$

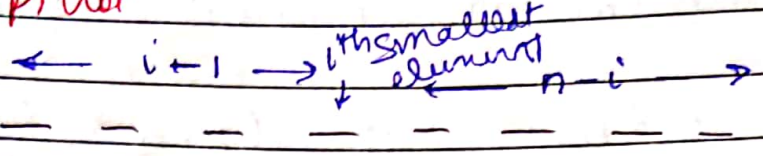
$C = C_0 \log_e 2$

2 2 2 2 2 all elements equal → worst case O(n²)

Date: / / Page no:

Improvements of Quick Sort :-

(I) Random pivot



i = random (1, n)
Interchange a[1] & a[i]
proceed with Normal Quick sort

Same.

$$C_A(n) = (n+1) + \sum_{k=1}^{n} p(k) [C_A(i-1) + C_A(n-i)]$$

Random no. pick rand() → O(1)

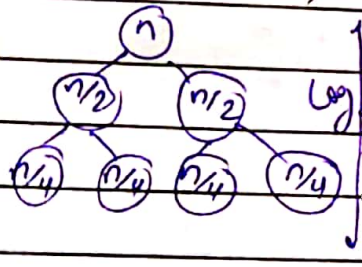
(II) Pick up the median of the list as the pivot element → O(n) → n/2 th smallest element

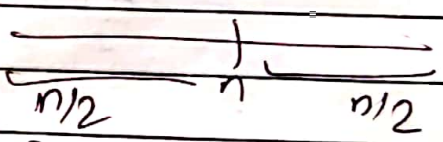
$$T(n) = (n+1) + n + 2T(n/2)$$

↓ n time to partition median

↳ time to find median

k th smallest element in O(n)





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

(3) Pick up pivot as median of 3 low, middle & high element (Median of 3) method

low mid high

↳ Median = O(1)

↳ Not necessarily median

T(n/2) not always.

Skewed trees can be reduced but can't guarantee.

06.03.2023 Monday

Date: / / Page no:

Improvement in Partition Algo :-

(Repetition of element)

< Pivot == Pivot Pivot <

Array

< Pivot = Pivot Pivot < (Divide into 3)

↓ QS ↓ QS

Q) Get -ve on LHS & +ve on RHS

Sol ⇒ 1 -3 9 -4 2 8 6 -7

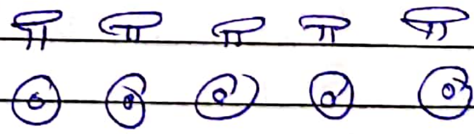
know partitioning criteria

↑ i ↑ j

Q) Nuts & Bolts problem → You can't match nut with Nut & Bolt with Bolt

Sol: Alternatively apply partition in both

pivot



1.

N N N N N

B B B B B

2. N (N) N N N

pivot

3. B (B) B B B

N (N) N N N

$O(n \log_2 n)$

only (This bolt & Nut in left) check

Time comp. + [illegible]
Step ASyn Recu. merge & sort

Date: / / Page no:

STRASSEN'S MATRIX MULTIPLICATION

A n×n			B n×n				C	
a_{11}	a_{1n}		b_{11}	b_{1n}			C_{11} C_{11}	C_{1n} C_{12}
A_{11}	A_{12}	×	B_{11}	B_{12}	=			
A_{21}	A_{22}		B_{21}	B_{22}			C_{21}	C_{22}
a_{n1}	a_{nn}	n×n	b_{n1}	b_{nn}	n×n		C_{n1}	C_{nn} n×n

matrix multiplication

$C_{11} = A_{11} \cdot B_{11} + A_{12} B_{21}$

$C_{12} = A_{11} \cdot B_{12} + A_{12} B_{22}$

$C_{21} = A_{21} \cdot B_{11} + A_{22} B_{21}$

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

$T(n) = 8\,T(n/2) + cn^2$ (addn)

$a = 8,\ b = 2,\ f(n) = cn^2$

$a = 8,\ b = 2,\ f(n) = cn^2$

$n^{\log_b a} = n^{\log_2 8} = n^3$

No Proof only formula

Strassen formula

$P = (A_{11} + A_{22})(B_{11} + B_{22})$

$Q = (A_{21} + A_{22}) \cdot B_{11}$

$R = A_{11}(B_{12} - B_{22})$

$S = A_{22}(B_{21} - B_{11})$

$T = (A_{11} + A_{12}) B_{22}$

$U = (A_{21} - A_{11})(B_{11} + B_{12})$

$V = (A_{12} - A_{22})(B_{21} + B_{22})$

$C_{11} = P + S - T + V$

$C_{12} = R + T$

$C_{21} = Q + S$

$C_{22} = P + R - Q + U$

T → S ↑ [11 12; 21 22] ↓ R, Q → B_{11} A_{11} A_{22} B_{22}

7 matrix multiplⁿ → 18 matrix addⁿ

$T(n) = 7\,(T(n/2)) + 18\,cn^2$

$a = 7,\ b = 2 \qquad f(n) = cn^2$

$n^{\log_b a} = n^{\log_2 7} = n^{2.81}$

$O(n^{\log_2 7})$ or $O(n^{2.81})$

2, 3, 20

If not in powers of 2 then append row of 0 & colum of 0 ↳ to make it power of 2

Date: / / Page no:

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

A_{11} A_{12} A_{21} A_{22} B_{11} B_{12} B_{21} B_{22}

$$P = \left(\begin{bmatrix} 1 & 2 \\ 2 & 0 \end{bmatrix} + \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}\right)\left(\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 2 \\ 1 & 2 \end{bmatrix}\right)$$

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

→ fill diagonal further division

$a_{11}=3,\ a_{12}=3,\ a_{21}=2,\ a_{22}=1,\ b_{11}=1,\ b_{12}=4,\ b_{21}=3,\ b_{22}=3$

$p' = (3+1)(1+3) = 16$ $\quad t' = (3+3)\cdot 3 = 18$

$q' = (2+1)\cdot(1) = 3$ $\quad u' = (2-1)\cdot 5 = 5$

$r' = (3)(1) = 3$ $\quad v' = (2)\cdot(6) = 12$

$s' = (1)(2) = 2$

$c_{11} = 12$ $\quad c_{12} = 21$ $\quad c_{21} = 5$ $\quad c_{22} = 11$

$$P = \begin{bmatrix} 12 & 21 \\ 5 & 11 \end{bmatrix} \rightarrow (I)$$

$$Q = \left(\begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix} + \begin{bmatrix} 2 & 1 \\ 0 & 1 \end{bmatrix}\right)\cdot\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 1 \\ 2 & 2 \end{bmatrix}\cdot\begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}$$

a_{11}

Date: / / Page no:

LONG INTEGER MULTIPLICATION

x, y
n digit integers

$x = a\,2^{n/2} + b \qquad y = c\,2^{n/2} + d$

$x \cdot y = (a\,2^{n/2} + b)\cdot(c\,2^{n/2} + d)$

$= ac\,2^{n} + bc\,2^{n/2} + ad\,2^{n/2} + bd$

$[(bc + ad)\,2^{n/2} = (a+b)(c+d) - ac - bd]$

$T(n) = 4\,T(\frac{n}{2}) + cn$ → 4 multiply

$T(n) = 3\,T(\frac{n}{2}) + cn$ → 3 recursive calls; → For addition

Book-2

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

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

$= 3^2\left[3\,T(\frac{n}{2^3}) + \frac{cn}{2^2}\right] + (\frac{3}{2})cn + cn$

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

In General $T(n) = 3^k\,T(\frac{n}{2^k}) + \left\{\frac{3^{k-1}}{2^{k-1}}cn + \cdots \frac{3}{2}cn + cn\right\}$

Let $n = 2^k$
$\Rightarrow \log_2 n = k$

$= 3^k T(1) + cn\left(1 + \frac{3}{2} + \cdots \frac{3^{k-1}}{2^{k-1}}\right)$

$+ cn\,1\,\frac{\left[(\frac{3}{2})^k - 1\right]}{\frac{1}{2}}$

$= 3^k(T(1)) + \left[(\frac{3}{2})^k - 1\right] 2cn$

$= 3^{\log_2 n}\,a + \left\{(\frac{3}{2})^{\log_2 n} - 1\right\} 2cn$

$= n^{\log_2 3}\,a + \left(n^{\log_2 3/2} - 1\right) 2cn$

Master's Theorem $T(n) = 3T(\frac{n}{2}) + cn$

$a = 3 \quad b = 2$

$n^{\log_b a} = n^{\log_2 3}$

$O(n^{\log_2 3})$

Date / / Page no.

$= n^{\log_2 3} a + n^{\log_2 3/2} 2cn - 2cn$

$= n^{\log_2 3} a + 2n^{\log_2 3 - 1} cn - 2cn$

$= n^{\log_2 3} a + 2cn^{\log_2 3} - 2cn$

$= (a + 2c) n^{\log_2 3} - 2cn$

TC $= O(n^{\log_2 3}) = O(n^{1.55})$

NOTE:

① Decreasing # of multiplication on the expense of # of Addⁿ

Can be done by dividing no. into power of base 10

$XY = (a\,10^{n/2} + b) \cdot (c\,10^{n/2} + d)$

$=$

merge

~~Quick~~ Sort using link Array.

```
int mergesort (low, high)
{   if (low == high)
        return low;
    else if (low < high)
    {   mid = ⌊(low + high)/2⌋;
        q = mergeSort (low, mid);
        r = mergesort (mid+1, high);
        return merge (q, r);
    }
}
```

"No changes"

```
int merge (q, r)
//
{ int i, j, k;
   i = q, j = r, k = 0;            // कोई भी zero तो loop End
  while (i != 0 && j != 0) do      // while both lists are not empty
  {   if (A[i] <= A[j])   // find smaller key
      {
          link [k] = i, k = i, i = link [i];   // add a new key to list
      }
      else
      {
          link [k] = j; k = j; j = link [j];
      }
  } // end while

  if (i == 0)   // first list exhausted
        link [k] = j;
  else
        link [k] = i;

  return link [0];
}
```

small one

Check, किसकी value से End हुआ loop तो पहली दूसरी Remaining list से जोड़ा

	0	1	2	3	4	5	6	7	8
p q r		50	10	25	30	15	70	35	55
2 5 (i j)	~~5~~ 2	0	3	4	1	7	0	8	6

k = 0

k = 5

10 < 15

3 ~~8~~ 7 5

25 > 15

Date: 4/1 Page no: 2

	0	1	2	3	4	5	6
25 < 35							
i j							
4 7		0	5	4	1	3	0	8	6

k = 3

30 < 35 | 0 5 4 1 3 0 8 6 | k = 4

1 7 | 0 5 4 7 3 0 8 6 | k = 7

50 > 35

1 8 | 0 5 4 7 3 0 1 6 | k = 1

50 < 55

end of i ∴ with [k] = j

0
2

8 5 4 7 3 0 1 6

(1) (2) (3) (4) (5) (6) (7) (8)

~~10~~ ~~15~~

50 10 25 30 15 70 35 55

10 < 15 < 25 < 30 < 35 < 50 < 55 < 70 sorted

Division:

i=1 j=2

with

2		
0	1	2
		k

Date / / Page no.

SPACE COMPEXITY

Algorithm Rsum (int a[], int n)
{ if (n <= 0)
    return 0;
  else
    return (Rsum (a, n-1) + a[n]);
}

~~sta~~ SC =

13.02.2023 Monday.

Date: / / Page no:

GREEDY :

n i/ps
↳ subset of inputs - problem constraints
↳ feasible solution
↳ optimal solution
↳ maximizes / minimizes objective function

Greedy Technique

↳ works in stats
↳ take one i/p at a time
↳ add it to ans vector (if optimal) (else reject)

min, max.

```
Procedure Greedy (A, n)
// A(1:n) contains n i/ps
solution ← φ
for i ← 1 to n do
    x ← select(A)
    if feasible(solution, x)
        then solution ← Union(solution, x)
    endif
repeat
return(solution)
end Greedy.
```

Knapsack (Fractional)

n objects $W = \{w_1, w_2, \ldots, w_n\}$ $P = (p_1, p_2, \ldots p_n)$

Knapsack - M $x = (x_1, x_2, \ldots, x_n)$ each $0 \le x_i \le 1$

↳ Fixed size solution vector

Optimal soln

Date: / / Page no:

Maximize $\sum_{i=1}^{n} P_i x_i$ (objective function)

subject to the constraint $\sum_{i=1}^{n} w_i x_i \leq M$ (problem constraint)

↓ Feasible soln

[In 0/1 knapsack, it is variable size soln vector]

length of $x = 1$ to n

each $x_i \in \{0,1\}$

$1 \leq x_i \leq n$ 1, 4, 7

or fixed length

each $x_i \in \{0,1\}$

$x = \{1,0,0,1,0,0,1\}$

$n = 3$ $M = 20$ $(P_1, P_2, P_3) = (25, 24, 15)$

$(W_1, W_2, W_3) = (18, 15, 10)$

x_1	x_2	x_3	$\sum w_i x_i$	$\sum P_i x_i$
1/2	1/3	1/4	16.5	24.25
1	2/15	0	20	28.2
0	2/3	1	20	31
0	1	1/2	20	31.5 ✓

$\sum w_i x_i = \frac{1}{2} \times 18 + \frac{1}{3} \times 15 + \frac{1}{4} \times 10$

$= 9 + 5 + 2.5 = 16.5$

$\sum P_i x_i = \frac{1}{2} \times 25 + \frac{1}{3} \times 24 \times + \frac{1}{4} \times 15 = 12.5 + 8 + 3.75 = 24.25$

① Greedy on profit.

$P = 25$, $W = 18$ (weight used)

(Profit earned)

Ram capacity $= 20 - 18 = 2$

$P = 25 + \frac{2}{15} \times 24 = 25 + 3.2 = 28.2$ $W = 18 + \frac{2}{15} \times 15 = 20$

	25	24	15
wt ↓	18	15	10
	=1.388	=1.6	=1.5
	24/15	15/10	25/18
P/wt	P2	P3	P1

1 24 7.5 5

31.5

$O(n \log n)$

Date: / / Page no:

② Greedy on Weight

Profit earned = $15 \times 1 = 15$ 3rd object

Wt. used $W = 10$

Rem. cap. = $20 - 10 = 10$ $x_2 = \frac{10}{15}$

$W = 10 + 10/15 \times 15 = 20$

$P = 15 + \frac{10}{15} \times 24 = 15 + 16 = 31$

③ Greedy on ~~in the do~~ P_i/w_i

Select objects in the ↓ing order of P_i/w_i

$\frac{P_1}{w_1} = \frac{25}{18}$ $\frac{P_2}{w_2} = \frac{24}{15}$ $\frac{P_3}{w_3} = \frac{15}{10}$ } $O(n)$

$= 1.388$ $= 1.6$ $= 1.5$

Sort. P_2 P_3 P_1 } $O(n \log n)$

1.6 1.5 1.388

$x_2 = 1$

$P = 24 \times 1 = 24$ $W = 15 \times 1 = 15$

Rem. cap. = $20 - 15 = 5$

$x_3 = \frac{5}{10} = \frac{1}{2}$ $P = 24 + \frac{1}{2} \times 15 = 24 + 7.5 = 31.5$

$W = 15 + \frac{1}{2} \times 10 = 20$ Rem. capacity $= 0$ $x_1 = 0$

$x_1 = 0$ $x_2 = 1$ $x_3 = 1/2$

Optimal Merge Pattern

$(x_1, x_2, x_3, \ldots x_n)$

$x_1 - n$
$x_2 - m$ $\rightarrow (x_1 + x_2) \rightarrow O(n+m)$

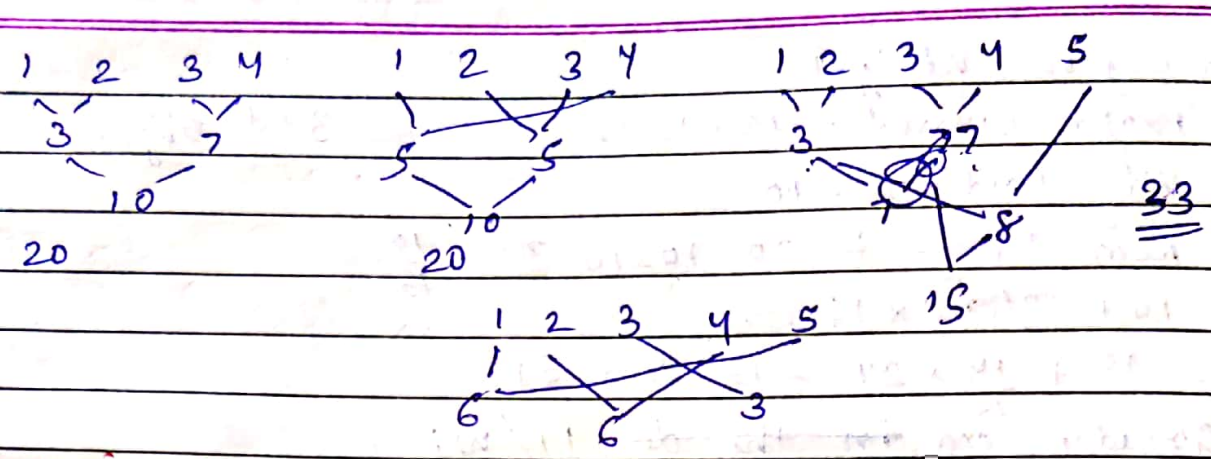
30 20 10 30 20 10

50 30

60 60

$30 + 60 = 110$ $x_1 + x_2 = 30 + 60 = 90$

Date: / / Page no:



Objective fn:

To minimize the no. of records movements

Greedy criterion:

Select the 2 smallest size files at each step ~~force~~ for merging

$f_1, f_2, f_3, f_4, f_5 = (20, 30, 10, 5, 30)$

Sorted order $(f_4, f_3, f_1, f_2, f_5) = (5, 10, 20, 30, 30)$

Merge f_4 & f_3 = z_1 $|z_1| = 15$ Total No. of record movements = 15

Sorted order $(z_1, f_1, f_2, f_5) = (15, 20, 30, 30)$

Merge z_1 & f_1 z_2 $|z_2| = 35$ Total No. of = 15+35=50

Sorted order $(f_2, f_5, z_2) = (30, 30, 35)$

Merge f_2 & f_5 z_3 $|z_3| = 60$ Total = 50+60 =110

Sorted order $(z_2, z_3) = (35, 60)$

Merge z_2 & z_3 z_4 $|z_4| = 35+60 = 95$

Verify correct Calculation: Total = $\sum$ elements in array

Total No. of = 95+110 = 205

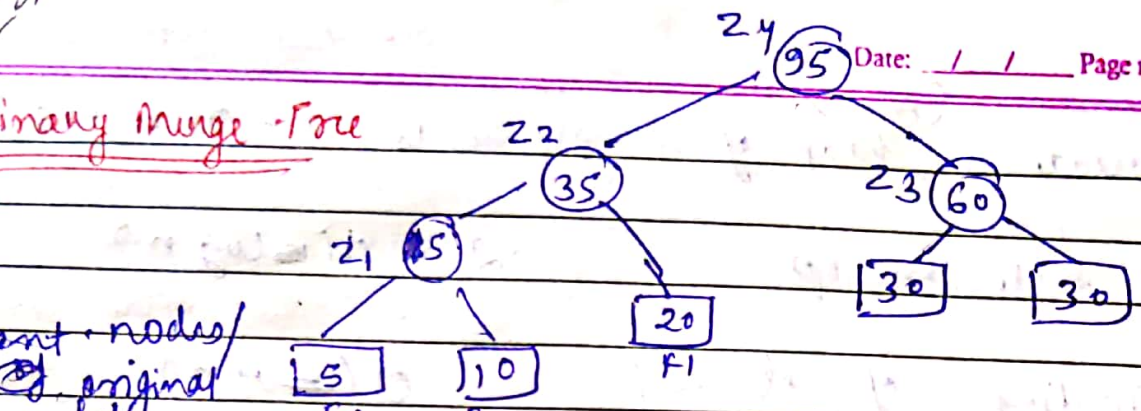
Min. Possible

14.02.2023 Tuesday.

Ternary Merge Tree

Date: / / Page no:

Binary Merge Tree



ext. nodes / original files

path length → Size of file (ext. Node)

Total No. of Record movements = $3 \times 5 + 3 \times 10 + 2 \times 20 + 2 \times 30 + 2 \times 30$

$= 15 + 30 + 40 + 60 + 60$

$= 205$

Weighted Path Length = $\sum_{i=1}^{n} d_i x_i$

d_i → Path length of node from root in Binary Merge Tree.

x_i → Size of the file.

```
Procedure TREE (L, n)
// L is a list of n single node binary tree
for i ← 1 to n do                          // O(n)        O(n)
    Call GetNode (T); // return new node in variable T
    LChild (T) ← least (L);                // constant time   O(log n)
    RChild (T) ← least (L);                //                 O(log n)
    Weight (T) ← Weight (LChild (T)) + Weight (RChild (T));
    Call Insert (L, T) // Inserting node T in sorted list → // O(n)
                                                            // O(log n)
repeat
return (least (L));
end Tree
```

$O(n^2)$ ↓ if we have list.

→ Stored as list

→ Stored as (min) Heap $O(n \log n)$

Date: / / Page no:

(min)

① create a heap of n elements — $O(n \log n)$ / $O(n)$

② 2 Deletion op — $2 * O(\log n)$

③ Adding of element — $O(\log n)$

Huffman Codes (Prefix codes)

$m_1, m_2, \ldots, m_n$ → message

$f_1, f_2, \ldots, f_n$ → frequency of messages

m_1	m_2	m_3	m_4	m_5	m_6	m_7
4	5	7	8	10	12	20

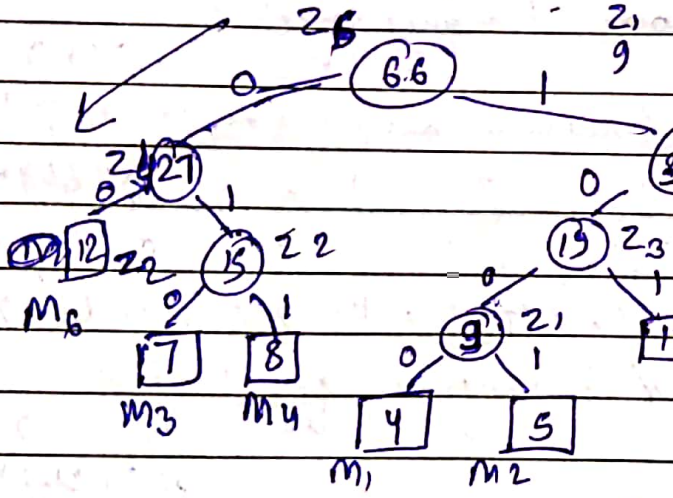
m_5	m_6	m_7
10	12	20

z_1	m_5	m_6	z_2	m_7
9	10	12	15	20

m_1	z_2	z_3	m_3
12	15	19	20

z_3	m_3	z_4
19	20	27

z_4	z_5
27	29



Start from root to message node

can be character of message

$m_1 = 1000$ — $f_1 = 4$

$m_2 = 1001$ — $f_2 = 5$

$m_3 = 010$ — $f_3 = 7$

$m_4 = 011$ — $f_4 = 8$

$m_5 = 101$ — $f_5 = 10$

$m_6 = 00$ — $f_6 = 12$

$m_7 = 11$ — $f_7 = 20$

Avg. no. of bits = $\frac{\text{Total bits}}{\text{No. of chars}}$

Date: / / Page no:

In naive method: Code length = fixed to 4 bit for each of these messages

M_1	M_6	M_7	M_2	M_3	M_4	M_5	
4	2	2	4	3	3	3	→ 21 bits

→ $7 \times 4 = 28$ bits

% Change in bits = $\frac{28-22}{28} \times 100$

$= \frac{6}{28} \times 100$

Decoding

1000 00 11 0 1001 11 010 00 011

M_1 M_6 M_7 M_2 M_7 M_3 M_6 M_4

There is no prefix code of any other [i.e., No confusion]

Start from root go till we reach leaf

1 → Right

0 → Left

NOTE: (1) None of the node is the prefix of the other node.

Confusion 101 / 10 if (present) ? wrong

1 end 1 end

HC ensures there no such present

(2) less frequent messages are away from root.

(3) Min. weighted external path length.

→ $\frac{\Sigma |\text{Mi}| fi}{\Sigma fi}$ = avg. no. of bits used.

Find length encoding

14.02.2023 Wednesday,

Date: / / Page no:

MINIMUM SPANNING TREE

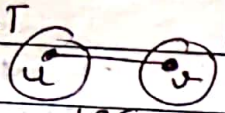
G: Connected subgraph G' such that it has no cycles - Spanning tree.

Minimum Spanning Tree :- Spanning Tree that has least cost Tree.

Cost of MST : sum of weights of edges of the tree.

Prim's

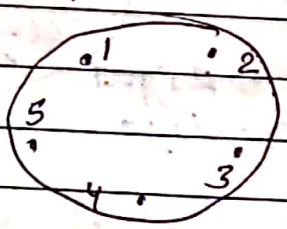
MST



Graph

LCE
(Least cost Edge)

Near(1) = 3



Cost(j, Near(j))

Greedy on cost
∴ Greedy Technique

(1,2) = 2
(1,3) = 1
(1,4) = 3

"Subgraph at each step in Prim's is a tree"

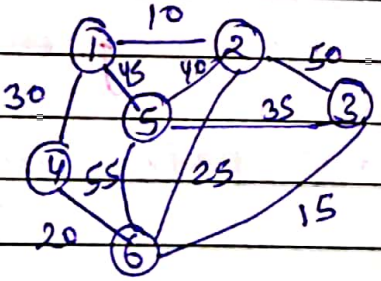
Kruskal's

" Subgraph at each step may be a forest "
collection of trees.
(all not connected)

e - V + 1 edge from cheapest graph in remaining to make it spanning tree

e - (V-1)

Q) Prim's

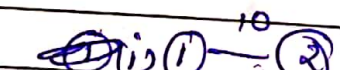


(i)

Near(4) = 6
Cost(4, Near(4)) = ∞

Near(5) = 6
Cost(5, Near(5)) = 15

Near



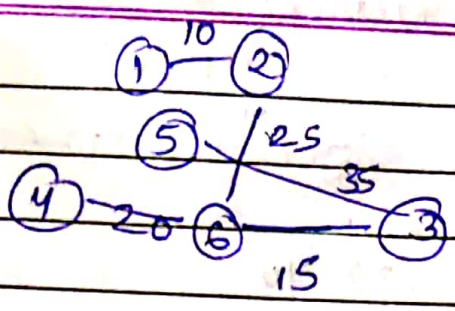
cost
10

at each step select one vertex inside tree & one outside graph to get least cost again & again as soon as we add vertex

Total no. of spanning trees for complete graph = n^{n-2}

For distinct wt. of edges there will be unique MST if it exists

Date: / / Page no



10 + 25 + 15 + 20 + 35 = 105

TC : $O(n^2)$

Main T
Tree

Procedure PRIM (E, Cost, n, T)
//
real Cost(n,n), mincost;
int Near(n), i, j, k, l, n, T(1:n-1, 2)
(k, l) ← edge with min cost → sorting min. edge O(log e)
T(1,1), T(1,2) ← (k, l)
for i ← 1 to n do
    if cost(i,l) < cost(i,k) then Near(i) ← l
    else Near(i) ← k

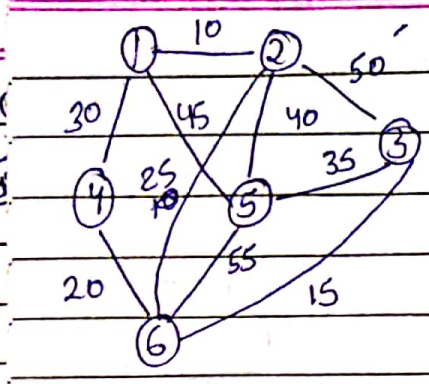
repeat
Near(k) ← Near(l) ← 0 part of
for i ← 2 to n-1 do // n-1 times
let j be an index such that Near(j) ≠ 0
and cost(j, Near(j)) is minimum → O(n) as list

(T(i,1), T(i,2)) ← (j, Near(j));
mincost + ← cost(j, Near(j));
Near(j) ← 0;

Overall $O(n^2)$

for k ← 1 to n do → O(n)
    if Near(k) ≠ 0 && cost(k, Near(k)) > cost(k, j)
        then Near(k) = j;
repeat
repeat
end PRIM

Date: Page no:



Near(1) = 0
Near(2) = 0
Near(4) = 1
cost(4,1) = 30
Near(3) = 2
cost(3,2) = 50
Near(5) = 2
cost(5,2) = 40
Near(6) = 2
cost(6,2) = 25

cost(4,1) = 30
cost(5,2) = 40
cost(3,2) = 50
cost(6,2) = 25

O(n log n) + heap creation
O(n)

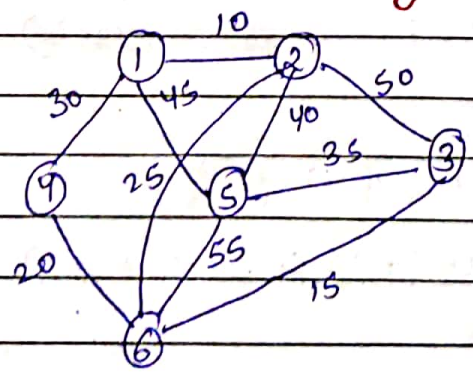
N(1) = 0 N(2) = 0
N(4) = 6
C(4,6) = 20
N(5) = 2 C(5,2) = 40
N(3) = 6 C(3,6) = 15
N(6) = 0

can be optimized to O(log n)
but another loop still O(n)

N(1) = 0 N(2) = 0
N(5) = 3 C(3,5) = 35
N(3) = 0
N(4) = 6 N(4,6) = 20
N(6) = 0

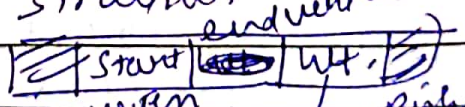
N(1) = 0 N(2) = 0
N(5) = 0
N(4) = 0
N(3) = 0
N(6) = 0

Kruskal's Algo



(1,2) = 10 ✓
(1,5) = 45
(1,4) = 30
(2,3) = 50
(2,5) = 40
(2,6) = 25
(3,5) = 35
(3,4) = 15 ✓
(5,6) = 55
(4,6) = 20 ✓

Min Heap structure



wt ptr
compare ptr
( )

Diff. b/w Prims & Kruskal's → starting vertex will be given in Prims.
↳ will never get disconnected graph

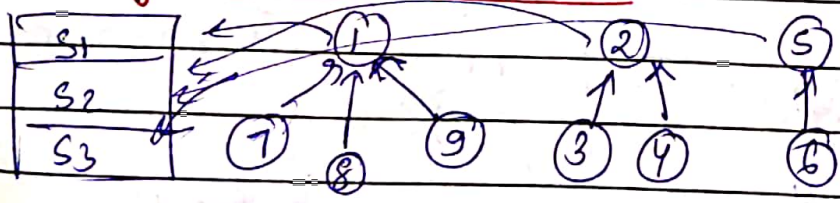
Date: / / Page no:

If not in same set → include else reject that edge
Select min. edge wt.

$S_1(1,2)$ ① —10— ②
$S_2(3,4,\ldots 6)$

$S(1,2)$ ① —10— ②
$S_2(3,4)$ ④ —20— ⑤ ③ —15— ⑥
$S_3(5)$

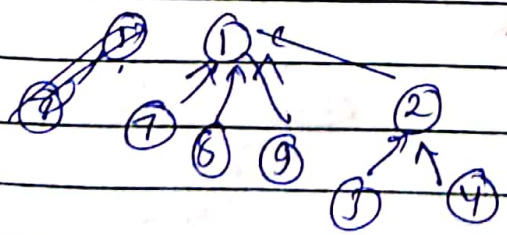
Disjoint set structure



1	2	3	4	5	6	7	8	9
0	0	2	2	0	5	1	1	1

U(1....n)
S1, S2, ... Sm
Tree Representation
- Union
- Find

S1 ∪ S2 → make any one as child of other.



1	2	3	4	5	6	7	8	9
0	1	2	2	0	5	1	1	1

n vertice complete graph has $\frac{n(n-1)}{2}$ edges

Weighted Union

Date: / / Page no:

N_1: No. of nodes की Negative / Set

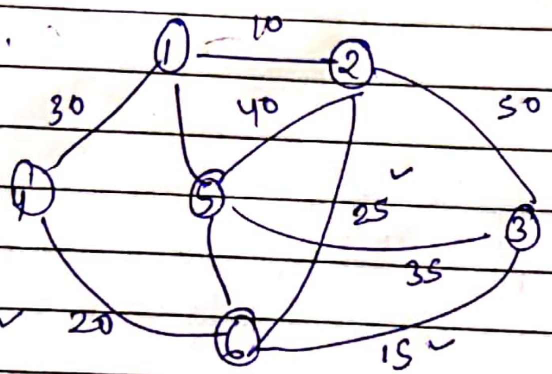
N_2: N / S

$N_1 < N_2$ → make less as child

To maintain $O(\log n)$

else Spanning Square

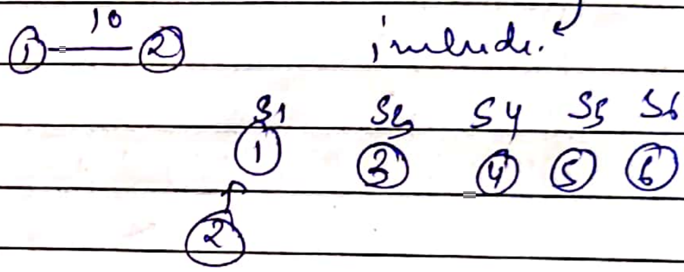
1 → 2 (simple union)



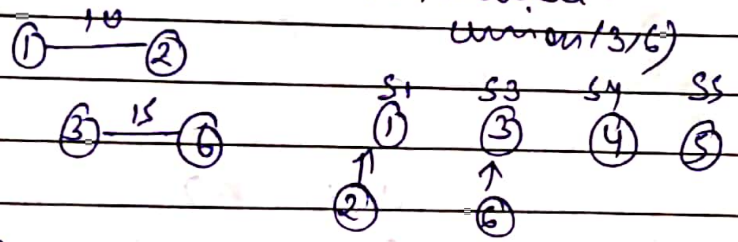
I. S1 S2 S3 S4 S5 S6 — 1 2 3 4 5 6

Take min. wt. (1,2)

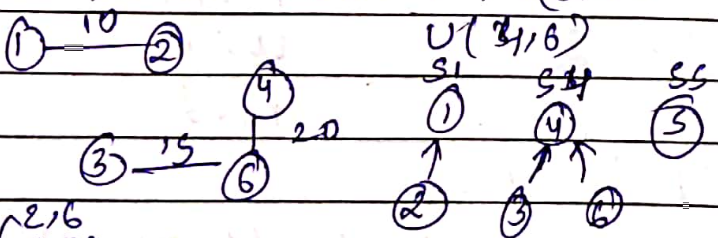
Find(1) = S1
Find(2) = S2] diff → include.



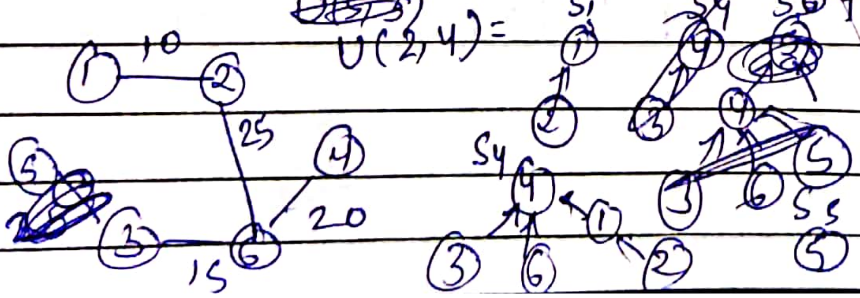
II. (3,6) F(3) = 3 ≠ F(6) = 6
∴ include.
Union(3,6)



III. (4,6) = 20 F(4) = S4 ≠ F(6) = S3
U(3,4,6)

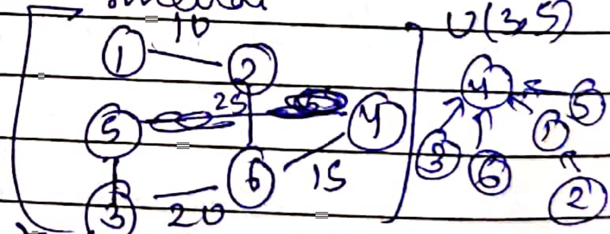


IV. (2,6) = 25
F(2) = S1 ≠ F(6) = S4 Include
U(2,4) =



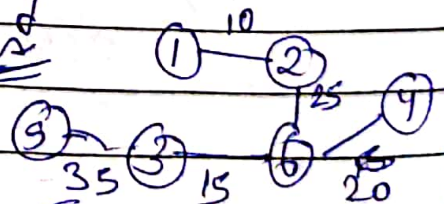
V. (1,4) = 30
F(1) = S4 ≠ F(4) = S4
Not include

VI. (3,5) = 35
F(3) = S4 ≠ F(5) = 5
include
U(3,5)



VII. (2,5) F(2) ≠

Ans



21.02.2023 Tuesday 3, 4, 8

cost matrix — Data structure to store MST — cost of MST

Date: / / Page no:

Procedure Kruskal (E, cost, n, T, mincost)

//

Disjoint set structure

e = E.size()

real mincost, Cost (1:n, 1:n)

int Parent (1:n), T(1:n-1, 2), n, j, k; , Bottom up Appr.

Construct a heap out of edge cost O(e)

Parent ← -1 // each vertex is in a disjoint set O(n)

counter of edges: i ← mincost ← 0;

while i < n-1 and heap not empty do

log(e) { Delete a min cost edge (u,v) from the heap and reheapify;

O(log(n)) { j ← Find (u) ; k ← Find(v);

Used weighted Union O(n): if (j ≠ k) then i++;

(T(i,1), T(i,2)) ← (u,v);

mincost ← mincost + cost(u,v);

call Union (j,k);

endif

repeat;

if i ≠ (n-1) then print ("No spanning tree")

end if

return;

end Kruskal.

	T 1	T 2
i=1	u	v

E = {{1, 2), . . . }

(n-1) log e, O(n-1) log n

O(n log e)

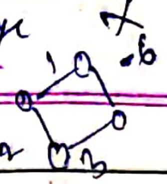
Time complexity : O(|V| log|E|)

|V| → No. of vertices, |E| → No. of edges

22-02-2023
Wednesday

Date: / / Page no:

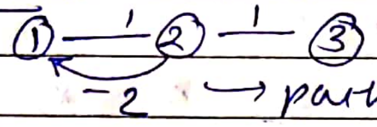
→ assumption in syllabus. Directed the edge



Single Source Shortest Path Algorithm

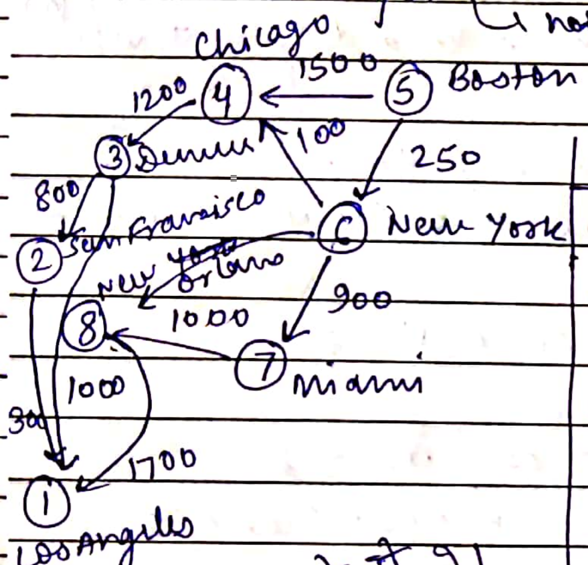
Works like Prim's but only src is fixed.

Limitation: Will not work for -ve cycle. (-ve weight cycle) → not always (Infinite)



→ path length would decrease in this cycle.

Will work for -ve edge but not for -ve cycle.
↳ not always



S	Vertex selected	1	2	3	4	5	6	7	8
5	~~∞~~	∞	∞	∞	1500	0	250	∞	~~∞~~
5	6	∞	∞	∞	1250	0	250	1150	1650
5	6,7	∞	∞	∞	1250	0	250	1150	1650
5	6,7,4	∞	∞	2450	1250	0	250	1150	1650
5	6,7,4,8	2350	∞	2450	1250	0	250	1150	1650
5	6,7,4,8,3	2350	[illegible]	2450	1250	0	250	1150	1650
5	6,7,4,8,3,2	2350	3250	2450	1250	0	250	1150	1650

Less ~~से~~ है तो Replace करो। बराबर है तो 1

1335 + 800 = 4150   2450 + 800 = 3250

3350 3250 2450
~~2750~~ 3350

Adjacency matrix — Adjacency list

↳ iterate all vertices. O(n)

Procedure ShortestPath (V, Cost, DIST, n)
// ...

V → src vertex; Cost → n×n adjacency matrix array; DIST → value of the shortest path; n → no. of vertices

1	2	3	4	...	n

Boolean S(1:n) → visited

Not included → 0 & 1 → included / circled

real Cost(1:n, 1:n), DIST(1:n)

Date: / / Page no:

```
int u, v, n, num, i, w;
for i ← 1 to n do        // Initialize S
    S(i) ← 0,  Dist(i) ← cost(v,i)
repeat
O(n)  S(v) ← 1 ;  Dist(v) ← 0; // put vertex v in set S
* for num ← 2 to n-1 do //determine n-1 paths to v.
O(n)  choose u such that Dist(u) = min{Dist(w)}
  if in                  and S(w) = 0
  list
    S(u) ← 1   // put vertex u in set S (visited)

  for all w with S(w)=0 do // update distances
O(n)    Dist(w) ← min{DIST(w), DIST(u) + cost(u,w)}
  repeat
```

$O(n^2)$ → Adjacency matrix

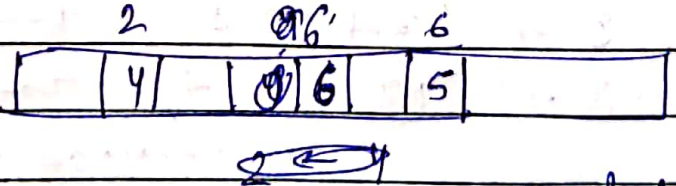
* repeat and Shortest paths.

$O(n^2)$

$O(n^2)$ → Adjacency list
↳ but we reduce its time complexity
$O(e)$ → No. of edges

** In this min Heap.
↳ ** time complexity does not changes.

How to get the 'shortest path?



↳ update when we update DIST-array.

Date: / / Page no:

on

Optimal Storage of Programs on Tape.

$P_1 \ldots P_n$ → programs

$l_1 \ldots l_n$ → length of programs

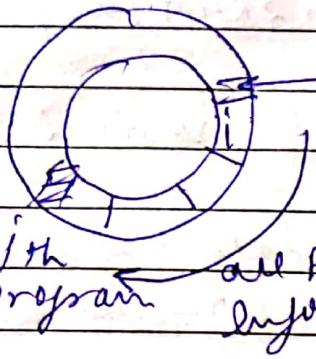
magnetic tape of length L is given.

$\left(\sum_{i=1}^{n} l_i \le L\right)$ → we can store.

$I = \{i_1, i_2, \ldots i_k, \ldots i_n\}$

Read program i_j

Retrieval time $t_j = \sum_{k=1}^{j} l_{i_k}$



all programs read before j

program stored at i

If all the programs are retrieved equally often

then the mean retrieval Time (MRT) $= \frac{1}{n}\sum_{j=1}^{n} t_j$

or $D(I) = \sum_{j=1}^{n}\sum_{k=1}^{j} l_{i_k}$

Probability of retrieval of all programs is equal (in this problem)

Minimize the MRT

$n = 3$, $l_1, l_2, l_3 = (5, 10, 03)$ $L = 20$

ordering I	D(I)	
1, 2, 3	(5) + (5+10) + (5+10+3) = 38	
1, 3, 2	(5) + (5+3) + (5+3+10) = 31	
2, 1, 3	(10) + (10+5) + (10+5+3) = 43	maximum
2, 3, 1	(10) + (10+3) + (10+5+3) = 41	
3, 1, 2	3 + (3+5) + — = 22	minimum
3, 2, 1	3 + (3+10) + — = 34	

Observation: Non Decreasing order (Increasing order) for min. time of retrieval

For maximizing: Order

Date: / / Page no:

Proof:- If $l_1 \le l_2 \le l_3 \le \dots \le l_n$ then the ordering $i_j = j \quad 1 \le j \le n$

minimize $\sum_{k=1}^{n} \sum_{j=1}^{k} l_{ij}$

Proof: $I = i_1, i_2, \dots i_n$ be any permutation of index set $\{1, 2, \dots n\}$ then $D(I) = \sum_{k=1}^{n} \sum_{j=1}^{k} l_{ij} =$

observed in previous example $\Rightarrow \sum_{k=1}^{n} (n-k+1)\, l_{i_k}$

If there exists a & b such that $a < b$ and $l_{i_a} > l_{i_b}$ then interchanging i_a & i_b results in permutation I' → position

$$D(I') = \sum_{k,\ k \ne a,b} (n-k+1) + (n-a+1) l_{i_b} + (n-b+1) l_{i_a}$$

Subtracting $D(I')$ from $D(I)$

$$D(I) - D(I') = (n-a+1)(l_{i_a} - l_{i_b}) + (n-b+1)(l_{i_b} - l_{i_a})$$

$$= (b-a)(l_{i_a} - l_{i_b}) > 0$$

Hence no permutation which is not in non decreasing order of l_i's can

$D(I)$ { $\dots a < b \dots n$: $l_{i_b}\ l_{i_a}$

$D(I)$ { $\dots a < b \dots n$: $l_{i_a}\ l_{i_b}$

program should have smaller length either $l_{i_a} > l_{i_b}$ Big

$D(I) > D(I')$

suboptimal $D(I) = \sum_{i=1,\ i \ne a,b}^{n} (n-k+1) l_{i_k} + (n-a+1) l_{i_a} + (n-b+1) l_{i_b}$

Objective fn is less optimal $D(I') = \sum_{i=1,\ i \ne a,b}^{n} (n-k+1) l_{i_k} + (n-a+1) l_{i_b} + (n-b+1) l_{i_a}$

Date: / / Page no:

Assign

① What will be the change in storage strategy if programs are to stored on multiple tapes.

$\Sigma l_i > L$

② What is the storage strategy if the programs being stored are retrieved at diff. freq. that is each program p_i we have a frequency f_i associated with it.

{ length, freq., freq/length } → greedy criteria.

1 → 1 2 → 2 3 → 3

①

| 2 | 3 | 2 + (2+3) = 7

| 4 | 5 | 4 + (4+5) = 13

20

2 3 4 5

| 2 | 4 | 2 + (2+4) = 8

| 3 | 5 | 3 + (3+5) = 11

11

min heap { }

element / no. of heaps ②

② 2 3 4 5

5 1 2 3

10 3 8

2×5 + (2+3)×1 + (2+3+4)×2

10 + 5 + 18 = 33

2×1 + (2+3)×1 + (2+3+4)×2 + (2+3+4+5)

3×1 + (3+4)×2 + (2+3+4)×5

= 3 + 14 +

2×5 + (2+4)×2 + (2+3+4)×1

= 10 + 12 + 9 = 31

Date: / / Page no:

(1) find $f_i S_{z_i}$

	1	1
	2	3
freq more if assigned first	$1\times3 + 2\times2$	
	$=3+4$	
value more if assigned first	$1\times2 + 2\times3$	

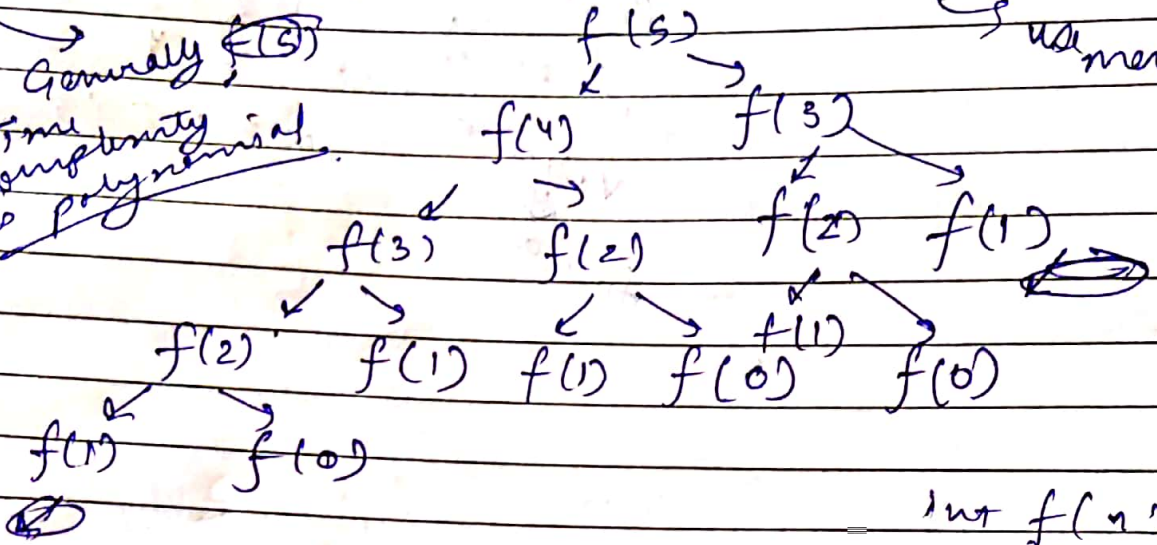
27.12.2023 Monday

Date: / / Page no:

DYNAMIC PROGRAMMING

Nature of Subproblem : overlapping → use memoization

Generally F(S) Time complexity is polynomial




```
int f(n) {
if(n==0)
  return 1;
if(n==1)
  return 1;
if(dp[n]!=-1)
  return dp[n];
dp[n] = f(n-1) + f(n-2)
}
```

Time complexity $O(n)$

Optimal Substructure

↳ solution to multiple subproblem must be in optimal

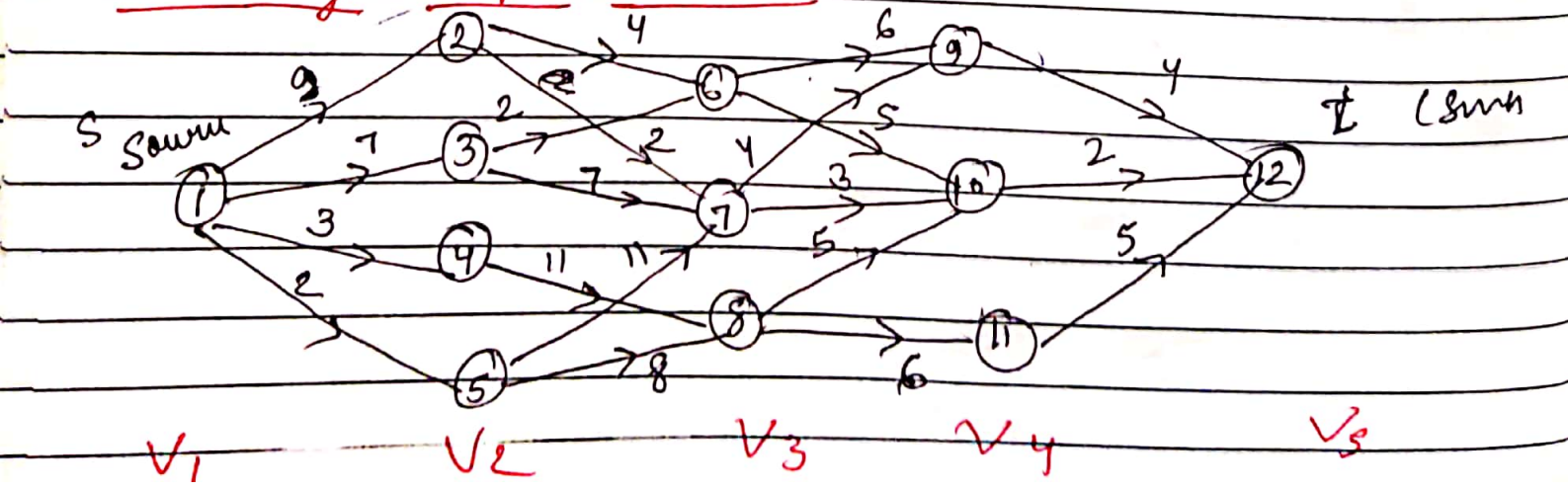
```
int dp[n];
memset(dp,-1,
sizeof(dp))
```

i, i1, i2, i3 ... ik j → shortest path

i, i1, i2 ... ik j' → shortest path

i, i3, ... i'

Multistage Graph Problem



DP & D&C
Similarity :- Subproblem solving
Dissimilar :- No overlapping subproblem

Date: / / Page no:

V_i → Stage i

Travel from Stage i to Stage i+1 & reach Stage n

k Stages

k-2 decisions

(1st & last ~~fix~~ option (hotel) which city to take at each city

Forward Approach

(Top Down) (~~Bottom up~~)

$C(k+1, i) = \min\{\text{possible}\}$ C(Stage, Hotel)

$C(k-2, 6) = \min\{$

Cost(stage, vertex); cost of stage if we take "vertex"

C(i, j) cost of edge i-j

Cost (4, 9) = C(9, 12) = 4

Cost (4, 10) = C(10, 12) = 2

Cost (4, 11) = C(11, 12) = 5

Cost (3, 6) = min{ C(6, 9) + Cost(9, 12), C(6, 10) + Cost(10, 12) }
= min(6 + 4, 5 + 2) = 7

4, 9 / 4, 10

store D(3, 6) = 10 to get shortest path from 6 you need to take 10

Cost (3, 7) = min{ C(7, 9) + Cost(9, 12), C(7, 10) + Cost(4, 10) }
= min{4 + 4, 3 + 2} = 5

4, 9

D(3, 7) = 10

Cost (8, 11) = min{ C(8, 10) + Cost(4, 10), C(8, 11) + Cost(4, 11) }
= min{7, 11} = 7

D(3, 8) = 10

Cost (2, 2) = min{ 4 + 7, 2 + 5 } = 7

6 / 11 / 7 / 7

D(2, 2) = 7

Date: / / Page no:

Cost(2,3) = min { Cost(3,6) + C(3,6), Cost(3,7) + C(3,7), Cost(3,8) + C(3,8) }
= min (7+2, 5+7) = 9 D{2,3} = 6

Cost(2,4) = min { Cost(3,8) + C(4,8) } = 7+11 = 18
D(2,4) = 8

Cost(2,5) = min (Cost(3,7) + C(5,7), Cost(3,8) + C(5,8) }
= (5+11, 7+8) = 15
D(2,5) = 8

Cost(1,5) = min { Cost(2,2) + C(1,2), Cost(2,3) + C(1,3), Cost(2,4) + C(1,4), Cost(2,5) + C(1,5) }
min { 7+9 = 16, 9+7 = 16, 18+3 = 21, 15+2 = 17 }
D(1,5) = 2

1 → 2 → 7 → 10 → 12

There is one more path shortest 1 → 5 → 6 → 10

Backward Approach (Bottom up)

Cost(2,i) = Cost(1,i) = C(1,i)
~~Shortest Path~~ from vertex i in stage 2 to source vertex in stage 1

Cost(2,2) = C(1,2) = 9 D(2,2) = 1
Cost(2,3) = C(1,3) = 7 D(2,3) = 1
Cost(2,4) = C(1,4) = 3 D(2,4) = 1
Cost(2,5) = C(1,5) = 2 D(2,5) = 1

Cost(3,6) = min { C(2,6) + Cost(2,2), C(3,6) + Cost(2,3) } = min { 4+9, 2+7 } = 9
D(3,6) = 3

Date: / / Page no:

cost(3,7) = min { C(3,7) + cost(2,3) ;
C(5,7) + cost(2,5)
C(2,7) + cost(2,2) }
= min { 7+7, 11+2, 2+9 } = 11 D(3,7) = 2

cost(3,8) = min(11+3 , 8+2) = 10 D(3,8) = 5
4 5

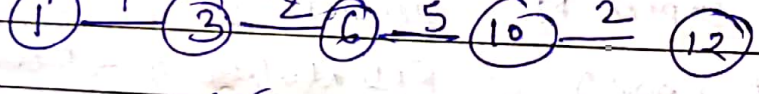
cost(4,9) = min(6+9 , 5+11) = 15 D(4,9) = 6
6 7

cost(4,10) = min(5+9, 3+11, 5+10) = 14,14 D(4,10) = 6, 7
6 7 8

cost(4,11) = min(6+10) = 16 D(4,11) = 8

cost(5,12) = min(4+15, 2+14, 5+16) = 16
9 10 11

D(5,12) = 10



Path wt. = 16

or

Path wt. = 16

28.02.2023 Tuesday

Date: / / Page no:

Reliability Design Problem

Get max^m reliability under the constraint of cost

$D_1 \rightarrow D_2 \rightarrow \cdots \rightarrow D_n$ $\quad D_i \rightarrow r_i$

Reliability of $D_i = r_i$

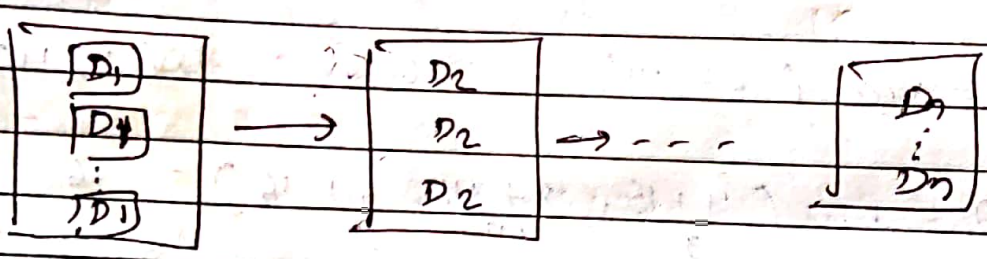
Reliability of circuit $= \phi = \prod_{i=1}^{n} r_i$

Ex: let $r_i = 0.99$

for $n=4$ devices $\phi = 0.96$

$n=10$ $\phi = 0.904$

$\phi \downarrow$ as $n \uparrow$



m_i copies of Device D_i:

let $D_i = r_i$

$D_i = C_i$

Cost of Device i

$\phi_i(m_i)$ = Reliability of Device i having m_i copies

Probability that all m_i copies will fail $= (1-r_i)^{m_i}$ ← no. of D_i on fail

∴ Probability that atleast one of them will work $= 1 - (1-r_i)^{m_i}$

$\Rightarrow \phi_i(m_i) = 1 - (1-r_i)^{m_1}$

$$\phi = \prod_{i=1}^{n} \phi_i(m_i) = \prod_{i=1}^{n} \left(1 - (1-r_i)^{m_i}\right)$$

Maximize ϕ subject to the constraint that $\sum_{i=1}^{n} m_i C_i \leq C$ such that $m_i \geq 1$

Date: / / Page no:

For each device we can compute the upper limit u_i

$$1 \le m_i \le u_i$$

$$u_i = \frac{C - \sum_{i=1}^{n} c_i}{c_i} + 1$$

↓ atleast

extra.

$n = 3 \quad C = 105 \quad c_1 = 30 \quad c_2 = 15 \quad c_3 = 20$

$r_1 = 0.9 \quad r_2 = 0.8 \quad r_3 = 0.5$

$$u_1 = \left[\frac{105 - (30+15+20)}{30}\right] + 1 = \left[\frac{40}{30}\right] + 1 = 1 + 1 = 2 \text{ (min.)}$$

$$u_i = \frac{C - \sum c_i + 1}{c_i} = \frac{C - \sum c_i + c_i}{c_i}$$

$$u_2 = \left[\frac{105 - (30+15+20)}{15}\right] + 1 = \left[\frac{40}{15}\right] + 1 = 2 + 1 = 3 \text{ (min.)}$$

$$u_3 = \frac{40+1}{20} = 2 + 1 = 3 \text{ min.}$$

Enumerate set → Reliability of Device set

$S_i^0 = \{(1, 0)\}$ → cost of Device set

return no device set — multiply in pair — started with 1 as multiplication in f_n

S_1^1 → No. of copies = $\{(0.9, 30)\}$ ↑ Add in pair

$\phi_1(m_1 = 1) = 1 - (1 - 0.9)^1 = 0.9$

$S_2^1 = \{(0.99, 60)\}$

$\phi_1(m_1 = 2) = 1 - (1 - 0.9)^2 = 0.99$

$S^1 = S_1^1 \cup S_2^1 = \{(0.9, 30), (0.99, 60)\}$ ← All the decisions of stage-1

$\phi_2(m_2 = 1) = 1 - (1 - 0.08)^1 = 0.8$, $c_2 = 15$

$S_1^2 = \{1D_1 + 1D_2, \quad 2D_1 + D_2\}$

$\{(0.9 \times 0.8, 30+15), \quad (0.99 \times .8, 60+15)\}$

$\{(0.72, 45), \quad (0.792, 75)\}$

Date: / / Page no:

$2D_2$ $m_2=2$ $\phi_2(m_2=2) = 1-(1-0.8)^2 = 1-(0.2)^2 = 1-0.04$
$= 0.96$

$S_2^2 = \{(0.9\times0.96, 30+2\times15), (0.99\times0.96, 60+30)\}$
$= \{(0.864, 60), (0.9504, 90)\}$ → Not a feasible solution
cost remaing $= 105-90 = 15 < 20$ D_3

$S_3^2 = (0.9\times0.992, 30+45), (0.8928, 75)$ as D_3 can't be there
$\phi_2(m_2=3) = 1-(1-0.8)^3 = 1-0.008 = 0.992$
$3\times C_2 = 3\times15 = 45$
Stage 2 completed Circuit incomplete.

$S^2 = \{(0.72, 45), (0.792, 75), (0.1864, 60), (0.8928, 75)\}$

$\phi_3(m_3=1) = 1-(1-0.5) = 0.5$ | $C_3 = 20$

t_1 t_2 t_1 t_2

$t_1.R < t_2.R$ R = Reliability
$\Delta\Delta\ t_1.C > t_2.C$ C = Cost
$\therefore t_2$ is dominating in cost remaining

$S_1^3 = \{(0.72\times0.5, 45+20), (0.864\times0.5, 60+20), (0.8928\times0.5, 75+20)\}$
$\phi_3(m=1 = (1-0.5)^1 = 0.5$ | $C_3 = 20$

$S_1^3 = \{(0.36, 65), (0.432, 80), (0.4464, 95)\}$
$1D_1+1D_2+1D_3$ $1D_1+2D_2+1D_3$ $1D_1+3D_2+1D_3$

$S_2^3 = \{(0.72\times0.75, 45+40), (0.864\times0.75, 60+40), (0.828\times0.75, 75+40)\}$
$\phi_3(m_3=2) = 1-(1-0.5)^2 = 1-0.25 = 0.75$ $2C_3 = 40$

$= \{(0.54, 85), (0.648, 100), (0.621, 105)\}$ 5

$\phi_3(m_3=3) = 1-(1-0.5)^3 = (1-0.5)$ $1-0.125 = 0.875$

$S_3^3 \{(0.72\times0.875, 45+60), (0.63, 105)$ other cost exceed

01/03/2023 Wednesday

Date: / / Page no:

$S^3 = \{ (0.36, 65), (0.432, 80), (0.446, 95), (0.54, 85), (0.648, 100), (0.684, 105), (0.63, 105) \}$

Pruned (Dominating)

$D_1 + 2D_2 + 2D_3$

$R = 0.648$ $C = 100$ (cost)

TRAVELLING SALESPERSON PROBLEM

Hamiltonian cycle + min. wt.

1 → (n-1) cities → 1

$g(1, V-\{1\}) = \min \{ C_{1k} + g(k, V-\{1,k\}) \}$

1 is fix

$2 \le k \le n$ choices

Path starting at 1, going through all vertices, ending at 1.

Shortest path starting at vertex k going through remaining (n-2) vertices & returning at vertex 1.

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

Shortest path starting at vertex i going through vertices in set S and ending at vertex 1.

$g(i, \phi) = C_{i1}$ (Null)

$|S| = 0$ Size of set = 0

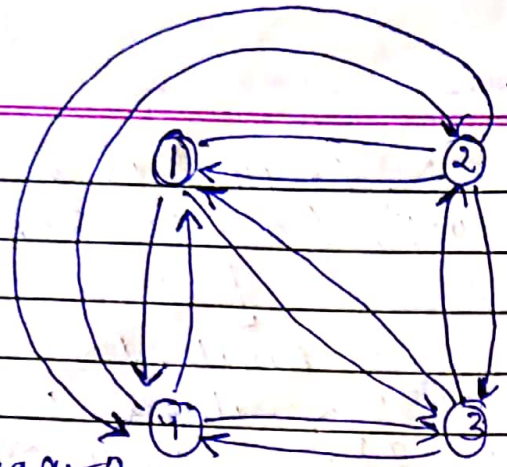
i → 1 : C_{i1}

Brute force

TC = O(n!)

n choices, (n-1)! all combinations

Date: / / Page no:



$n = 4$

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

Stage-0

$g(2, \phi) = C_{21} = 5$ $\quad g(3, \phi) = C_{31} = 6$ $\quad g(4, \phi) = 8$

Stage-1

$g(2, \{3\}) = C_{23} + g(3, \phi) = 9 + 6 = 15$

2 3 1 → starting at 2 going through 3 ending at 1

$g(2, \{4\}) = C_{24} + g(4, \phi) = 10 + 8 = 18$

$2 \to 4 \to 1$

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

$3 \to 2 \to 1$

$g(3, \{4\}) = C_{34} + g(4, \phi) = 12 + 8 = 20$

$3 \to 4 \to 1$

$g(4, \{2\}) = C_{42} + g(2, \phi) = 8 + 5 = 13$

$4 \to 2 \to 1$

$g(4, \{3\}) = C_{43} + g(3, \phi) = 9 + 6 = 15$

$4 \to 3 \to 1$

Stage-2

With 2 intermediate vertices :-

No min( ) comparision is required till now.

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

$= \min(9 + 20, 10 + 15)$

$= 25$

$2 \to 3 \to 4 \to 1$

or $2 \to 4 \to 3 \to 1$ ✓

Already computed

Matrix

$g(2, \{3,4\}) = 4$

$J(2,\{3,4\}) = 4$ Backward tracing

When starting at 2 I will get shortest path through 4

Date: / / Page no:

$= 25$

$g(3,\{2,4\}) = \min(c_{32} + g(e,\{4\}),\ c_{34} + g(4,\{2\}))$

$3 \to 2 \to 4 \to 1$

or $3 \to 4 \to 2 \to 1$

$= \min(13 + 18,\ 12 + 13) = 25$ (31, 25)

$J(3,\{2,4\}) = 4$

$= 23$

$g(4,\{2,3\}) = \min(c_{42} + g(2,\{3\}),\ c_{43} + g(3,\{2\}))$

$= \min(8 + 15,\ 9 + 18)$ (27) $= 23$

$4 \to 2 \to 3 \to 1$

$4 \to 3 \to 2 \to 1$

$J(4,\{2,3\}) = 2$

Reusing such results

Overlapping subproblems

DP

Stage-3

$g(1,\{2,3,4\}) = \min\{$

$1 \to 2 \longrightarrow 1$: $c_{12} + g(2,\{3,4\})$,

$1 \to 3 \longrightarrow 1$: $c_{13} + g(3,\{2,4\})$,

$1 \to 4 \longrightarrow 1$: $c_{14} + g(4,\{2,3\})$

$\}$

$= \min\{$

$10 + 25$,

$15 + 25$,

$20 + 23$

$\}$

$= \min\{35, 40, 43\}$

$= 35$

Calculated in Stage-2

Overlapping subproblem.

"Stored" dp

cost matrix

$J(1,\{2,3,4\}) = 2$

$J(3,\phi) = 1$

Retracing (Reverse order)

$1 \xrightarrow{10} 2 \xrightarrow{9} 4 \xrightarrow{9} 3 \xrightarrow{6} 1$

$J(1,\{2,3,4\}) = 2$

$J(2,\{3,4\}) = 4$

$J(4,\{3\}) = 3$

$10 + 9 + 9 + 6 =$

Brute force

$TC = O(n!)$

$2 \to 3 \to 4$

Combinational problem

$g(i,S)$ $\quad 2 \le i \le n-1$

$2, 3, 4, \ldots$ $\quad (n-1)$

$0 \le size(S) \le n-2$

$\phi, 1, 2, \ldots n-2$

$g(i, S)$

Date: / / Page no:

k = size of set (S)

$(n-1)$ $\binom{n-2}{k}$ vertices

For each $(n-1)$ vertices we need to construct set of k size where total no. of elements that can be in set $S = n-2$

i

No. of sets computed

$\sum_{k=0}^{n-2} (n-1)\binom{n-2}{k} = (n-1)2^{n-2}$

we fixed 1, Δ i

remaining $(n-2)$ elements

For set size $= k$

we need k comparision k paths.

$$\sum_{k=0}^{n-2} \left((k)(n-1)\binom{n-2}{k} \right) = n^2 \cdot 2^n$$

k comparisions

No of sets, evaluation of k size set

No. of comparisions needed to evaluate for set of size k.

$|S| = k$ = size of S

Time Complexity : $O(n^2 \cdot 2^n)$

$(TC)_{DP} < (TC)_{\text{Brute Force}}$

Repeated fn calls $n-2$ $(n-1)2^{n-2}$ in recursive call tree

21.03.2023 Monday

Date: / / Page no:

DP

Overlapping Subproblems → Subtopics

0/1 Knapsack Problem

Given $x_1, x_2, \ldots, x_n$

Element x_i has a wt. of w_i and profit p_i

Sack capacity m

Determine the subset of items to be placed in the knapsack in order to maximize the profit.

	x_1	x_2	x_3
w	10	20	30
P	20	20	35

x_1 x_2 ... x_n

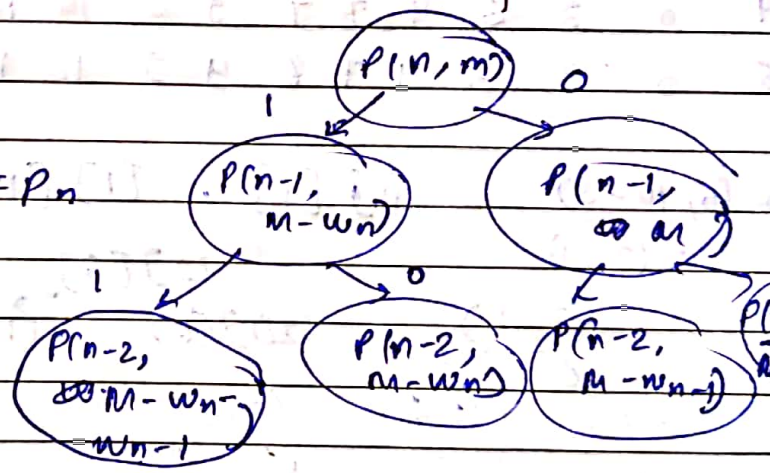
0 0 0 1

.

1 1 1 1 → 2^n

greedy works but not for all cases. $T(n) = 2T(n-1) + 1$

Brute force.



Case 1 : x_n (taken)

$x_1, \ldots x_{n-1}$, $m - w_n$, $P = P_n$

Case 2 : x_n (not taken)

$x_1, \ldots, x_{n-1}$

Overlapping subproblems. ✓

Optimal Substructure ✓

∴ DP soln will work.

Recurrence :-

Profit fn

$$P(i, c) = \begin{cases} 0 & , \text{if } i = 0 \text{ or } c \leq 0 \\ \max(P(i-1, c),\ P(i-1, c-w_i) + P_i), & \text{if } w_i \leq c \\ P(i-1, c), & \text{otherwise} \end{cases}$$

Start P(n, M)

Scanned with CamScanner

Bad Case:
Cutoff Node
Elements over
w capacity

Date: / / Page no:

Data Structure
Mind open
2D array
BST

$2n^2 \sim O(n^2)$
$3n^2 + 4n \sim O(n^2)$
Better as better coefficients

Top Down
Bottom-up is better than top down.

3)

	x_i	w_i	P_i
	1	3	4
	2	5	6
	3	5	5
	4	1	3
	5	4	5

$m = 10$

i \ M	0	1	2	3	4	5	6	7	8	9	10	
0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	4	4	4	4	4	4	4	4	4
2	0	0	0	4	4	6	6	6	10	10	10	6
3	0	0	0	4	4	6	6	6	10	10	11	5
4	0	3	3	4	7	7	9	9	10	13	13	3
5	0	3	3	4	7	8	9	9	12	13	14	5

$f(m, 10-3) = (0, 7)$

(3,1) → (2,1) → (2,0)

(3,6) → (2,6) → (2,1) + P_3

[1][1], [1][1-5]

[2][6] → [1][1] + 6

(3,5) → (2,5) → (2,0) + P_3
6
0 + 5
5

[2][5] → [1][5] → [1][5-5]
4
[1][0] + 6
= 6

[2][8] → (1,8) → (1,3) + 6
4
4 + 6

21.03.2023 Tuesday

Date: / / Page no:

BACKTRACKING

Solution vector $x_1, x_2, \ldots x_n$

↳ constraint / Bounding / criterion function $P(x_1, x_2, \ldots x_n)$

S_1 S_2 S_n

$m = m_1 \times m_2 \times \ldots \times m_n$

we need to check 'm' vectors

$P_i(x_1, x_2, \ldots x_i)$

Follows criterion function "true"

Does not follow criterion function "false"

$x_{i+1} \ldots x_n$

Backtrack → NO → Other choice x_i

If there is x_i's choice

$P(x_1, x_2, x_3, \ldots x_{i-1})$

Continue

Constraints

Explicit constraint

$x_i \geq 0$ $S_i = \{$ all +ve no. $\}$

$x_i = 0/1$ $S_i = \{0, 1\}$

$S_i = [0, 1]$

what values x_i can take.

Implicit Constraint

- how x_i relates to each other

eg: each x_i is distinct

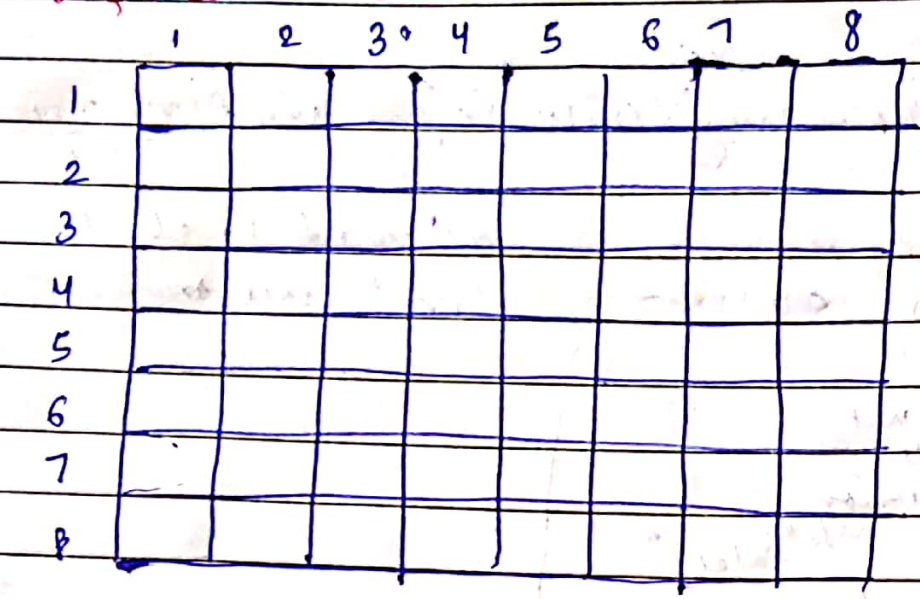
Not optimization problem

Date: / / Page no:

(Decision problem)

N-Queen Problem

8 Queen Problem



→ no two queens should be on same row,
→ same column
→ & on same diagonal

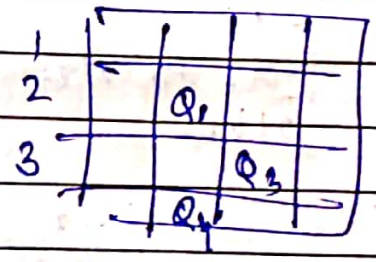
$q_1\ q_2\ q_3\ q_4 \cdots q_8$
↓ ↓ ↓
row1 row2 row

↙ Not in same row (by this)

Soln vector -
8 tuple $(x_1, x_2, \cdots x_8)$
each x_i = the col. no. on which i^{th} queen is placed on i^{th} row

Explicit constraint $1 \le x_i \le 8$ — 8^8 tuples

Implicit constraint each x_i should be distinct → (I)
- Permutation of vector $(1, \cdots 8)$ — 8! tuples (reduction)



$x_i = (i, j)$
$x_k = (k, l)$
$Q_2 = (2,3)$ $Q_3 = (3,4)$
i, j k, l
$i - j = 2 - 3 = -1$
$j - l = 3 - 4 = -1$

$Q_3 = (3,4)$ $Q_4 = (4,3)$
$i - k = 3 - 4 = -1$
$j - l = 4 - 3 = 1$

cond^n $|i - k| \neq |j - l|$ → (II)

More reduction of tuples

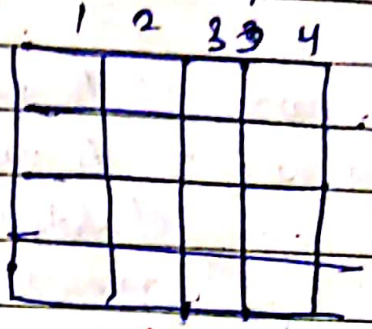
- Travelling Salesman → Optimization problem (min cost)
Hamiltonian path → decision problem (No weight)
N Queen problem → decision problem

Date: / / Page no:

4 Queen Problem

x_1, x_2, x_3, x_4

Level-i → choices for x_i



In brute force each parent will have N nodes

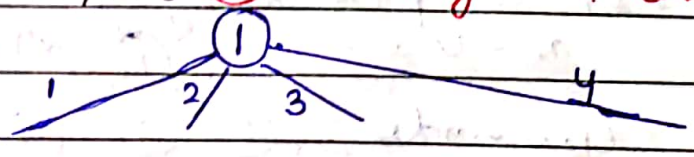
but here it will reduce due to implicit constraint

State based tree

(I) → distinct column violation

(II) → diagonal cond^n violation.

Can be in form DFS / BFS forms



$x_1 =$

DFS: As soon as a new child C of current E-node R is generated, C becomes the new E-node and exploration of R is suspended

↳ live node

List of live Nodes is stored in stack

Backtrack :- DFS + bounding functions to kill non-promising node

A tree can have atmost 1 E node at an instance but multiple live node.

Node No. → Choice No.

Solution state :- That contains all the variables but not necessarily answer state

that follows all explicit constraint

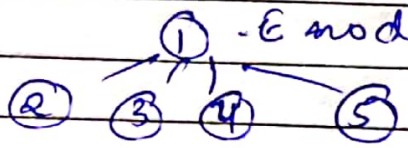
Date: / / Page no:

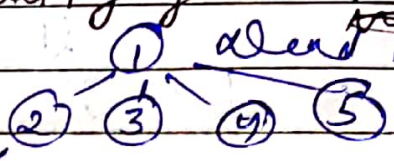
Types of Nodes in Stage based tree exploration

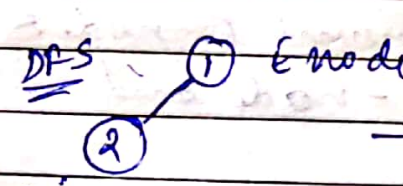
E-node → A node whose node is generated currently

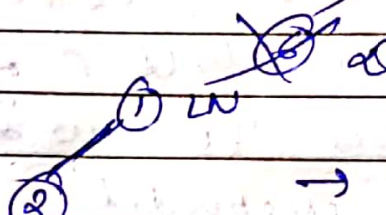
Live Node → A node which is generated but their children are not generated

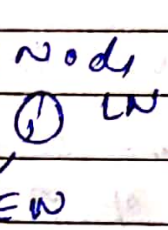
Dead Node → When the all the nodes are generated or not a possibility of answer.





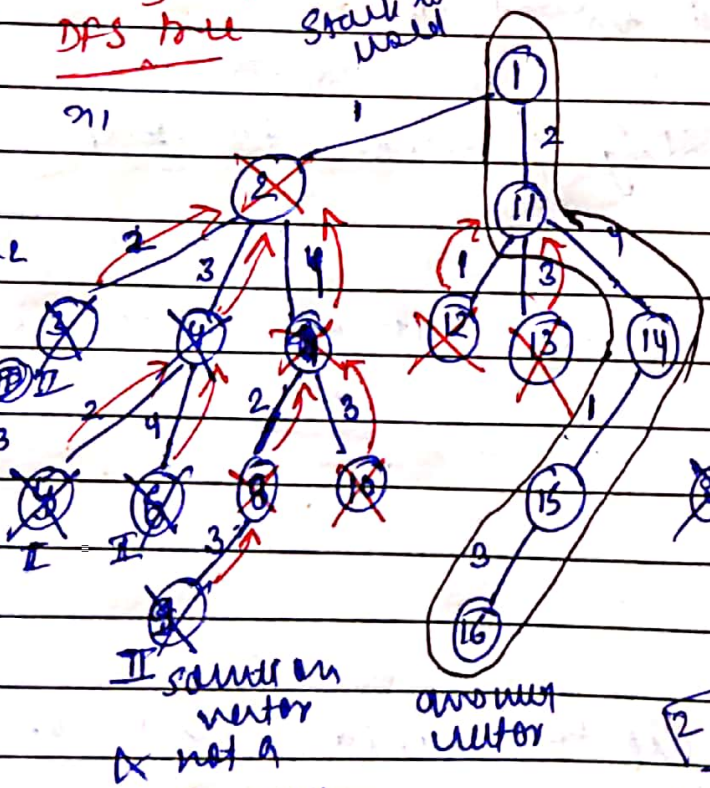




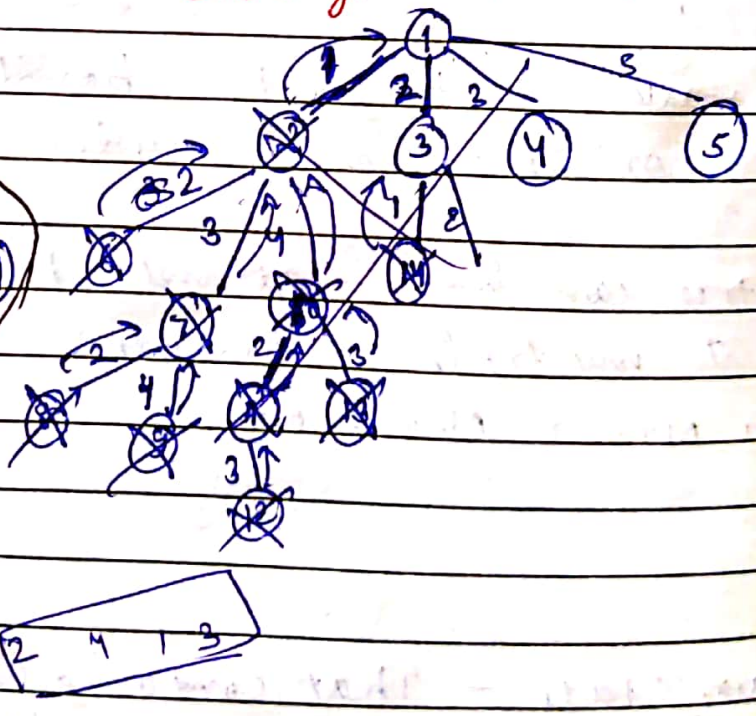


Queue is used

Based tree DFS full state space tree



BFS Stage Based Tree

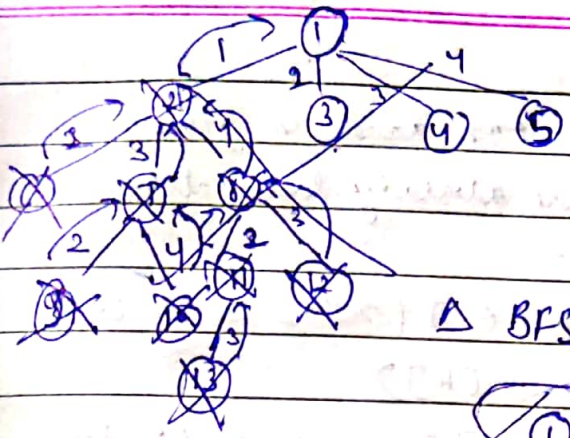


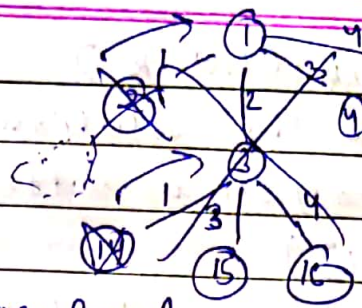
2 4 1 3

Depth Wise Node

List of live nodes are stored as Stack ⇒ DFS

Date: / / Page no:

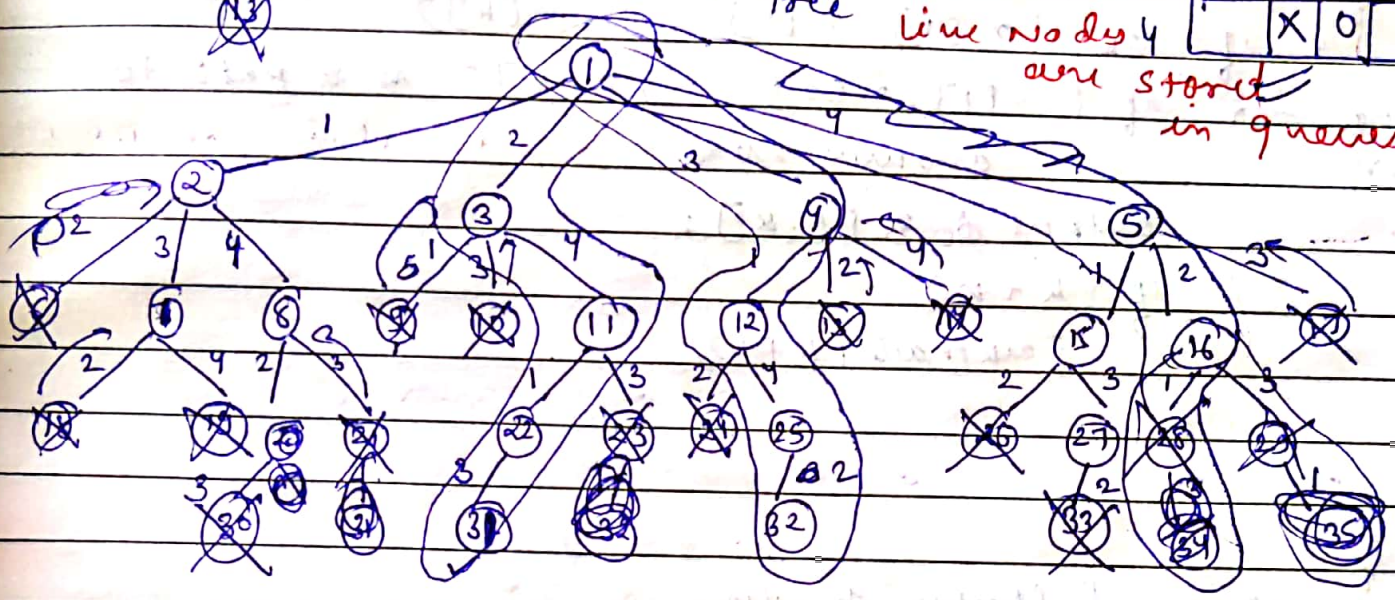




	1	2	3	4
1		0	X	
2	X			0
3	0			X
4		X	0	

Live Nodes are stored in queues.

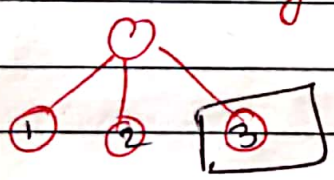
△ BFS Stage Based Tree



Ans: 2, 4, 1, 3 3, 1, 4, 2 ~~4, 2, 1, 3~~

△ Branch and Bound → BFS Stage Based tree.
level by level चलता है।
↳ use Bounding fn to kill Not desired nodes

D-Search ~~work by stack~~ Stack in live Nodes based wise



1	2	3

Stack

This node is explored first
(Not level wise then its children)

// call it mirror DFS

27.03.2023 Monday.

Date: / / Page no:

```
Void Backtrack (int k)
// x[1...n] is the solution vector and is global
// x[1..k-1] values of soln in vector have already been chosen
// T [x[1]..x[k-1]])
{ for each x[k] such that x[k] ∈ T(x[1]....x[k-1])
    { if Bk [x[1], x[2], ..., x[k]]
      { if (x[1], x[2], ..., x[k]) is a path to
                    answer node
          output ∀ x[1:k];
          if (k<n)
            Backtrack(k+1);
      }
    }
}
```

Bounding function

```
Bounding fn    // Checker function
  ↳ if simple : write here
  ↳ if complex : write as a separate fn
```

Backtracking algo : uses bounding fn + backtrack logic

NQueens Algo

```
void NQueens (int k, int n)
{ // x[1...n] →

for ( int i=1; i<=n; i++)
{       if (Place (k, i)) // Bounding function
          { x[k] = i;
            if (k==n)
              { output x[1:n]; }
```

i/p → n×n chessboard
o/p →

Date: / / Page no:

```
            else
                NQueens(k+1, n);
            }
        }
    }

boolean place(k, i)
// Returns true
//
//
{   for (j=1; j<k; j++)
        if (x[j] == i) || (abs(x[j]-i) = abs(j-k))
    // two queens on same column or same diagonal
            return false;

    return true; // renders when for loop ends
}
```

Same column

$x[j] = i$

previous रखी किसी भी Queen की column value के बराबर then return false

	row	col
j^{th} Queen	j	$x[j]$
k^{th} Queen	k	i
	$abs(d_2)$	$abs(d_1)$

if (equal) on diagonal ∴ return false

Q) Find all possible answer states. Ans: Draw complete stage based tree

Q) Find whether any solⁿ exists.
Draw Tree until we find first ans.

Date: / / Page no:

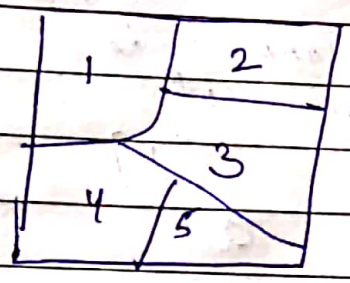
GRAPH COLORING PROBLEM

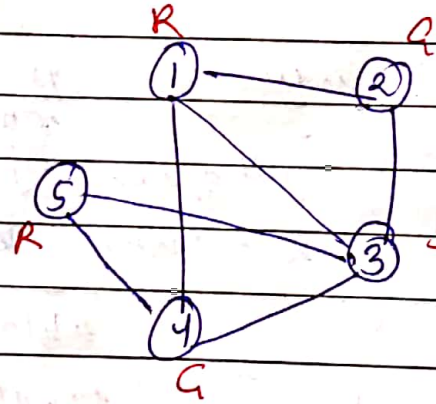
① is decision version
② is optimisation version.

① m- Colorability decision problem. (input : m)
(Chromatic Number)
Possible configuration for given m

② m - Colorability optimisation problem
m = ? need to be found, min. m such that we can color graph such that no two vertices are have same color

Application Planar graph
When we need to convert map into graph.





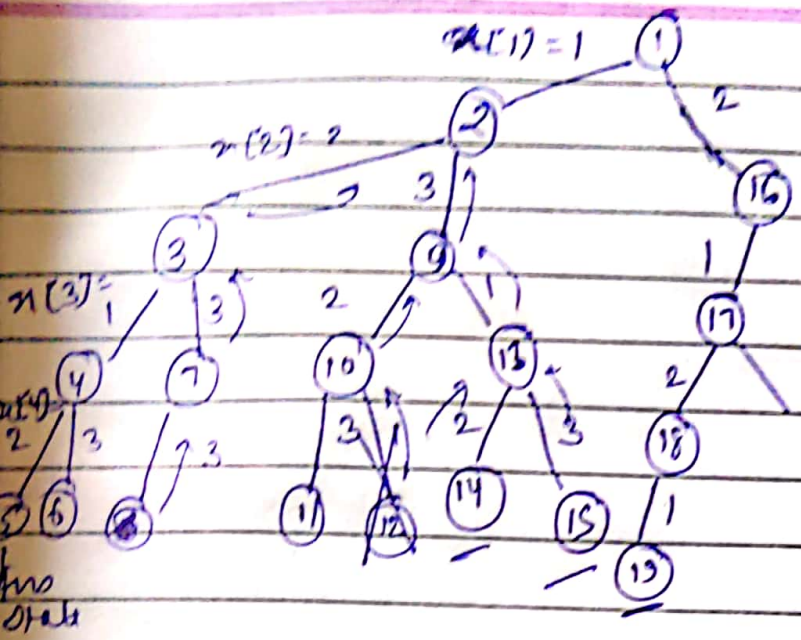
① Draw areas as nodes
② Add edges which having sharing neighbours

③ Check adjacency list of each node before making decision of color such that no two neighbouring vertex gets same color

NOTE:-
Minimum colors required to ~~draw~~ configure ~~colors~~ graph with no neighbouring vertices with same color in a 2D planar graph (drawn on paper) = 4

x[i] → color for i^{th} node

Date: / / Page no.



2.