Collecting water 3D

2

3 6

4 3 2 4 3 3

2 0 0 3 0 2

2 2 2 2 1 1

3 6

4 3 2 4 3 3

2 0 1 3 0 2

2 2 2 2 1 1

Output

5

4

Water will be accumulated only till the minimum boundary so we place all border element to a min heap. With tht element we check neighbours above which we can fill water then the element is marked visited and the neighbour is the boundary element then we repeat same.

(a)

arr: 2 3 17

Product of XOR of all distinct pairs

```
ans = 1   N = min (N, 3001)
for (i → 0 : N)
    for (j → i+1 : N)
        ans *= (a[i] ^ a[j])
print(ans)
```

2, 3, 1, 7

010 011 001 111

1×3×5×2×4×6

$1 \le T \le 10$

$1 \le N \le 10^5$

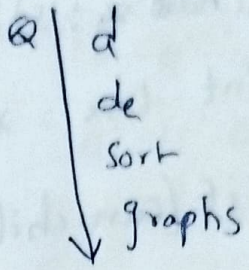
$1 \le arr[i] \le 3000$

(b) Virus Expansion

06/02/2025

- Given a list of words and Q queries

data	ascending
dque	descending
deck	root
dequeue	tree
sort	graph,
reversesort	cycle

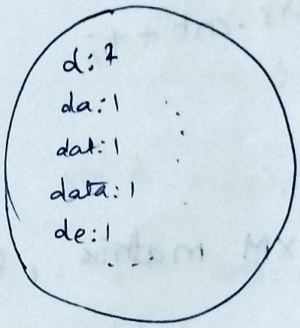


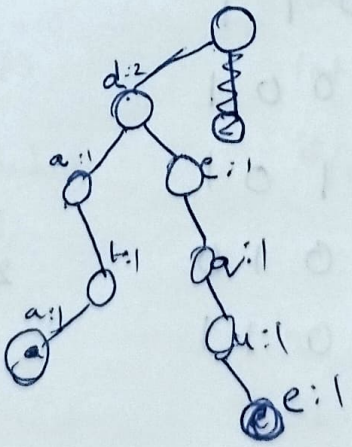
Find the words that have Q as a prefix

Solutions

Brute Force

1. Q×N×M, 1
2. sort
3. HashMap
   M×N + Q×M, M×N
4. Rolling hash
   M×N + Q×M, M×N
5. Trie
   M×N + Q×M, M×N





```
void insert(String s) {
    Node curr = root;
    for(char x : s){
        int idx = x-97;
        if (curr.child[idx] == null) {
            curr.child[idx] = new Node();
        }
        curr = curr.child[idx];
        curr.cnt++;
    }
}
```

- Given NXM matrix, Find the distinct rows.

1	0	0	1	0
1	1	0	0	1
1	1	0	0	0
1	0	0	1	0
0	0	1	0	1
1	1	0	0	1

Solutions

1. Brute force $M \times (N \times M)$, visit all rows, compare every new row with visited rows, if not same inc++.
2. Trie $N \times M + N \times M$,

- arr_N : 8 16 25 20 5 1 3

Find the max value of arr[i] ^ arr[j], i != j

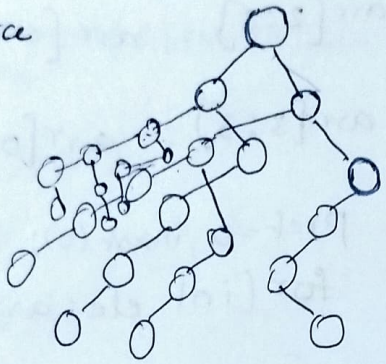
Solutions

(1) Brute force

N^2, 1

(2) Trie

N×31 + N×31 / N×31



16	8	4	2	1
0	1	0	0	0
1	0	0	0	0
1	0	0	0	1
1	0	1	0	0
0	0	1	0	1
0	0	0	0	1
0	0	0	1	1

```
void insert(int num){
  Node curr = root;
  for(int i=31; i>=0; i--){
    if(num>>i & 1 == 1){
      if(curr.child[1] == null)
        curr.child[1] = new Node();
      curr = curr.child[1];
    } else {
      if(curr.child[0] == null)
        curr.child[0] = new Node();
      curr = curr.child[0];
    }
  }
}
```

```
void maxXOR(int num){
  int ans = 0
  Node curr = root;
  for(int i=31; i>=0; i--){
    int bit = (num>>i & 1)
    if(curr.child[bit^1]){
      ans = ans | (1<<i);
      curr = curr.child[bit^1]
    } else {
      curr = curr.child[bit]
    }
  return ans;
}
```

Main fn

```
TrieNode root = null;
for(int ele : arr){
  insert(ele);
}
ans = 0;
for(int ele : arr){
  ans = max(ans, maxXOR(ele))
}
```

- arr_N Find the max subarray XOR

Solution

① Brute force

$N^2 \times N, 1$

② Carry forward

$N^2, 1$

③ Tric

$N \times (31+31), 31 \times N$

len N=3

arr[0:3] = arr[0:3] ^ 0

arr[1:3] = arr[0:3] ^ arr[0:0]

arr[2:3] = arr[0:3] ^ arr[0:1]

arr[3:3] = arr[0:3] ^ arr[0:2]

```
pref=0, insert(0);
for(int ele: arr){
    pref ^= ele;
    insert(pref);
    ans = max(ans, maxXOR(pref));
}
```

13/08/25

* String Partitioning

```
bool f(s, root, idx){
IF(idx == len(s) )return True;
Node curr = root;
for(int i=idx; i<=len(s); i++){
    if(root.child[s.charAt(i)-97] != null){
        root = root.child[s.charAt(i)-97];
        if(root.flag && f(s, curr, i+1)){
            return T;
        }
    } else break;
}
return false;
}
```

Dynamic Programming

```
int fibR(int N){
  if(N==1 || N==2)
    return 1;
  return fib(N-1) + fib(N-2);
}
```

$O(2^N)$

```
int fibI {
  int a=1, b=1;
  for(int i=3; i<=N; i++){
    c = a+b;
    a = b;
    b = c;
  }
  return c;
}
```

$O(N)$

Optimized recursion

```
int[] dp = {-1};
int FibR(int N){
  if(N==1||N==2)return 1;
  if(dp[N] != -1) return dp[N];
  return dp[N] = fibR(N-1) + fibR(N-2);
}
```

$O(N)$

When DP?

1. Overlapping subproblems
2. dp state
3. dp expression
4. dp table size
5. Base condition
6. Time complexity
7. Space complexity
8. Space optimization

29/11/24

HEAPS

Data Structure	Insert(x)	getMin(x)	delMin(x)
Arrays	1	N	N
Sorted Array	N	1	1
Stacks	1	N	N
Queues	1	N	N
SLL	1	N	N
Sorted SLL	N	1	1
BST	H	H	H
BBST	$\log_2 N$	$\log_2 N$	$\log_2 N$

Heaps are used to store the minimum element as the root node of the tree, so that we can get the min element of tree at time O(1).

It is implemented as a CBT, But we can implement the same using Array list.

Parent node(i) = (i-1) // 2

child node(i) = (2*i+1)

getMin() → O(1)

delMin() → O(H) ≈ O($\log_2 N$)

④ Given an array of size N, find k^{th} smallest elements

A_N : -2 10 3 5 -10 6 4 8

k = 4

Solutions

1. Sort the elements and print k^{th} element

$O(N \log_2 N + k, 1)$

2. Use MinHeap

$O(N \log_2 N + k(1 + \log_2 N))$

3. Treeset $O(N \log_2 N + k, N)$

4. Add first 'k' elements to heap and from then for every element getMax(), compare with incoming element and delete max and insert in coming (Max Heap)

PriorityQueue<> pq = new PriorityQueue<Collections.reverseOrder>();

To implement max heap with the help of min heap insert the "-ve" values of all elements into heap.

$T(④) = \overset{\text{Add to max heap (K elements)}}{\rightarrow} k \log_2 k + (N-K) * (\overset{\text{Delete max}}{\rightarrow} 1 + \log_2 k + \overset{\text{Percolate up}}{\rightarrow} \log_2 k) + k \log_2 k$ (Remaining element, Insert, Print)

$= k \log_2 k + (N-K) * (2 \log_2 k) + (N-K) + k \log_2 k$

$= (N-K) + 2N \log_2 k$

⑤ A_N : 2 10 13 23 29 36 43 49 50

K = 4

→ sort the k-sorted array.

A_N : 29 15 10 43 2 23 36 50 49

For every element, it can be swapped with the adjacent k elements is what happened, we need to get sorted array

For getMin : $T(1)$

delete : $\log_2 N$

Solutions :

① $N \log_2 N$, 1 (sorting)

② $N \log_2 N + N$, N (Tree set)

③ $N \log_2 N + N \log_2 N$ (Min Heap)
(Insert) (delete)

④ Take Minheap of size k+1, print the min elements, and add incoming element every time delete the min element

$T: (k+1)\log_2 k + (N-k-1)(\log_2 k + k \log_2 k)$
Delete Insert

$= N \log k$

⑤ Given a matrix $mat_{N \times M}$, every row is sorted. Print the entire matrix data in a sorted order.

P1	5	7	9	11	13
P2	-10	-6	2	3	6
P3	-1	0	20	21	29
P4	-3	2	10	18	20
P5	6	9	10	15	25

Solutions

① $NM + NM \log_2(NM)$, NM
Store in array and sort

② $N(MN)$, N

③ Heap Min
$O(\log_2 N * (NM), N)$

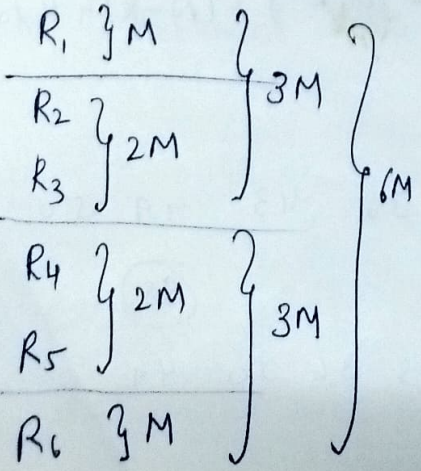
④ $O(N^2 M, NM)$

⑤ $\log_2 N * O(NM)$

③ Take a min heap & store N row elements
heap shld store [ele, [i,j]]
For every node, del Min
$O(\log_2 N * MN, N)$

④ Use Merge sort
Add every row and Merge

⑤



$T(N) = T(N/2) + T(N/2) + O(NM)$

$= 2T(N/2) + O(NM)$

$= \log_2 N * O(NM)$

20/2/25

Stair Case

A person standing at 0^{th} step wants to reach N^{th} step. moves (1, 2)
Find no. of ways to reach N^{th} step

N=4 1+1+1+1

1+1+2

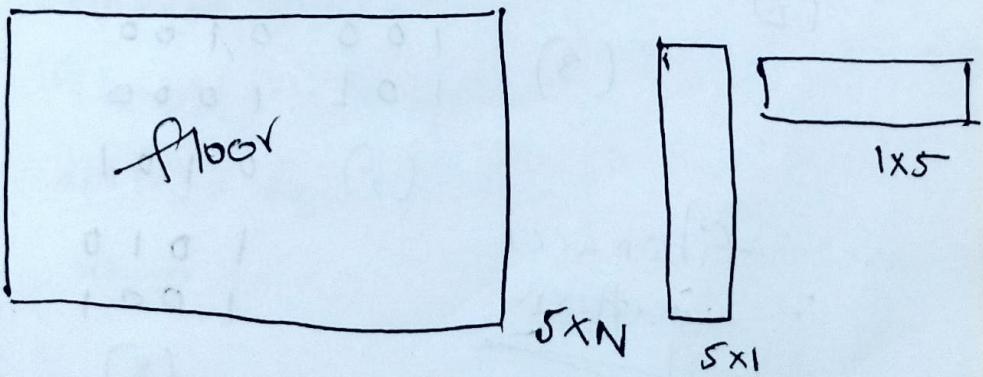
1+2+1

⋮

```
int f(N){
  if(N==1||N==2) ret N;
  ret f(N-1) + f(N-2)
}
```

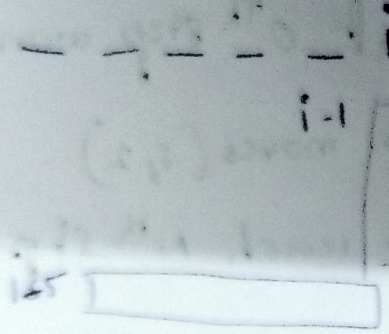
$i-1$ step $\xrightarrow{+1}$ i^{th} step

$i-2$ step $\xrightarrow{+2}$ i^{th} step



Entire floor must be covered with tiles
No tile shld be broken

dp → No. of ways to tile the floor till i^{th} index




```
if(i<5) return 1;
return dp[i-1]+dp[i-5];
```

TC → O(N) SC → O(N)

Space Optimization → O(6)

- Given an integer N, count no. of binary strings having len N (having no adjacent 1's)

N=1	N=2	N=3	N=4
0	00	000	0000
1	01	001	0001
(2)	10	010	0010
	(3)	100	0100
		101	1000
		(5)	0101
			1010
			1001
			(8)

fibonacci pattern

```
int f(n){
  if(n==-1) ret 2;
  ret fo(n)+f1(n);
}
int fo(n){
  if(n==1) ret 1;
  xt f1(n-1)+fo(n-1);
}
```

```
int f1(n){
  if(n==1) ret 1;
  xt fo(n-2);
}
```

//dp

```
int f(n){
  dp0[n+1]={-1};
  dp1[n+1]={-1};
 ret fo(n,dp0,dp1)+f1(n,dp0,dp1);
}
 int fo(n,dpo,dp1){
    if(n==1) return 1;
    if(dpo[n]!=-1) ret dpo[n];
     dpo[n] = fo(n-1)+f1(n-1);
     return dpo[n];
  }
```

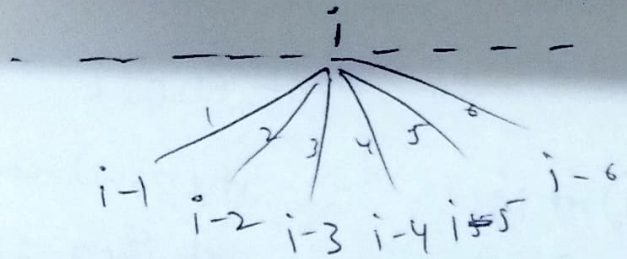
```
int f1(n,dpo,dp1){
 if(n==1) return 1;
if(dp1[n]!=-1) xt dp1[n];
dp1[n] = dpo[n-1];
 return dp1[n];
```

TC → O(N) SC → O(N) → O(3)

Base dpo[0] = 1 , dp1[1] = 1

ans = dpo[N] + dp1[N]

Q) Given a 6-sided dice roll find how many ways to reach N ?



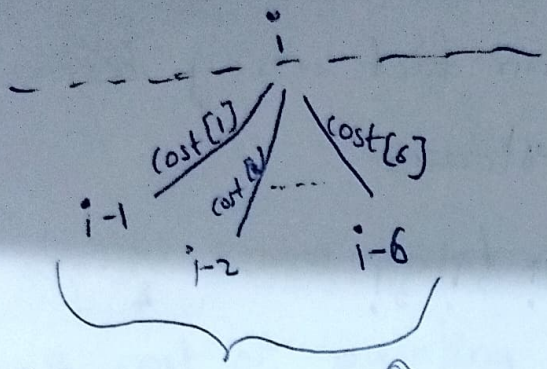

```
int f(n){
    if(n<=6) ret (1<<(n-1));
    ret Σ_{i=1}^{6} f(n-i);
}
```

- $dp[i] = \sum_{j=1}^{6} dp[i-j];$
- Base condition $dp[0] = 1$

$$\forall_{i=1}^{6} dp[i] = \sum_{j=0}^{j<i} dp[j]$$

- $O(N \times 6)$ TC
- $O(N)$ SC
- Space optimisation $\rightarrow O(7)$

⑥ min cost to reach N



cost[i] = min of all these

```
int f(n, dp){
    if(dp[n] != -1) return dp[n];
    dp[n] = min( ∀ (j=1 to 6) f(n-i) + cost[i] )
    return dp[n];
}
```

- dp expression

$$dp[i] = \min\left(\forall_{j=1}^{6} dp[i-j] + cost[j]\right)$$

- Base condition

$$dp[i] = cost[i]$$

$$\forall_{i=2}^{6} dp[i] = \min\left(\forall_{j=1}^{j<i} dp[j] + cost[i-j]\right)$$

TC → O(N) SC → O(N) SO → O(7)

27/02/2025

factorial

Precompute an fact array before reading testcases.

fact[] = $1!$ $2!$ $3!$ - - - $2^{10^5}!$

```
main()
{
  Read N
  print fact[N]
}
```

O(N+T)

x & y set bits

Precompute powers of 2 array

power[] = 2^0 2^1 2^2 - - - 2^{10^5}

```
main()
{
  Read x, y
  if x == y
    print power[x]
  else
    print power[x]+power[y]
}
```

O(N+T)

x 1's y 0's

Red	2	2	16	10	13
Green	6	4	7	3	20
Blue	3	1	9	11	16

There are N houses, paint N houses, with a cost of painting i^{th} building with Red colour → R[i]

① Greedy approach

$$\sum_{i=0}^{N-1} \min(R[i], B[i], G[i])$$

Variation No two adj buildings must have same colour.

(Find costs for starting with (red, green, blue) min of those is answer)

```
int f(R, G, B, idx, prev){
  int cost = int_max;
  if(prev == 0)
    cost = R[i] + min(f(R,G,B, idx+1, 1)
                      , f(R,G,B, idx+1, 2)
```

```
Main(){
    cost = f(R,G,B, 0, -1);
}

int f(R,G,B, idx, prev){
    if(idx == len(R)) ret 0;
    cost = INT_MAX
    if(prev != 0)
        cost = min(cost, f(R,G,B,idx+1,0)+R[i];
    if(prev != 1)
        cost = min(cost, f(R,G,B,idx+1, 1)+G[i];
    if(prev != 2)
        cost = min(cost, f(R,G,B,idx+1,2)+B[i];
    ret cost
}
```

main()

cost = min(f(R, G, B, 0, 0)

f(R, G, B, 0, 1)

f(R, G, B, 0, 2)

// dp approach

```
int f(R,G,B, idx, prev, dpR, dpG, dpB){
    if (idx == N) ret 0;
    cost = INT_MAX;
    if(prev == 0 && dpR[idx] != -1)
        ret dpR[idx];
    if(
        ret
    if(
        ret
    if(prev != 0)
        cost = min (cost, f(R,G,B, idx+1, 0)
                          + R[i]
    if(
        cost =
    if(
        cost =
```

```
if(prev=0)  dpR[idx]=cost;
if(prev=1)
if (prev=2)
ret cost;
}
```

- dp expressions.

$$dpR[i] = \min(dpG[i-1], dpB[i-1]) + R[i]$$

$$dpG[i] = \min(dpR[i-1], dpB[i-1]) + G[i]$$

$$dpB[i] = \min(dpR[i-1], dpG[i-1]) + B[i]$$

- Tabulated code

$$dpR[0], dpG[0], dpB[0] = R[0], G[0], B[0];$$

$$\forall_{i=1}^{N-1} \begin{bmatrix} dpR[i] = \min(dpG[i-1], dpB[i-1]) + R[i] \\ dpG[i] = \min(dpR[i-1], dpB[i-1]) + G[i] \\ dpB[i] = \min(dpR[i-1], dpG[i-1]) + B[i] \end{bmatrix}$$

$$print(\min(dpR[N-1], dpG[N-1], dpB[N-1]))$$

- dp state

$dpR[i]$ = min cost to paint all buildings till i ending with red colour

TC → O(N) SC → O(N) SO → O(6) (70)

(2) Jobs & Machines

You are given N tasks on machine A and B

i^{th} task → 'A' or 'B'

Find min time to complete N tasks

A: 5 5 4 13 17

B: 7 2 6 10 9

variation There exists some switching time also 't'

```
int f(A, B, idx, curr, t){
 if(idx == len(A)) ret 0;
 time = int_max;
 if(curr == 0){
   time = min(time, f(A,B, idx+1, 0, t)+A[idx])
   time = min(time, f(A,B, idx+1, 1, t)+B[idx]+t);
 }
 if(curr == 1){
   time = min(time, f(A,B, idx+1, 1, t)+B[idx]);
   time = min(time, f(A,B, idx+1, 0, t)+A[idx]+t);
 }
 return time;
}
```

dpA, dpB

dpA[i]: min time performing all tasks till i where ith task is performed on machine A

dpA, dpB = {0}xN

dpA[0], dpB[0] = A[0], B[0]

$$\forall_{i=1}^{N} \begin{cases} dpA[i] = \min(dpA[i-1], dpB[i-1]+t) + A[i] \\ dpB[i] = \min(dpB[i-1], dpA[i-1]+t) + B[i] \end{cases}$$

print(min(dpA[N-1], dpB[N-1]))

TC → O(N) SC → O(N) SO → O(4)

⑥ Maximum Subarray Sum. (Kadane's)

A: 4 2 -3 10 6 -10 -10 7 3 2

dp[i]: max subarray sum till ith index

dp[i] = max(dp[i-1], 0) + A[i]

dp[0] = A[0]

$\forall_{i=1}^{N-1}$ dp[i] = max(dp[i-1], 0) + arr[i];

return max(dp)

TC → O(N) SC → O(N) SO → O(1)

```
int f(int arr[], int N){
    int sum = 0
    int ans = int_min;
    for(int ele : arr){
        sum += ele;
        ans = max(ans, sum)
        Sum = max(Sum, 0)
    }
    ret ans;
}
```

(Q) Given an array A, find the max subsequence

A_N: 3 8 100 -6 20 3 6 200 -4 -6 -9 3

Sum of all positives

variation

Max Non-Adjacent Subsequence sum (House robber)

```
int f(arr, idx){
    if(idx <0) ret 0;
    if(idx==0) ret arr[0];
    ret max(f(arr,idx-2)+arr[i],
            f(arr, idx-1), arr[idx]);
}

// memoization
int f(arr, i, dp){
    if (dp[i] != -∞) ret dp[i];
    dp[i] = max(f(arr, i-2, dp)+a[i],
                f(arr,i-1,dp), a[i])
    return dp[i];
}

// table) if(n==-1) return a[0]
dp[N] = [-∞]
dp[0] = a[0]
dp[1] = max(a[0], a[1])
n-1
∀ i=2 [ dp[i] = max(dp[i-2]+a[i]), dp[i-1], a[i])

return dp[N-1];
```

74

TC → O(N) SC → O(N) S.O → O(3)