REMAINIG QUES SHORT EXPLANATION !!

* The integers were very big ; thus long long was used

QUES -2

DIJKSTRA'S ALGORITHM

```
#include <iostream>
#include <vector>
#include <queue>
#include <utility>
#include <climits>
using namespace std;

typedef pair<long long, int> pii;

void dijkstra(int n, int e, vector<long long>& distances, vector<vector<pii>>& adj_list) {
    // Initialize distances to infinity
    distances.assign(n + 1, LLONG_MAX);
    distances[1] = 0;  // Distance to the source node itself is zero

    // Priority queue to process the nodes with the smallest distance first
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    pq.push({0, 1});  // Push the source node with distance 0

    vector<bool> visited(n + 1, false);

    while (!pq.empty()) {
        long long now_dist = pq.top().first;
        int now_node = pq.top().second;
        pq.pop();

        if (visited[now_node]) {
            continue;
        }

        visited[now_node] = true;

        for (const auto& edge : adj_list[now_node]) {
            int next_node = edge.first;
            long long weight = edge.second;
            long long dista = now_dist + weight;

            if (dista < distances[next_node]) {
                distances[next_node] = dista;
                pq.push({dista, next_node});
            }
        }
    }
}

int main() {
    int n, m;
    cin >> n >> m;

    vector<vector<pii>> adj_list(n + 1);
    for (int i = 0; i < m; i++) {
        int a, b;
        long long c;
        cin >> a >> b >> c;
        adj_list[a].emplace_back(b, c);
    }

    vector<long long> distances;
    dijkstra(n, m, distances, adj_list);

    for (int i = 1; i <= n; i++) {
        cout << distances[i] << " ";
    }
    cout << endl;

    return 0;
}
```

This $a \xrightarrow{c} b$ (adjacency list)

array of distances

[illegible]

→ analogy to be used using graphs
→ and bit conversion (to ease out the process)

QUES_3

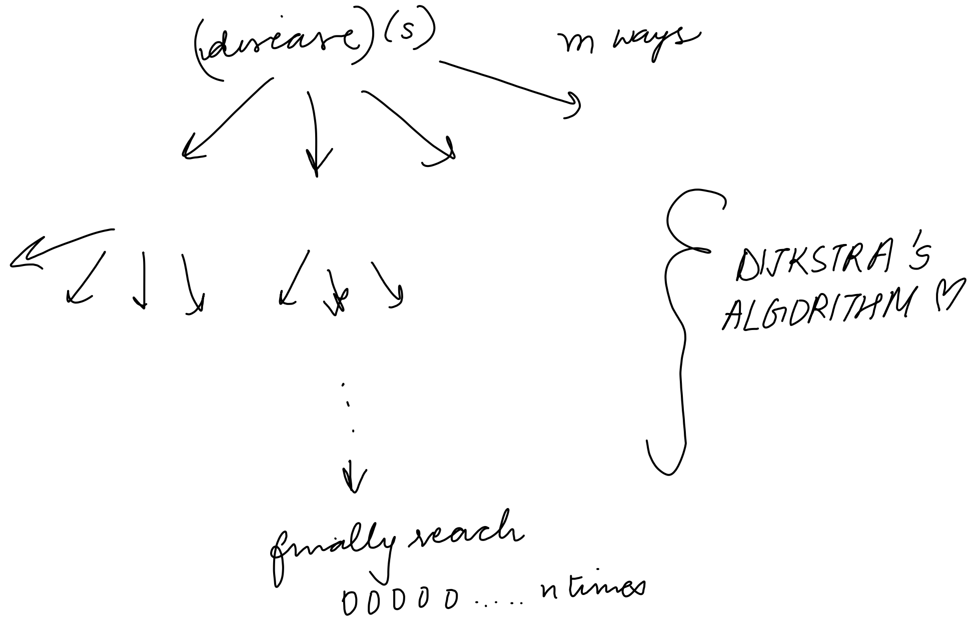
disease → _ _ _ _

cure → _ _ _ _

harm → _ _ _ _

final ans → $(\text{disease} \,\&\, \overline{\text{cure}}) \mid \text{harm}$

```
#include<iostream>
#include<vector>
#include<queue>
using namespace std;

struct edge{
    int d;
    int cure;
    int harm;
};

int change(long long x){
    int ans = 0;
    int i = 0;
    while(x){
        if(x%10) ans|=1<<i;
        i++;
        x/=10;
    }
    return ans;
}

int main(){
    int t;cin>>t;
    while(t--){
        int n,m; cin>>n>>m;
        long long s;cin>>s;
        int initial = change(s);

        struct edge edges[m];
        for (int i = 0; i < m; i++){
            int d;cin>>d;
            long long cure;cin>>cure;
            long long harm;cin>>harm;
            struct edge new_edge;
            new_edge.d = d;
            new_edge.cure = change(cure);
            new_edge.harm = change(harm);
            edges[i] = new_edge;
        }

        vector<int32_t> arr(1<<(n), INT32_MAX);
        arr[initial] = 0;

        priority_queue<pair<int, int>> pq;
        pq.push({-arr[initial], initial});

        while(!pq.empty()){
            pair<int, int> top = pq.top();
            pq.pop();

            for (int i = 0; i < m; i++){
                int a = top.second;
                int b = a & (~edges[i].cure);
                b |= edges[i].harm;


                if(arr[b]>(-top.first + edges[i].d)){
                    arr[b] = (-top.first + edges[i].d);
                    pq.push({-arr[b], b});
                }
            }

        }
        cout<<((arr[0]==INT32_MAX)? -1: arr[0]) << endl;

    }
}
```



PTO

* here the jist is that one can buy and sell happiness also!

* cannot spend the same month's money

* MAX_heap is used

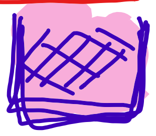
QUES_1

```
#include <iostream>
#include <vector>
#include <queue>
#include <algorithm>
using namespace std;

int main()
{
int t;
cin>>t;

while(t--){
   int n,m;// n- months & m- money each month
   cin>>n >>m;
   vector <int> arr(n);
   for (int i=0;i<n;i++)
      {
      cin>>arr[i];
      }
   priority_queue<int>pq;
   int mon=0;
   int hap=0;
   for(int i=0;i<n;i++)
      {

       if  (arr[i] <= mon)
         {
         mon -= arr[i];
         pq.push(arr[i]);
         hap++;
         }

      else if  (!pq.empty() && pq.top()>arr[i])
         {
         mon+=pq.top();
         pq.pop();
         mon-=arr[i];
         pq.push(arr[i]);
         }
         mon+=m;
      }
cout<<hap<<'\n';

}

return 0;}
```

arr[n] = { _ _ _ _ _ _ }

For example { 6 4
4 10 3 8 6 10 }

HAP BOX



1st N — mon = 4
can't spend

2nd N — mon = 4+4 ✗

3rd N — mon = 8-3
mon = 5+4

3

4th N — mon = 9-8
mon = 1+4

3 8 ; 8 3

5th N — mon = 5 ✗
But we sell max happ
mon = 5+8-6
mon = 7+4

8 3 6 ⇒ 8 6 3

6th N — mon = 11-10
mon = 1+4 = 5

8 6 3 ↑ 10 ⇒ 10 8 6 3

PTO