Date
DELTA Pg No.

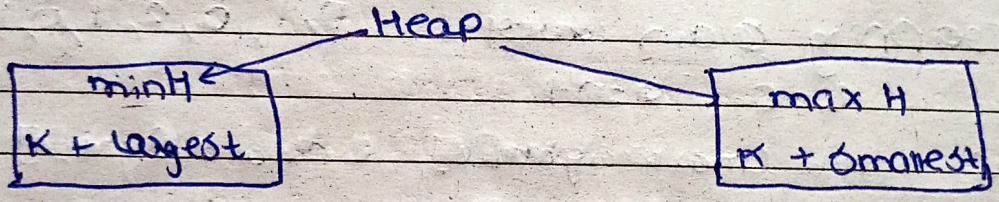
Binary Search

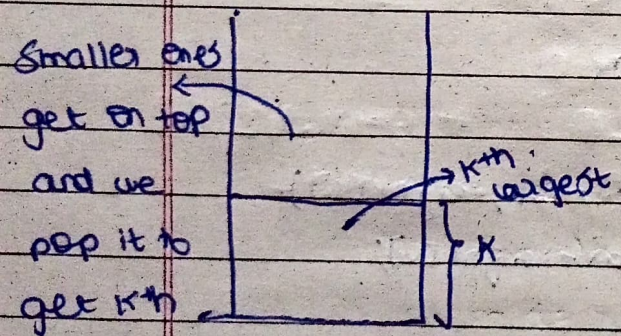
~~Ist~~

1) → Two pointers s, e
2) → while loop (s <= e)
3) → mid = s + (e-s)/2
4) → mid == target return mid
5) → nums[mid] > target e = mid-1
else s = mid+1

Heap

1) identification
2) Smallest / largest } agar dono nai to question heap ka nai





max Heap

priority_queue <int> maxh;

min Heap

priority_queue <int, vector<int>, greater <int>> minh;

if there is pair we can use

typedef pair <int, pair<int, int>> ppi;

Date
DELTA Pg No.

Sliding window

identify → subarray + target / min/max

① define i, j = 0
② max/min/target
③ while loop (j<n){
④ sum for target ++
⑤ while(sum>=target)
minf = min(minf, j-i+1)
sum -= nums[i]
i++
}
j++

→ Trapping rain water

0 1 2 3 4 5 6 7 8 9 10 11
0, 1, 0, 2, 1, 0, 1, 3, 2, 1, 2, 1

l=0 r=n-1

int lmax = height[l] rmax = height[r]
int ans = 0
while (l<r){
if(lmax < rmax){
l++;
lmax = max(lmax, height[l])
ans += lmax - height[l];
else{
r--
rmax = max(rmax, height[r]);
ans += rmax - height[r];
}
}
return ans

ans = -1

Date
DELTA Pg No.

Bit manipulation

→ & operator

let a=5 b=6 5=101 6=110

a&b 101
110
100

& operator
all 1 → 1
any 0 → 0

→

or operator
any 1 → 1
all 0 → 0

^ or ⊕, ^ operator
even 1 → 0
odd 1 → 1

→ binary into integer

$\overleftarrow{100} = 0\times2^0 + 0\times2^1 + 1\times2^2 = 4$

$\overleftarrow{101} = 1\times2^0 + 0\times2^1 + 1\times2^2 = 5$

→ Recursion

1) Express on term of i, j.
4 Explore all the paths ↙ ↕ ↘
4 max/sum/min or other

→ Tabulation

① Base case
② form the nested loop
ind → (1 → n-1)
target → (1 → target)
③ look at the changing parameters & write nested loops
④ Copy the relation.

Date
DSA Pg No 2

	0	1	2
0	1	2	5
1	3	1	1
2	3	3	3

	0	1	2
0	-1	-1	-1
1	-1	-1	-1
2	-1	-1	-1

solve(3, points, 0, 0, dp)

int right = 0

right = solve(n, points, 1, 1, dp) + points[0][1]

down = 0

down = solve(n, points, 1, 2, dp) + points[0][2]

down = solve(2, 2, 0, dp) + 3 → 6

right = 0

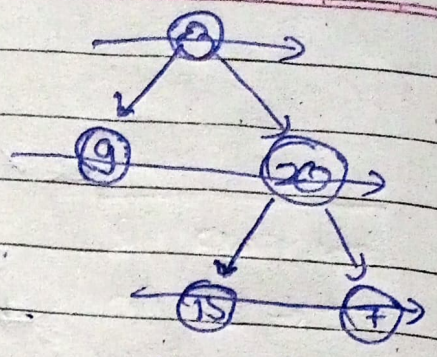
down = solve(2, 2) + 1 → 4

solve(1, 0) + points[1][0]

solve(2, 1) + points[

Date
DELTA Pg No.

Graph - (BFS)



level wise traversal

```
vector<vector<int>> ans;
if(root == null) return ans;
Queue<TreeNode*> q;
q.Push(root);
while(!q.empty()){
    int n=q.size();
    vector<int>p;
    for(int i=0 i<n)
    TreeNode* node= q.front();
    q.pop()
    if(node->left != null) q.Push(node->left);
    if(node->right != null) q.Push(node->right);
    p.push_back(node->val);
    }
    ans.push_back(p)
return ans
```

Identifiable/usage

- Kadane algo → use to compute maximum in circular subarray
  ↳ use to compute subarray sum (338-1)
- Heap → K + smallest / largest
  ↳ min → K + largest (smaller one get on top)
  ↳ max → K + smallest
- Sliding window → ① Subarray / substring
- Backtracking → ① Choice + Decision
  ② All combination (asked)
  ③ Constraint recursion
  ④ Number of choice (high)
  ⑤ Size of constraints
  ↳ 1 < n < 10 or single digit
  ⑥ Don't be greedy
- Binary search → from constraints
  ↳ 10^9

→ dp :→ so many choice + constraints

↳ base recursion
↳ convert all variable in term of i, j
↳ traverse all possibility
↳ find sum / max

like $1 < n < 1000$ or $1 < n < 2000$
$1 < n < 45$, $1 < n < 80$
$1 < n < 8$ like these
or $1 < n < 10^6$

→ graph → (component word) → or something like map or graph
constraints like dp but here 4 choice

Searching/Sorting ~~DP~~ Important question/Algo

→ Binary Search
→ Quick Sort
→ merge Sort
→ order Statistics
→ KMP algorithm
→ Rabin Karp
→ Z's algorithm
→ Aho corasick String matching
→ counting Sort
→ manacher's algorithm

KMP algorithms

txt = "ABABCABCABABABD"
Pattern = 'ABABD

Jo humari naiv approch thi vo bhi same work karti thi but ye hume Pichli ~~Pichle~~ ko Store bhi karta hai like AAAABCD & we have to find AABC in naive ~~#~~ vo 0 index se start karta nhi mila ~~second~~ pr jata ~~first~~ nhi mila ~~first~~ 2nd pr ~~3rd~~ pr jakar AABC ye milta but in KMP ye AA at 0 declare nhi hai to 0 index wala A ko remove karke 2nd Index wala A add kardge similarly Sliding window approch hai.

→ So time complexity of naive approach is O(n×m) which give me TLE
& KMP TC = O(n+m)

text = 'ABABABD'
Pattern = 'ABABD
0 1 2 3 4

LPS = | 0 | 0 | 1 | 2 | 0 |
0 1 2 3 4

from i=0 → text & j=0 → Pattern
i==j i++ j++
i=1, j=1 i==j i++ j++
:
i=4 j=4 i≠j so ab (j-1) ki sabse
bada prefix konsa hai tele pass 2 (len) so
j = j-2 & continue approach

code
lps[0] = 0, length = 0 i = 1
while (i < m) {
    if (pattern[i] == pattern[length]) {
        length++
        lps[i] = length
        i++
    }
    else {
        if (length > 0) {
        length = lps[length-1]
        }
        else lps[i] = 0 i++;
    }
}

kmp
i=0, j=0
while (i < N)
    if (text[i] == pattern[j]) {
        i++, j++; }
    if (j == M) { // found & pattern finished
        result . push_back (i-j+1)
        j = lps[j-1];
    }

else if (pattern[j] != text[i]) {
    if (j != 0) {
        j = lps[j-1];
    } else
        i++
}

DP types & Pattern & Hints

1D DP → Simple recursion from n to n-1.
Simple recursion acc to question

2D DP on Grid.
↳ Reverse order from n-1, m-1 to 0,0 & or left & up (taken)

DP on subsequence → If all above taken current element or not taken