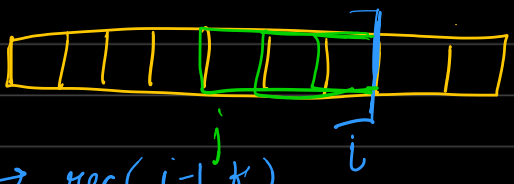
DP

Q Given an array, take K disjoint subarrays such that their sum is maximum.

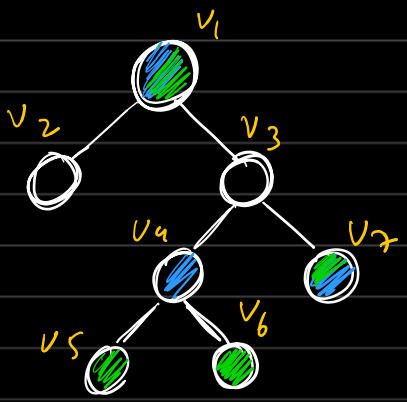


rec(i, k) { leave → rec(i-1, k)
take → rec(i-j, k-1) + p[i] - p[i-j]

instead of using dp[i][k]; use done[i][k] as -1 might be a possible ans

TREE - DP

Q Given a tree, select some nodes ∈ they aren't adjacent & sum of values is maximum



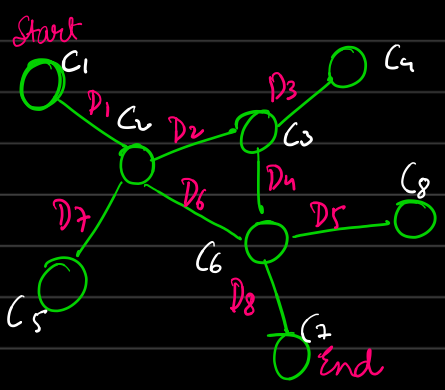
$\max(v_1 + v_4 + v_7, v_1 + v_5 + v_6 + v_7)$

0 → not taking that node
1 → taking that node

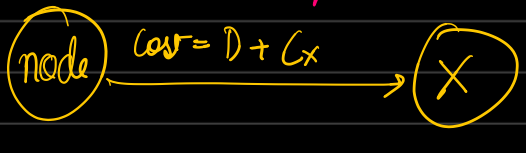
$dp[node][0] = \sum \max(dp[child][0], dp[child][1])$
$dp[node][1] = val[node] + \sum dp[child][0]$

SHORTEST PATHS

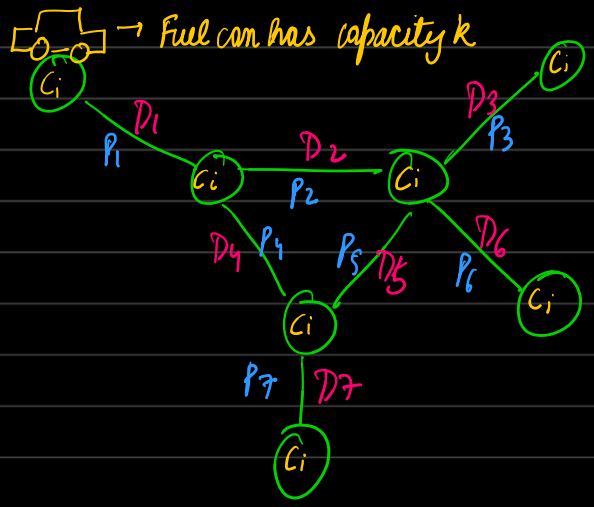
Q



You have to reach end from start.
Each node has a cost.
Each edge has a cost.
Find the shortest path.

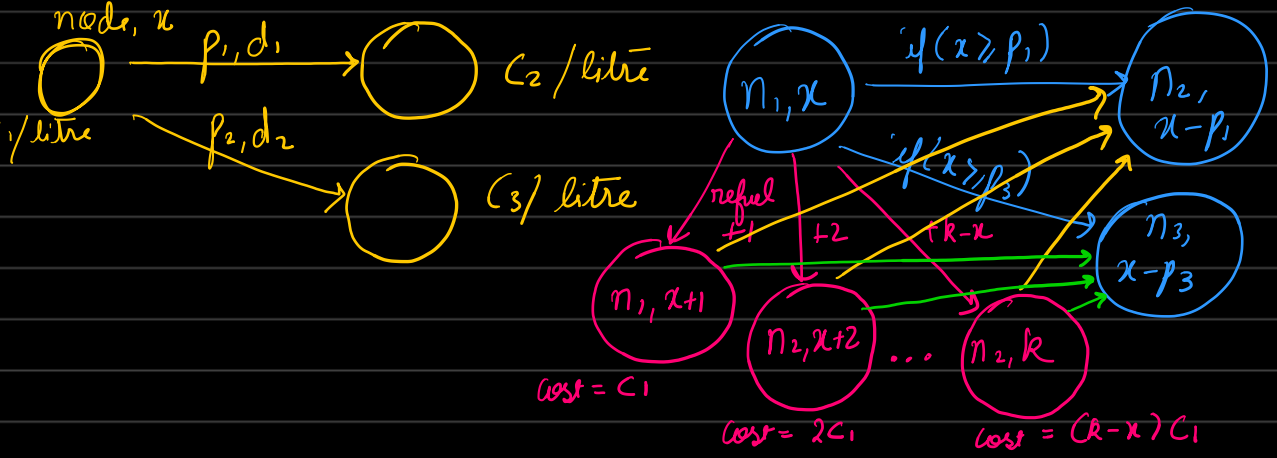


Q/



D → Petrol needed to cover that road
D → distance of that road
Each node has petrol pump with cost C_i / litre
Go from 1 to n.

① Shortest Path → dijkstra using 'D' parameter
② Minimum money is spent → (We can go to a city with lesser cost of petrol and fill it then come back) → parameters important are 'P' & 'C_i'



petrol req.
distance
node info for that edge
current fuel
current node

```
using pii = pair<int, int>
vector<vector<pair<int, pii>>> g;
minheap pq; pq.push({0, {1, 0}}); dist[1][0] = 0;
while(pq.size())
{
    int curr_dist = pq.top().ff;
    pii curr_state = pq.top().ss;        pq.pop();
    if(visited[curr_state.ff][curr_state.ss]) continue;
    visited[curr_state.ff][curr_state.ss] = 1;

    for(auto v: g[curr_state.first])   // try to go to neighbour
    {
        if(curr_state.ss < v.ss.ff) continue;
        if(dist[v.first][curr_state.ss - v.ss.ff] > curr_dist + 0)
        {   dist[v.ff][curr_state.ss - v.ss.ff] = curr_dist + 0;
```

also check not visited

```
        pq.push ({dist[v.ff][curr_state.ss-v.ss.ff], {v.ff, curr_state.ss-v.ss.ff}}); }
    }
    //try to refill
    if(dist[curr_state.ff][curr_state.ss+1] > cur_dist + c[curr_state.ff])
    {
check not   dist[ ][ ] = ____ ;
visited     pq.push({dist[ ][ ], {curr_state.ff, curr_state.ss+1}});
    }
}
```

KRUSKAL ALGORITHM

```
vector<pair<int, pii>> v;
```

cost, edge

```
sort(all(v));
int cost=0;   int edges=0;
for(auto x:v)
{
    int u = x.ss.ff;
    int v = x.ss.ss;
    if(dsu.find(u) != dsu.find(v))
    {
        cost += x.ff;   edges++;
        dsu.unite(u,v);
    }
}
```

if(edges != n-1) ⟶ no MST exists
else cout << cost << endl;

Range Maintenance Ideas

Q Given N intervals [Li, Ri]
Q queries ? Y → no. of open segments at Y

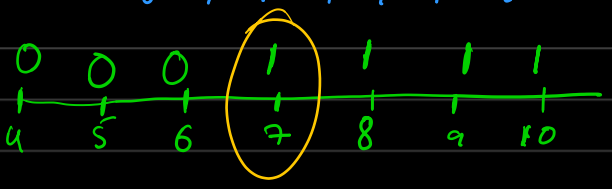
For each Query Y ⇒ Ans is N − (no. of Ri < Y) + (no. of Li > Y)

Q Given 2 arrays A of size N, B of size M and integer K. Create new array C of size N*M consisting of A[i]+B[j] for 1≤i≤N, 1≤j≤M and find Kth smallest element in C.

suppose A = 3 5 K = 3

B = 2 3 4

C = { 5, 6, 7, 7, 8, 9 }



0 → if number of elements <= mid is < k

1 → if number of elements <= mid is >= k

Loop A[i] & get B[j] <= (mid - A[i]) { we can do upper bound }

Q Given an array A of N integers
You can do an operation any no. of times
choose i, j set A[i] = A[i] | A[j]
A[j] = A[i] & A[j].
What can be max. $\sum_i A[i]^2$

A A+B = (A|B) + (A&B) hence sum would remain constant
x^2+y^2 is max for given $x+y=k$ if one is as large as possible

position

2 →	0	0	1	1
1 →	0	1	0	1
0 →	1	1	1	0
	1	3	5	6

free fall →

1	1	0	0
1	1	0	0
1	1	1	0
↓	↓	↓	$\Rightarrow 7^2+7^2+1^2$
7	7	1	$\Rightarrow 99$

Q Given an Array A of N positive integers, find Maximum Bitwise AND of all subsequences of A of size X.

$a_1 \& a_2 \ldots a_x$ = ANS

Setting a higher contributing bit is the key

```
vector<int> a(n);   read(a,n);
int ans=0;
for(i=29; i>=0; i--)
{
    vector<int> temp;
    for(auto x: a)   if(x&(1<<i)) temp.pb(x);
    if(temp.size()>=x)  ans1 = (1<<i), ans = temp;
}
cout<<ans<<endl;
```

Q Given an Array A and N Queries. In each query given [L, R]. Find

- Smallest $X_1 \in (A_L \hat{} X_1) + (A_{L+1} \hat{} X_1) \ldots (A_R \hat{} X_1)$ is max
- Smallest $X_2 \in (A_L | X_2) + (A_{L+1} | X_2) \ldots (A_R | X_2)$ is max
- Smallest $X_3 \in (A_L \& X_3) + (A_{L+1} \& X_3) \ldots (A_R \& X_3)$ is max

Create bit_prefix[n][31] then do bit_prefix[r][i] - bit_prefix[l-1][i] then iterate and solve accordingly

Q

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

Find the farthest 0 from the closest 1.

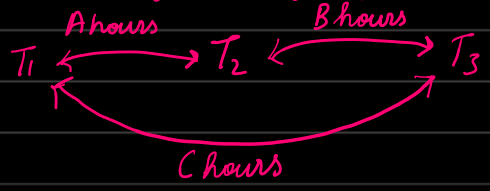
Do multi-source BFS from all 1's.
Answer will be the max distance - 1.

1	2	2	1	1
2	3	2	1	2
1	2	3	2	3
1	2	3	3	3
1	2	2	2	2
1	1	1	1	1

Answer = 3 - 1 = 2

Q

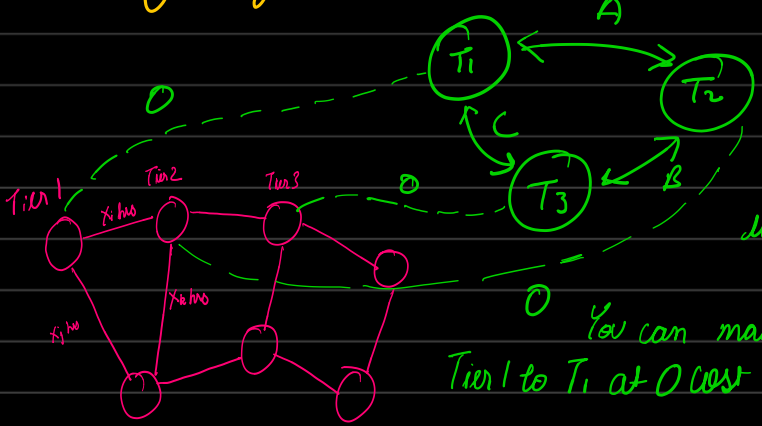
There are 3 types of cities → T1, T2, T3
You are given length of each road.



Find minimum time to a tier 1 city from every city.

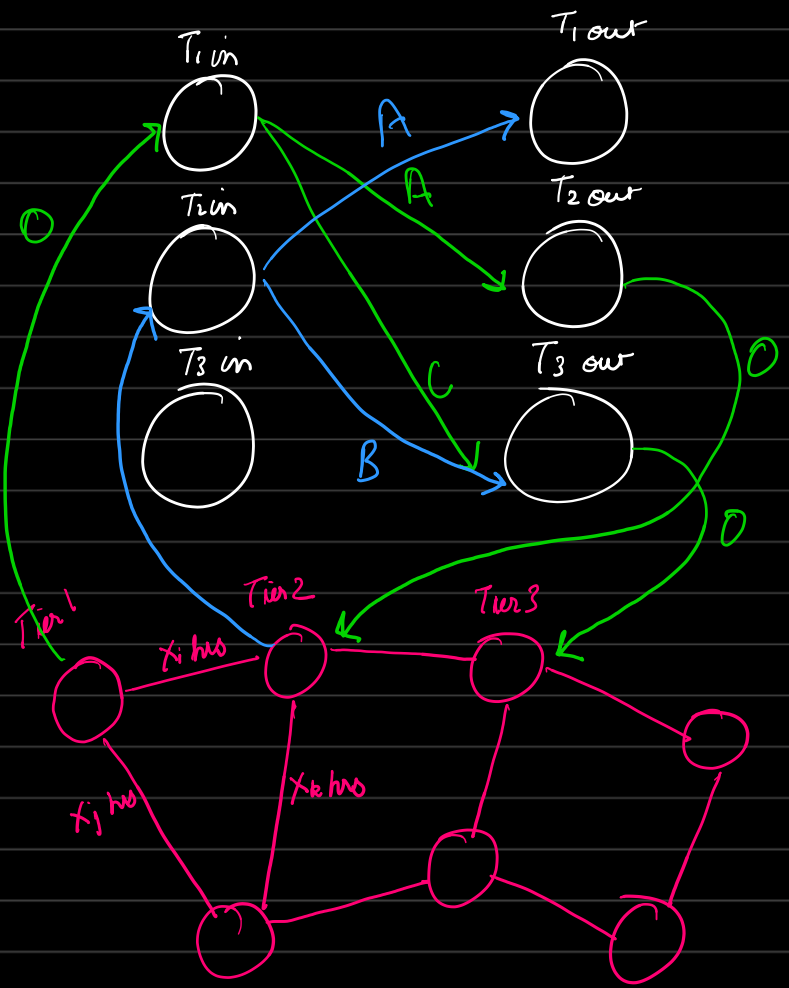
Solution 1 ⇒ Add the extra edges and do multi-source shortest path algorithm from tier-1 cities.
But adding edges will be order N^2 and so it can give TLE.

Solution 2 ⇒



But if Tier 1 → T_i → Tier 1 distance will be 0 which is wrong.

You can make directed edge from Tier 1 to T_i at 0 cost then go Tier 2 or Tier 3



GREEDY

Q $A \to \langle a_1\ a_2 \dots a_n \rangle$ } You can rearrange in such a way
$B \to \langle b_1\ b_2 \dots b_n \rangle$ } that $\sum a_i b_i$ is maximum.

Simply sort both of them and take dot product.

Q There are N problems. In which order to solve the problem to get max score at end.

Problems →	P_1	P_2	P_3	....	P_N
Score →	S_1	S_2	S_3	....	S_N
Decay →	D_1	D_2	D_3	....	D_n (per second)
Time to Solve →	T_1	T_2	T_3	....	T_n

OPTIMAL

$$P_1 \quad P_2 \quad \overset{\text{swap}}{\Longleftrightarrow} \quad P_2 \quad P_1$$

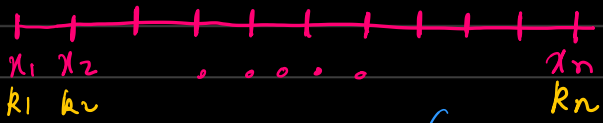
$$\Rightarrow (S_1 - T_1 D_1) + (S_2 - (T_1+T_2) D_2) \geq (S_2 - T_2 D_2) + (S_1 - (T_1+T_2) D_1)$$

$$\Rightarrow -T_1 D_2 \geq -T_2 D_1$$

$$\Rightarrow T_1 D_2 \leq T_2 D_1$$

$$\Rightarrow \frac{D_2}{T_2} \leq \frac{D_1}{T_1} \quad \therefore \text{sort by } \frac{D_i}{T_i} \text{ in decreasing}$$

Q There are N points on a line. Each point has weight. To move x steps its power used will be xk_i. Find a point where everyone reaches with minimum power used.



Basically minimise $\sum_i (k_i |x - x_i|)$

x_i	1	3	7
k_i	1	1	3

$|x-1| + |x-3| + 3|x-7|$
$= |x-1| + |x-3| + |x-7| + |x-7| + |x-7|$

x_i	1	3	5	9
k_i	3	1	2	3
prefix on k_i	3	4	6	9

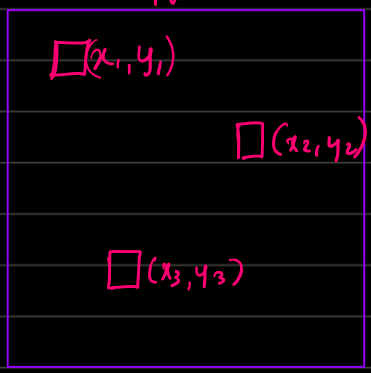
→ find 5th element
↓
do lower_bound for 5
↓
get x_i for that k_i

Q You have 2 operation ① $x \to 2x$
② $x \to x+1$
Initially num is 0. You have to make num = k in min operations. How many operations are required.

example you've to make 12. → 1100

$0 \xrightarrow{+1} 1 \xrightarrow{<<1} 10 \xrightarrow{+1} 11 \xrightarrow{<<1} 110 \xrightarrow{<<1} 1100$

Q



Make these pieces meet at a certain point such that $\sum |x_i - x| + |y_i - y|$ is minimum as possible.

Solution → $F(x,y) = \sum_i |x_i - x| + \sum |y_i - y|$ → median for x, y separately