```python
# write algo. for adding two matrices
# and find its time complexity
x = [[1,2,3],
     [4,5,6],
     [7,8,9]]
y = [[1,2,3],
     [4,5,6],
     [7,8,9]]
result = [[0,0,0],
          [0,0,0],
          [0,0,0]]
for i in range(len(x)):
    for j in range(len(x[0])):
        result[i][j] = x[i][j] + y[i][j]
for r in result:
    print(r)
```

Complexity - o(m * n)

```python
# generate fibonacci series of 10 terms and count
nterms = int(input("how many terms?"))
n1 = 0
n2 = 1
count = 0
if nterms <= 0:
    print("enter positive number : ")
elif nterms == 1:
    print("fibonacci sequence upto", nterms)
    print(n1)
else:
    print("fibonacci sequence")
    while count < nterms:
        print(n1)
        nth = n1 + n2
        n1 = n2
        n2 = nth
        count +=1
```

Perform linear and binary search to find 15 in a given list of numbers as below :

5 7 9 12 13 15 21 25

Count the number of comparisons in both the search methods.

Linear search

Compare 5 with 15 → Not a match (1st comparison)
Compare 7 with 15 → Not a match (2nd comparison)
Compare 9 with 15 → Not a match (3rd comparison)
Compare 12 with 15 → Not a match (4th comparison)
Compare 13 with 15 → Not a match (5th comparison)
Compare 15 with 15 → Match found (6th comparison)

Binary search

First, compare the middle element (13) with 15:

- 13 is less than 15, so search in the right half.
- (1st comparison)

In the right half, the middle element is 15:

- 15 is equal to 15, so we found the target.
- (2nd comparison)

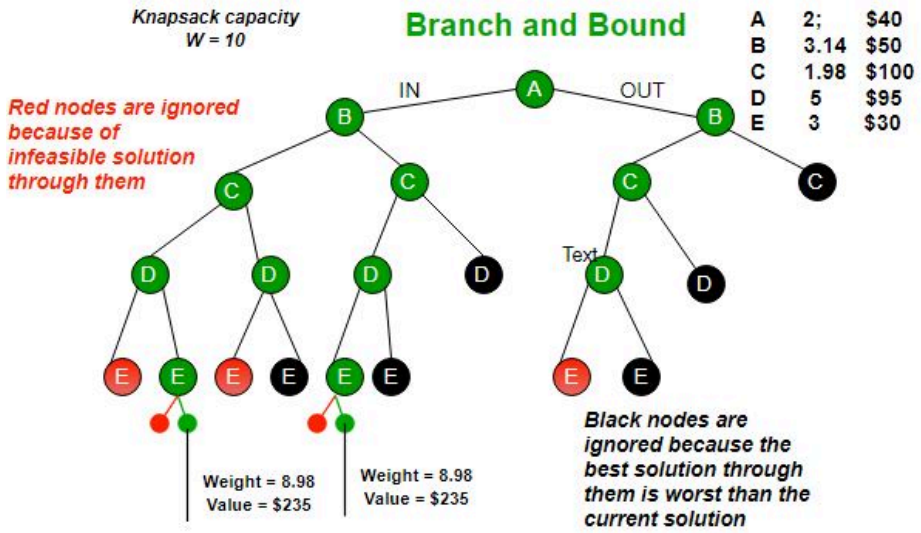
Linear Search Comparisons: 6
Binary Search Comparisons: 2

Branch

Branch me function ki problem ko subproblems me divide karta h

Bounding

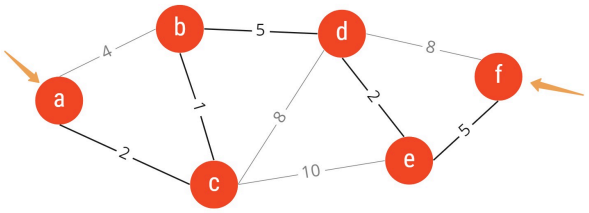
Bounding me functions ki subproblems ko eleminiate kya jata h
Bounding Jabtak chalta h jab tak optimal solution nhi mil jata



***Path**

Vertices ka sequence hota h

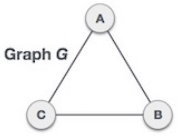
jo edges se connected hota h
Jha vertix repeated nhi hote

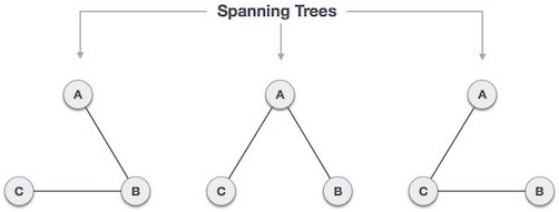


2 + 1 + 5 + 2 + 5 = 15

***SPANNING TREE**

Spanning tree ek graph ka woh subgraph hota hai
jo saare nodes ko connect karta hai bina kisi cycle ke
aur usme minimum possible edges hoti hain.



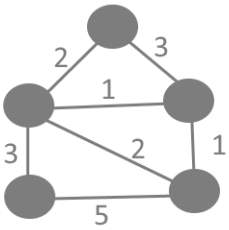


*** ST APPLICATIONS**

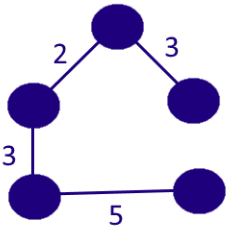
Network Design: Computer networks ko achhe se connect karta h
Minimum Cost Wiring: Electrical wiring ya phone lines ko asan tarike se jodta h
Routing Protocols: Internet par data ko order me bhejta h aur loop se bachata h
Cluster Analysis: Data ko groups mein baat ta h
Transportation Networks: Sadkon ya railways mein sahi rasta dhoondta h
Graph Algorithms: Algorithms jo best routes ya connections dhoondta h

***Minimum spanning tree**

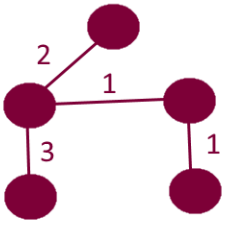
nodes ko jodta h
par sabse kam cost ka total karta h
jismein sabhi vertics ka total cost sabhi spanning tree ke total cost se kam hota hai.



Graph



Spanning Tree
Cost = 13



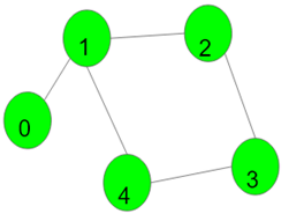
Minimum Spanning
Tree, Cost = 7

EUCLID

1. Euclid algorithm jo do numbers ka sabse bada common factor (GCD) nikalta hai
2. jahan bade number ko chhote number se baar-baar divide karte h aur remainder use karte hain, jab tak remainder zero na ho jaaye.

***Cycle in an undirected graph**

1. Cug mein ek aisa path hota hai jo kisi vertex se shuru hoke kuch edges se hote hue wapas usi vertex par aa jata hai
2. bina kisi vertex ko do baar visit kiye
3. Aur is graph me cycle hota h

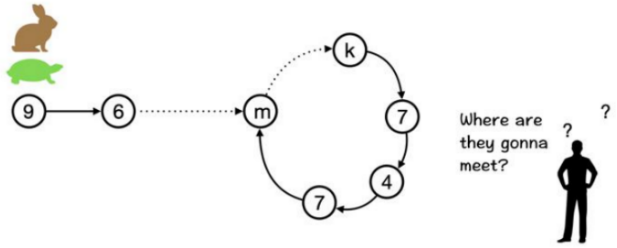


***COMPLETE GRAPH**

Cg mein Me vertices ka har pair edge se connected hota h

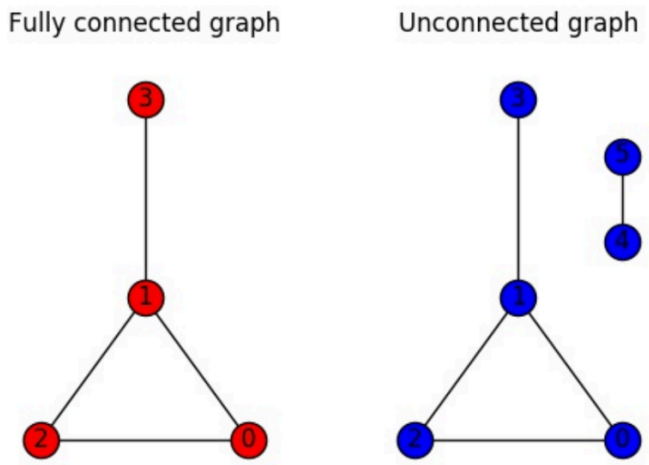
***CYCLE**

Ek path hota h
Jha first aur last vertices same hota h
Ye close loop hota h
Cycle me repeated veritcs hote h



***CONNECTED GRAPH**

1. har do vertices ke beech mein kam se kam ek path hota h
2. jo graph ke kisi bhi do nodes ek doosre se connect hote h
3. har point (node) kisi na kisi raste se doosre point (node) se jud sakta h
4. Cycle hota h



ALGORITHM

set of instruction hota hai
jo computer ki particular problem ko solve karta h
Adv
Easy to understand
Har problem ko step by step solve karte h

***ALGORITHM KA CHARATER**

1. Input:

User ke diye hue data per kam karta hai aur problem ko solve karta hai

2. Output:

Hamara algorithm problem ka solution produce karta hai

3. Finiteness:

Algorithm ko kuch steps ya kam time mein complete hona chahie

4. Generality:

Algorithm same problem ke various interface per kam karna chahie

5. Conciseness:

Algorithm simple or easy to understand hona chahiye

***MATHMATICAL PROBLEMS**

basic calculations like addition, subtraction, multiplication, and division
Search problems:
Ek item ko list mein se find karna
aur do cities ke bich mein se ek shortest route find karna
Sorting problems:
Numbers ki list ko assending aur descending order mein arrange karna chahie
Optimization problems:
Problem ka best solution find karna

Recurrence relation

1. Rr ek mathematical equation hota hai
2. Rr me kisi sequence mein har term ko uske pehle wale term ke saath relate karta hai
3. jo ek series (sequence) ke terms batata h

Notation

1. notation ek chota code hota hai

2. ye batata hai koi kaam kitni jaldi or efficient hota h
3. ye performance ko describe karta hai

Asymptotic Notetimes

1. ye batata h Ek algorithm dusre algorithm se kitna bhetar h
2. ye mathematics ka tool hota h
3. ye complexity ko represent karta h,
4. Ex. Big O, Big Omega, Big Theta

Big O

1. Algorithm ki efficiency ko input size ke sath compare karta h
2. Big o notation is used to describe asymptotic upper bound
3. worst case me use kiya jata h
4. Ek function maximum no. Of steps perform karta h

***Upperbound**

Ek sorting algo h
Algo ka time complexity input ki size k square k barabar hota h
Upperbound big o me hota h

Big Omega

1. best case me use kiya jata h
2. Ek function minimum no. Of steps perform karte h
3. Big Omega notation is used to describe asymptotic lower bound

Big theta

1. Big Theta average case me use Kiya jata h
2. Ek function average no. Of steps perform karte h

Complexity

1. ek measument hota hai
2. koi bhi operation karne me kitna time or space lagta hai
3. jisse programming ko effiecient banate hai

Time complexity

1. Hamara program kitna time leta hai ek specific task ko complete karne me

Space complexity

Program ko kitni extra memory ki jaroorat Hoti hai

Divide

Divide mein Ek problem ko chote sub problem me divide karte h

Conquer

Coquer mein Sub problem ko solve karte h recursion method se

Combine

Comibine mein Sub problem k solutions ko ek final solution me combine karta h

Bubble sort

1. Bubble sort ek sorting algorithm hai

2. array me 1st element aur 2nd element ko compare kiya jata h
3. agar wo galat order mein hain toh unhe swap kiya jata h
4. Aur aage 2nd aur 3rd element ko compare kiya jata h aur same rule follow kiya jata h
5. isme iteration steps ka use Kiya jata h

Quick sort

1. ek sorting algorithm hai
2. divide and conquer ka principal par kam karta h
3. array mein elements ko pivot, p aur q select kiya jata h
4. Aur inke according array ko divide aur merge karke organise kiya jata h

Selection Sort

1. Ek sorting algorithm hai
2. unsorted list me repeatedly minimum element ko dhundta hai
3. Aur beginning organise hota h sirf ek time me
4. Aur maximum bhi select karta h end me rakhta h vice verca

***Heap sort**

1. Ye selection sort ki tarah similar hota hai
2. Unsorted list mein minimum element ko find karta hai aur shuruaat mein rakhta hai
3. isme min or max operation hote h jo bhot fast hote h ye element ko add or remove karne k liye use hota h

insertion sort

1. insertion sort ek sorting algo. Hai
2. array k andar har ek element ko sorted sequence me rakkha jata hai
3. jese playing cards ko ek ek karke apni sahi position par rakhte h

merge sort

1. Merge sort ek sorting algo. Hai
2. Ms divid and conquar algorithm par kaam karta hai
3. Sabse phele isme data ko beech me se do parts me divide karte hai phir each part ko recursively divide karte rhete h jab tak individual element sepereate na hojaye
4. phir har individual part ko sort merge karke data ko sort karte h

Genetic Algorithm

1. ek search aur optimization technique hai
2. jo natural selection aur genetics ke principles ka use karkar optimization aur search problems ko solve karta ha

Optimization par focus karta h
Di hui problem ke multiple sol. Nikalta h

Ch Recursive algorithm

1. Ek particular problem ko chhote sub problem mein solve karta hai khud ki copy ko call karke karta hai

2. Ex. ek folder ke andar ke sabhi files aur folders ko dhoondhna, jaise ek folder ke andar doosre folders ho sakte hain, toh algorithm har folder ke andar jaake yeh kaam karta ha

Ch Strassen's Algorithm

1. "divide and conquer" ka principal follow karta h
2. Strassen's algorithm बड़ी do matrices को गणु करने का tez tarika का है जो standard tarike se से jyaada fast है
   Jisme p,q,r,s,t, u,v formules use hote h

***Karatsuba algorithm**

1. bade numbers ko small numbers me tezi se multiply karta h
2. ka divide and conquer algorthm ko follow karta h
3. Multiplication karne ka fast tarika h s3-s2-s1

Ch Greedy algorithm

हर टेप पर सबसे best aur immediate solution select karta h
ये हर बार सह जवाब nhi deta lekin kai problems ko solve karta h
Greedy about profit, Greedy about weight, Greedy about both

example

Prism, Kruksal, Traveling salesman algorithm

Searching algorithm

Ek dataset me Ek ya ek se jyada element ko sequence me search karte h
Types:
Linear, Binary search algorithm

Linear search

1. Ek searching algorithm hai
2. Array me har ek element ko sequence me check karte hai aur jab element mil jata hai tab use locate kar dete hai
3. Aur uska index return kardete h

Binary search

1. Ek searching algorithm hai
2. binary search divide or conquer ka principal follow karta h
3. unsorted array me middle me se element ko dhundna start karte hai agar element middle se chota hai to uski first half me dhundte hai nhi 2nd half me

Dynamic programming

ek problem solving approach h
Jisme big problems ko chote chote sub problems me divide karte h phir unko solve aur store karte h aur baad me reuse karte h

BFS

1. Sabse updar wale se nodes ko level by level explore karte h
2. shuruwat source node se hoti hai Destination node tak phocha jata h
3. Isme queue ka istemal hota hai
4. Back front

DFS

1. Sabse upar wale nodes se shuru karte h
2. aur uske adjacent nodes ko puri depth tak explore karte h
   phir doosre adjacent nodes ki taraf badhte hain
3. Isme recursion or stack ka use hota hai
4. top

Dijkstras Algorithm

Map k andar do place के beech mein shortest path find karta h

Applications:

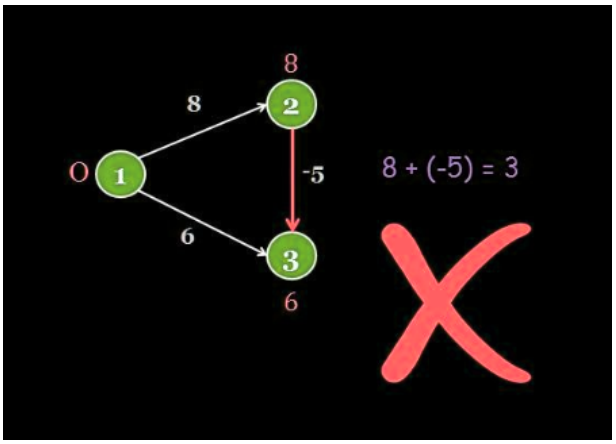
Game development, Circuit design, Robotics, Transportation, Find location in map

Purpose: To find the shortest path from a starting point (source) to all other points in a weighted graph.

1. Initialize:
   - Set the distance to the source node as 0 and all other nodes as infinity (∞).
2. Visit Neighbors:
   - Look at the current node's neighbors and update their distances if a shorter path is found
3. Mark as Done:
   - Once all neighbors are checked, mark the current node as visited.
4. Repeat:
   - Pick the next unvisited node with the smallest distance and repeat steps 2 and 3.
5. Finish:
   - Continue until all nodes are visited.

***Ch Dijkstra algo doesnt work on negative weight**

Dijsktra algo negative weight k saath kaam nhi karta h kyuki
Path par negative weight nodes ka revisit rheta h
Jisse ye infinity loop me stuck ho jata h
Aur optimal shortest path nhi mil pata



Prims algorithm

1. prism algo. minimum spanning tree hota h
2. Mein kisi bhi vertex se shuru karte h

3. Graph mein Minimum edges ka set find karta h
   jo sabhi vertics se connected hota h
4. Vetics ko jodte hue mst banate h

Ch Applications electrical wiring, traffic management, fraud detection, sales man, social media analysis, e commerce recommendation

Kruskal's Algorithm

1. ek graph ka minimum spanning tree hota h
2. Sabse chote edges se shuru karte h
3. Minimum cost find karta h
   Jo nodes se connected hota h
4. jo edges ko jodte hue mst banate h
   weight ke ascending order mein select karke bina cycle banaye graph ko connect karta.hai.

Ch Applications network design, circuit design, robotics, lan - local area network

Tree traversal

1. jab tree ko explor karte hai tab tree ko specific order me visit karte hai
2. ise 2 category me classified kiya jata h
3. har node ko sirf ek baar visit karte hai

Skip list

sorted list h, jo linked list se better h or jaldi search performance karta h
Ye linked list ki tarhe kaam karta h
एक डाटा structure h jo element ko dusro se jaldi dhundta h

hash function

hash algorithm, special function h jo kisi bhi data ko short or fixed / unique code me badal deta h

Encryption Algorithm

jo data ko unreadable bana deta h ya gupt kar deta h
taki use sirf authorized log hi access kar saken ya pad sake

cryptographic algorithm

data ko unreadable code me badal deta h
security ke sath bina authorisation k convert karta h

Ch Radix sort algorithm.

Radix sort is a digit-by-digit sorting technique
ye numbers ko unke individual digits se sort karta h
shuruwaat sabse chote waale digit se karte h
numbers ko ek-ek digit ke hisaab se sort karta hai

Direct recursive algorithm.

ek function sidhe khud ko call karta hai.

inDirect recursive algorithm.
Ek function khudko sidha call nhi karta
balki dusre function se apne aap ko call
karwata h

Control abstraction
Program k control ki complex chijo ko chupata h aur asan process provide karta h

Ch *Feasible solution
problem k solution ki conditon ko pura karta h

***optimal solution**
kisi problem का sabse best solution होता है
जो सभी condition ko pura karta h aur सबसे accha outcome देता है
kam resourse me best outcome deta h

***backtracking**
isme alag alag options try kiya jata h
aur agar valid solutoin nhi mil pata to backtrack kiya jata h
dfs ka rule follow karta h
Har possible combination ko search karte h
Iske solution ko tree ki form me represent karte h
Ex chess

implitcite
jo clearly bataya nhi jata lekin algoridam ke kam karne ke tarike se samajh jata h
explicite
jo sidha bataya jata aur algoridam ke kam karne ke tarike se samajha jata h

Horner's rule
Horner's rule polynomials (x wale expressions) solve karne ka short tarika hai
no. Of multiplications ko reduce karta h

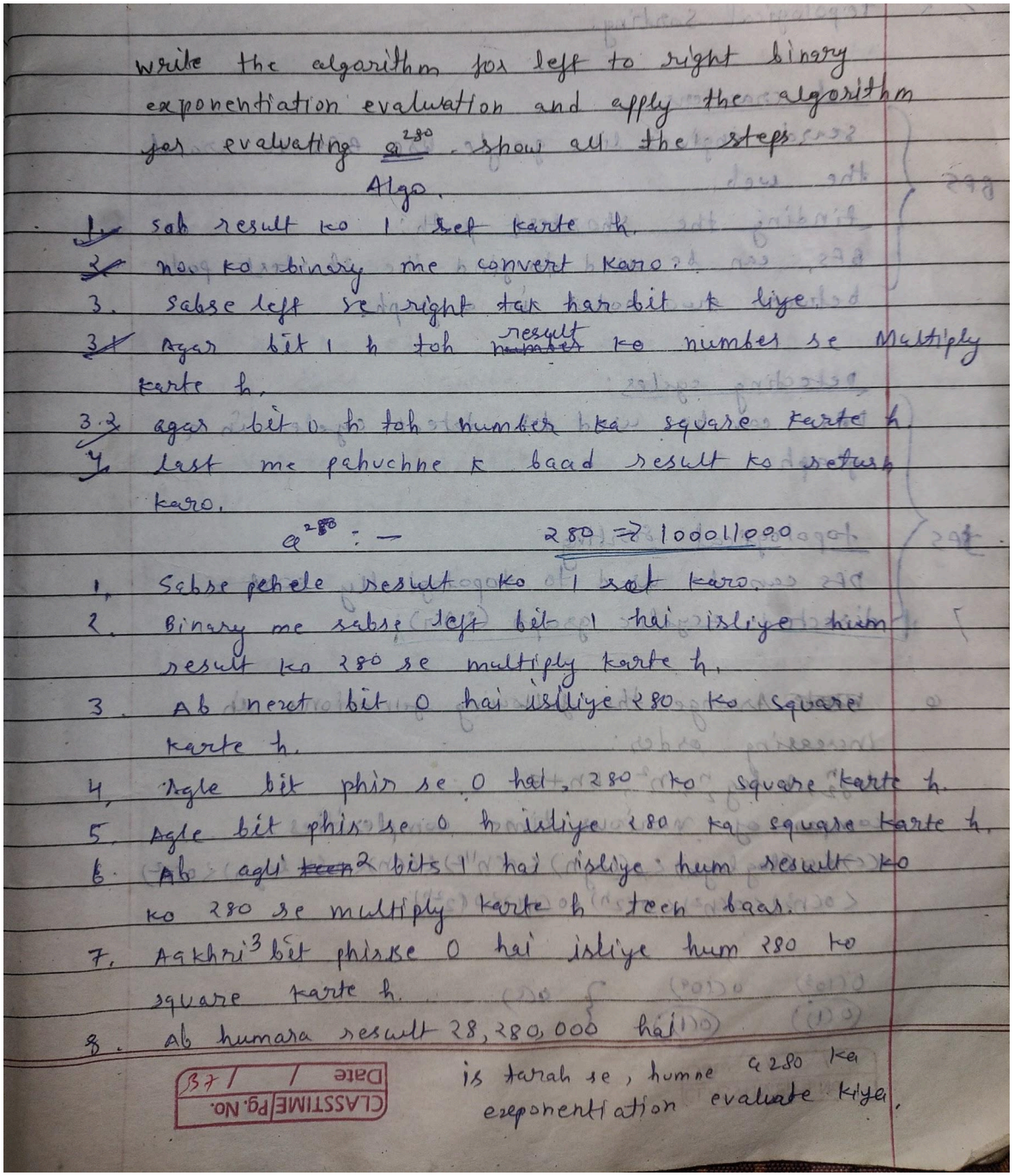
dynamic tree
tree k structure ko change aur update kiya ja sakta h

***Static tree**
tree k structure ko change nhi kiya ja sakta h but iske andar information ko change kiya ja sakta h
jiske node aur connection fixed hota h element ko remove nhi kar sakte h

***Ch quick sort worst case**
Quicksort का worst case तब होता है जब हर बार pivot चननु me सबसे छोटा या सबसे बड़ा element chun liya jata h

Belford

Casemap



Write the algorithm for left to right binary exponentiation evaluation and apply the algorithm for evaluating a^{280}. Show all the steps.

Algo.

1. Sab result ko 1 set karte h.
2. no. ko binary me convert karo.
3. Sabse left se right tak har bit k liye.
- 3.1 Agar bit 1 h toh result ko number se Multiply karte h.
- 3.2 agar bit 0 h toh number ka square karte h.
4. Last me pahuchne k baad result ko return karo.

a^{280} :— $280 \Rightarrow 100011000$

1. Sabse pehele result ko 1 set karo.
2. Binary me sabse left bit 1 hai isliye hum result ko 280 se multiply karte h.
3. Ab next bit 0 hai isliye 280 ko square karte h.
4. Agle bit phir se 0 hai, 280 ko square karte h.
5. Agle bit phir se 0 h isliye 280 ka square karte h.
6. Ab agli 2 bits 1 hai isliye hum result ko ko 280 se multiply karte h teen baar.
7. Aakhri 3 bit phirse 0 hai isliye hum 280 ko square karte h.
8. Ab humara result 28,280,000 hai

is tarah se, humne a^{280} ka exponentiation evaluate kiya.

DFS aur BFS k applications

BFS

web crawlers
search engines like google use BFS to crawl the web.

finding the shortest path:
BFS can be used to find the shortest path between two nodes in a graph.

DFS

detecting cycles:
DFS can be used to detect cycles in a graph.

topological sorting:
DFS can be used to topologically sort a directed acyclic graph (DAG).

Write the names of the following symbols:

Θ = theta used in Big O notation upper bound

Ω = omega used in Big notation lower bound

$\forall$ = for all

$\in$ = Belong to

8. $O(n \log_2 n)$ is better than $O(n^2)$ but not as good as $O(n)$.

(i) $O(n \log n)$ is better than $O(n^2)$.

True

because $O(n \log n)$ is ~~is~~ grow slower than $O(n^2)$ and meaning it ~~is~~ more efficient.

(ii) $O(\log n)$ is not as $O(n)$.

True,

Because $O(n \log n)$ grows faster than $O(n)$. meaning it is les efficent.

statement is ~~true~~.