1. **Maximise collection (toys wala)**

```
#include<bits/stdc++.h>
using namespace std;
#define IO ios_base::sync_with_stdio(false);cin.tie(NULL);cout.tie(NULL);
#define ll                      long long

int main(){
    ll t;
    cin>>t;
    while(t--){
        ll n;
        cin>>n;
        vector<pair<ll ,ll>>A(n);
        for(int i=0;i<n;i++){
            cin>>A[i].first;
            A[i].second = i;
        }
        sort(A.begin() , A.end());
        unordered_map<ll, ll>new_ind;
        for(int i=0;i<n;){
            ll j=i;
            while(j<n && A[j].first==A[i].first) j++;
            for(int k=i;k<j;k++){
                new_ind[A[k].second] = i;
            }
            i = j;
        }

        vector<ll>pref(n, 0);
        for(int i=0;i<n;i++){
            pref[i] = (i?pref[i-1]:0) + A[i].first;
        }

        ll q;
        cin>>q;
        while(q--){
            ll c, k;
            cin>>c>>k;
            vector<ll>temp(k);
            for(int i=0;i<k;i++){
                cin>>temp[i];
                temp[i]--;
                temp[i] = new_ind[temp[i]];
```

```
            }
            sort(temp.begin() , temp.end());

            ll l=0;
            ll r = lower_bound(pref.begin() , pref.end() , c+1) - pref.begin()-1;
            ll cnt =0;
            ll want = c;
            for(int i=0;i<k;i++){
                if(temp[i]<=r){
                    cnt++;
                    want += A[temp[i]].first;
                    r = lower_bound(pref.begin() , pref.end() , want+1) - pref.begin()-1;
                }
                else break;
            }
            cout<<(r+1)-cnt<<endl;
        }
    }
    return 0;
}
```

2. Prime path (grid)

```
#include<bits/stdc++.h>
using namespace std;
#define int long long
#define v(int) vector<int>
int const maxv=1e6+1;
int const inf=1e18;
int factor[maxv];
void countf(){
    factor[1]=1;
    for(int i=0;i<maxv;i++){
        if(factor[i]!=0)continue;
        for(int j=i;j<maxv;j+=i)factor[j]++;
    }
}
void solve(){
    int n;
    int a[n+1][n+1];
    for(int i=1;i<=n;i++)for(int j=1;j<=n;j++)cin>>a[i][j];
    priority_queue<v(int),v(v(int)),greater<v(int)>>pq;
    v(v(int))cost(n+1,v(int)(n+1,inf));
    pq.push({0,1,1});
    cost[1][1]=0;
    while(pq.size()){
        auto tp=pq.top();
        pq.pop();
        int x0=tp[1],y0=tp[2];
        if(cost[x0][y0]<tp[0])continue;
        int val=a[x0][y0];
        int p=factor[val];
        for(int i=0;i<=p;i++)
        for(int j=0;j+i<=p;j++){
            int x=tp[1]-i,y=tp[2]-j;
            if(x>=1&&x<=n&&y>=1&&y<=n&&cost[x][y]>tp[0]+sqrt(val)){
                cost[x][y]=tp[0]+sqrt(val);
                pq.push({cost[x][y],x,y});
            }
             x=tp[1]-i,y=tp[2]+j;
            if(x>=1&&x<=n&&y>=1&&y<=n&&cost[x][y]>tp[0]+sqrt(val)){
                cost[x][y]=tp[0]+sqrt(val);
                pq.push({cost[x][y],x,y});
            }
             x=tp[1]+i,y=tp[2]-j;
```

```
            if(x>=1&&x<=n&&y>=1&&y<=n&&cost[x][y]>tp[0]+sqrt(val)){
                cost[x][y]=tp[0]+sqrt(val);
                pq.push({cost[x][y],x,y});
            }
             x=tp[1]+i,y=tp[2]+j;
            if(x>=1&&x<=n&&y>=1&&y<=n&&cost[x][y]>tp[0]+sqrt(val)){
                cost[x][y]=tp[0]+sqrt(val);
                pq.push({cost[x][y],x,y});
            }


        }

    }

}
int32_t main(){

countf();
int t;
cin>>t;
while(t--)solve();
}
```

3. Median (tree wala)

```
#include<bits/stdc++.h>
using namespace std;
#define ll long long int
#define fr(i,n) for(ll i=0;i<(n);++i)
#define fr1(i,n) for(ll i=1;i<=(n);++i)
#define rep(i,a,b) for(ll i=a;i<=b;++i)
#define per(i,a,b) for(ll i=a;i>=b;i--)
#define TxtIO   freopen("input.txt","r",stdin); freopen("output.txt","w",stdout);
#define MP make_pair
#define mod 1000000007

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

// namespace necessary for GNU based
// policy based data structures
using namespace __gnu_pbds;


// Declaring ordered_set for pair<int,int>
typedef tree<pair<int,int>, null_type,
        less<pair<int,int>>, rb_tree_tag,
        tree_order_statistics_node_update>
        ordered_set_pair;

ll n;
vector<int>arr;
ll ans;
vector<vector<int>>g;
vector<int>num_child;
void dfs_on_tree(int node,int parent,int depth,ordered_set_pair&exploring){
    num_child[node]=0;
    exploring.insert(MP(arr[node],node));
    for(auto v:g[node]){
        if(v!=parent){
            num_child[node]++;
            dfs_on_tree(v,node,depth+1,exploring);

        }
    }
    if(num_child[node]==0) {
```

```
            ll sz=(ll)exploring.size();
            if(sz%2){
            auto it=exploring.find_by_order(sz/2);
            ans+=((*it).first);
            }
    }
    if(exploring.find(MP(arr[node],node))!=exploring.end()){
    exploring.erase(MP(arr[node],node));
    }
}
void solve(){
cin>>n;
g.resize(n+1);
arr.resize(n+1);
ans=0;
num_child.assign(n+1,0);
fr1(i,n){
    cin>>arr[i];
}
fr(i,n-1){
    ll a,b;
    cin>>a>>b;
    g[a].push_back(b);
    g[b].push_back(a);
}
ordered_set_pair ex;
dfs_on_tree(1,0,0,ex);
ans+=arr[1];
cout<<ans<<"\n";
g.clear();
arr.clear();
num_child.clear();
}

signed main(){
ios_base::sync_with_stdio(0);
cin.tie(0);cout.tie(0);

int _t;cin>>_t;while(_t--)
solve();
}
```

4. **Hybrid wala**

```
#include <bits/stdc++.h>
using namespace std;
void solve()
{
    int n, m;
    cin >> n;
    vector<int> a(n);
    for (int &x : a)
        cin >> x;
    cin >> m;
    vector<int> b(m);
    for (int &x : b)
        cin >> x;

    vector<pair<int, int>> mna(n + 1, {0, 1e9});
    vector<pair<int, int>> mnb(m + 1, {0, 1e9});
    for (int i = 1; i <= n; i++)
    {
        mna[i].first = max(mna[i - 1].first, a[i - 1]);
        mna[i].second = min(mna[i - 1].second, a[i - 1]);
    }
    for (int i = 1; i <= m; i++)
    {
        mnb[i].first = max(mnb[i - 1].first, b[i - 1]);
        mnb[i].second = min(mnb[i - 1].second, b[i - 1]);
    }
    vector<vector<int>> dp(n + 1, vector<int>(m + 1));
    for (int i = 0; i <= n; i++)
    {
        for (int j = 0; j <= m; j++)
        {
            if (i != n)
                dp[i + 1][j] = max(dp[i + 1][j], dp[i][j] + max(mna[i + 1].first, mnb[j].first) - min(mna[i +
1].second, mnb[j].second));
            if (j != m)
                dp[i][j + 1] = max(dp[i][j + 1], dp[i][j] + max(mna[i].first, mnb[j + 1].first) -
min(mna[i].second, mnb[j + 1].second));
        }
    }
    cout << dp[n][m];
}
```

```
int main()
{
    int t;
    cin >> t;
    while (t--)
    {
        solve();
    }
}
```

5. Special subsequences

```
long long power(long long x, int y, int p)
{
    long long res = 1;

    x = x % p;
    while (y > 0) {
        if (y & 1)
            res = (res * x) % p;
        y = y >> 1;
        x = (x * x) % p;
    }
    return res;
}
long long modInverse(long long n, int p)
{
    return power(n, p - 2, p);
}
long long mul(long long x,
                    long long y, int p)
{
    return x * 1ull * y % p;
}
long long divide(long long x,
                      long long y, int p)
{
    return mul(x, modInverse(y, p), p);
}
long long nCrModPFermat(long long n,
                              int r, int p)
{
    if (n < r)
        return 0;
    if (r == 0)
        return 1;
    if (n - r < r)
        return nCrModPFermat(n, n - r, p);
    long long res = 1;
    for (int i = r; i >= 1; i--)
        res = divide(mul(res, n - i + 1, p), i, p);
    return res;
}
```

```
long long solve(int n, int k, string s) {
    vector<int> freq(26,0);
    for(int i=0;i<n;i++) {
        freq[s[i]-'a']++;
    }
    sort(freq.begin(),freq.end(),greater<int>());
    int cnt=0,cnt1=0;
    for(int i=0;i<26;i++) {
        if(freq[i]==freq[k-1]) {
            cnt++;
            if(i<k) cnt1++;
        }
    }
    long long M = 1e9+7;
    long long ans = nCrModPFermat(cnt,cnt1,M);
    for(int i=0;i<k;i++) {
        ans=(ans*freq[i])%M;
    }
    return ans;
}
```