Contents

1 DP

1.1 LCS

```cpp
#include <bits/stdc++.h>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0)
using namespace std;
string s1, s2;
int dp[505][505];
int main(){
    IOS
    cin >> s1 >> s2;
    memset(dp, 0, sizeof(dp));
    int l1 = s1.size(), l2 = s2.size();
    for(int i = 1;i <= l1;i++){
        for(int j = 1;j <= l2;j++){
            if(s1[i - 1] == s2[j - 1]) dp[i][j] =
                dp[i - 1][j - 1] + 1;
            else dp[i][j] = max(dp[i - 1][j], dp[i][j
                - 1]);
        }
    }
    cout << dp[l1][l2] << '\n';

    return 0;
}
```

1.2 LIS $O(n^2)$

```cpp
#include <bits/stdc++.h>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
using namespace std;
typedef long long ll;
int main(){
    IOS
    int arr[100];
    int n;
    cin >> n;
    for(int i = 0;i < n;i++) cin >> arr[i];
    int dp[100];
    for(int i = 0;i < n;i++) dp[i] = 1;
    for(int i = 0;i < n;i++){
        for(int j = 0;j < i;j++){
            if(arr[i] > arr[j])
                dp[i] = max(dp[j] + 1, dp[i]);
        }
    }
    int ans = 1;
    for(int i = 0;i < n;i++) ans = max(ans, dp[i]);
    cout << ans << '\n';

    return 0;
}
```

1.3 LIS $O(n \log n)$

```cpp
class Solution {
public:
    int lengthOfLIS(vector<int>& nums) {
        vector<int> v;
        int n = nums.size();
        for(int i = 0;i < n;i++){
            int p = lower_bound(v.begin(), v.end(),
                nums[i]) - v.begin();
            if(p == v.size()) v.push_back(nums[i]);
            else v[p] = nums[i];
        }
        return v.size();
    }
};
```

1.4 LIS $O(n \log n)$

```cpp
for(int i=0;i<num.size();i++){
    if(lis.empty()||lis.back()<num[i]){
        lis.push_back(num[i]);
        dp[i]=lis.size();
    }
    else{
        auto iter=lower_bound(all(lis),num[i]);
        dp[i]=iter-lis.begin()+1;
        *iter=num[i];
    }
}
int length=lis.size();
for(int i=num.size()-1;i>=0;i--){
    if(dp[i]==length){
        ans.push_back(num[i]);
        length--;
    }
}
```

2 Prime

2.1 質數篩 CPP

```cpp
bitset<MAXN> prime_bool;
vector<ll> prime;
void find_prime(){
    prime_bool.set();
    for(int i=2;i<MAXN;i++){
        if(prime_bool[i]){
            prime.push_back(i);
        }
        for(auto j:prime){
            if(j*i>=MAXN)
                break;
            prime_bool[j*i]=0;
            if(i%j==0)
                break;
        }
    }
}
```

2.2 質數篩 PY

```python
is_prime = n * [1]
is_prime[0] = is_prime[1] = 0

for i in range(2, n):
    if is_prime[i]:
        for j in range(2, n):
            if i * j >= n:
                break
            is_prime[i * j] = 0
```

2.3 單一質數

```cpp
bool prime(int n){
    if(n < 2) return false;
    if(n <= 3) return true;
    if(!(n % 2) || !(n % 3)) return false;
    for(int i = 5;i * i <= n;i += 6)
        if(!(n % i) || !(n % (i + 2))) return false;
    return true;
}
```

2.4 egcd CPP

```cpp
int exgcd(int a,int b,int &x,int &y){
    if(b==0){
        x=1,y=0;
        return a;
    }
    int gcd=exgcd(b,a%b,y,x);
    y-=a/b*x;
    return gcd;
}
```

2.5 egcd PY

```python
def egcd(a: int, b: int) -> Tuple[int, int, int]:
    """return (g, x, y) such that a*x + b*y = g = gcd(a, b)"""
    """x % y"""
    if a == 0:
        return (b, 0, 1)
    else:
        b_div_a, b_mod_a = divmod(b, a)
        g, x, y = egcd(b_mod_a, a)
        return (g, y - b_div_a * x, x)
```

3 Graph

3.1 Floyd Warshall

```cpp
int n, rd, l, r, v;
cin >> n >> rd;
vector<vector<int>> dp(n + 1, vector<int>(n + 1, 1e9));
for(int i = 0; i < rd; i++){
    cin >> l >> r >> v;
    dp[l][r] = dp[r][l] = v;
    //每條路皆雙向
}

//以下即 Floyd-Warshall
for(int k = 1; i <= n; i++){
    for(int i = 1; j <= n; j++){
        for(int j = 1; k <= n; k++){
            dp[i][j] = min(dp[i][k] + dp[k][j] , dp[i][j]);
            //窮舉所有鬆弛可能
        }
    }
}
cin >> l >> r;
cout << dp[l][r];
```

3.2 Bellman Ford

```cpp
#include <bits/stdc++.h>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define INF 0x3f3f3f3f
using namespace std;
typedef long long ll;
struct Edge{
  int x, y, t;
};
int dis[1005];
int main(){
  IOS
  int c;
  cin >> c;
  while(c--){
    vector<Edge> edge;
    int n, m;
    cin >> n >> m;
    for(int i = 0;i <= n;i++) dis[i] = INF;
    dis[0] = 0;
    for(int i = 0;i < m;i++){
      int x, y, t;
      cin >> x >> y >> t;
      edge.push_back({x, y, t});
    }
    for(int i = 0;i < n - 1;i++){
      for(int j = 0;j < m;j++){
        if(dis[edge[j].x] + edge[j].t < dis[edge[j].y]){
          dis[edge[j].y] = dis[edge[j].x] + edge[j].t;
        }
      }
    }
    bool judge = true;
    for(auto e : edge){
      if(dis[e.x] + e.t < dis[e.y]){
        judge = false;
        break;
      }
    }
    cout << (judge ? "not possible" : "possible") << '\n';
  }

  return 0;
```

```
}
```

3.3 SPFA

```
#define mem(x) memset(x, 0, sizeof(x))
struct road{
  int r, val;
};
int main(){
  int n, e, l, r, v;
  cin >> n >> e;
  vector<int> dp(n + 1, 1e9);
  vector<road> rd[n + 1];
  for(int i = 0; i < e; i++){
    cin >> l >> r >> v;
    rd[l].push_back({r, v});
    rd[r].push_back({l, v});
  }
  cin >> l >> r;
  dp[l] = 0;
  queue<int> que;
  que.push(l);
  bool check[n + 1]; mem(check);
  int cnt[n + 1]; mem(cnt);
  while(!que.empty()){
    int tmp = que.front(); que.pop();
    check[tmp] = 0, cnt[tmp]++;
    if(cnt[tmp] >= n) {cout << "neg cycle\n"; break;}
    for(auto & i : rd[tmp]){
      if(dp[i.r] > dp[tmp] + i.val){
        dp[i.r] = dp[tmp] + i.val;
        if(!check[i.r]) check[i.r] = 1, que.push(i.r);
      }
    }
  }
  for(auto & i : dp) cout << i << ' ';
  return 0;
}
```

3.4 Dijkstra

```
#include <iostream>
#include <algorithm>
#include <vector>
#include <queue>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define INF 2147483647
using namespace std;
int n, m;
vector<pair<int, int>> adj[100005];
bool visited[100005] = {false};
priority_queue<pair<int, int>> pq;
int dis[100005], parent[100005];
void solve(){ // Dijkstra
    dis[0] = 0;
    for(int i = 1;i < n;i++) dis[i] = INF;
    pq.push(make_pair(0, 0));
    while(!pq.empty()){
        auto node = pq.top();
        pq.pop();
        int v = node.second; // parent
        if(visited[v]) continue;
        visited[v] = true;
        for(auto i : adj[v]){
            int vertex = i.first, weight = i.second;
            if(visited[vertex]) continue;
            if(dis[v] + weight < dis[vertex]){
                dis[vertex] = dis[v] + weight;
                parent[vertex] = v;
                pq.push(make_pair(-dis[vertex],
                    vertex));
            }
        }
    }
    int maxd = -1, cnt = 0;
    for(int i = 0;i < n;i++){
        if(dis[i] < INF){
            if(dis[i] > maxd) maxd = dis[i];
        }
        else cnt++;
    }
    cout << maxd << '\n' << cnt << '\n';
}
int main(){
    IOS
    cin >> n >> m;
    for(int i = 0;i < m;i++){
        int u, v, w;
        cin >> u >> v >> w;
        adj[u].push_back(make_pair(v, w));
        adj[v].push_back(make_pair(u, w));
    }
    solve();

    return 0;
}
```

3.5 SecondSP

```
#include <bits/stdc++.h>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define INF 0x3f3f3f3f
using namespace std;

typedef pair<int, int> pii;
typedef long long ll;

const int MAXN = 1005;

vector<pii> adj[MAXN]; // adj[u] stores pairs {v,
    weight}
int dist[MAXN], sec_dist[MAXN]; // shortest and
    second shortest distances

void dijkstra(int s, int n){
    // Min-heap to store {distance, node}
    priority_queue<pii, vector<pii>, greater<pii>> pq;
    fill(dist, dist + n + 1, INF);
    fill(sec_dist, sec_dist + n + 1, INF);

    dist[s] = 0;
    pq.push({0, s});

    while(!pq.empty()){
        int d = pq.top().first;
        int u = pq.top().second;
        pq.pop();

        // If we found a path longer than the second
            shortest, skip it
        if (sec_dist[u] < d) continue;

        for (auto &edge : adj[u]){
            int v = edge.first;
            int w = edge.second;

            int new_dist = d + w;

            // If this gives a new shortest path to v
            if(new_dist < dist[v]){
                sec_dist[v] = dist[v];
                dist[v] = new_dist;
                pq.push({dist[v], v});
                pq.push({sec_dist[v], v});
            }
            // If this gives a new second shortest
                path to v
```

```
            else if(new_dist > dist[v] && new_dist < sec_dist[v]){
                sec_dist[v] = new_dist;
                pq.push({sec_dist[v], v});
            }
        }
    }
}

int main() {
    IOS
    int t;
    cin >> t;
    while(t--){
        int n, m, s, d;
        cin >> n >> m >> s >> d;

        // Reset adjacency list
        for(int i = 1; i <= n; i++) adj[i].clear();

        for(int i = 0; i < m; i++){
            int u, v, w;
            cin >> u >> v >> w;
            adj[u].push_back({v, w});
            adj[v].push_back({u, w}); // If the graph is undirected
        }

        dijkstra(s, n);

        if(sec_dist[d] == INF) cout << -1 << '\n'; // No second shortest path
        else cout << sec_dist[d] << '\n';
    }

    return 0;
}
```

3.6 Kurskal's Algorithm

```
#include <bits/stdc++.h>
#define IOS ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
using namespace std;
int parent[10005];
struct Edge{
    int u, v, w;
    bool operator < (Edge &b){
        return w < b.w;
    }
};
int query(int a){
    if(parent[a] == -1) return a;
    return parent[a] = query(parent[a]);
}
bool merge(int a, int b){
    int r1 = query(a);
    int r2 = query(b);
    if(r1 == r2) return false;
    if(parent[r1] < parent[r2]) parent[r2] = r1;
    else parent[r1] = r2;
    return true;
}
int main(){
    IOS
    int n, m;
    memset(parent, -1, sizeof(parent));
    cin >> n >> m;
    vector<Edge> adj;
    for(int i = 0;i < m;i++){
        int u, v, w;
        cin >> u >> v >> w;
        adj.push_back({u, v, w});
    }
    sort(adj.begin(), adj.end());
    // for(int i = 0;i < m;i++) cout << adj[i].w << ' ';
    int cost = 0, n_edge = 0;
    for(Edge e : adj){
        if(merge(e.u, e.v)){
            cost += e.w;
            n_edge++;
        }
    }
    if(n_edge == n - 1) cout << cost << '\n';
    else cout << -1 << '\n';

    return 0;
}
```

3.7 SecondMST

```
#include <bits/stdc++.h>
#define IOS ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define INF 0x3f3f3f3f
using namespace std;
typedef long long ll;
int anc[105], sz[105];
struct Edge{
    int a, b, c;
};
int Find(int a){
    return (anc[a] == a ? a : anc[a] = Find(anc[a]));
}
bool merge(int a, int b){
    a = Find(a);
    b = Find(b);
    if(a == b) return false;
    if(sz[a] < sz[b]) swap(a, b);
    anc[b] = a;
    sz[a] += sz[b];
    return true;
}
int main(){
    IOS
    int t;
    cin >> t;
    while(t--){
        int n, m;
        vector<Edge> edge;
        cin >> n >> m;
        vector<pair<int, int> > vec[105];
        for(int i = 0;i < m;i++){
            int a, b, c;
            cin >> a >> b >> c;
            edge.push_back({a, b, c});
        }
        for(int i = 1;i <= n;i++){
            sz[i] = 0;
            anc[i] = i;
        }
        sort(edge.begin(), edge.end(), [&](Edge &u, Edge &v){return u.c < v.c;});
        int cost1 = 0, cnt = 0;
        vector<int> mst;
        for(int i = 0;i < edge.size();i++){
            if(merge(edge[i].a, edge[i].b)){
                cost1 += edge[i].c;
                mst.push_back(i);
                if(++cnt == n - 1) break;
            }
        }
        int cost2 = INF;
        for(int i = 0;i < mst.size();i++){
            cnt = 0;
            int res = 0;
            for(int i = 1;i <= n;i++) anc[i] = i;
            for(int j = 0;j < edge.size();j++){
                if(mst[i] == j) continue;
                if(merge(edge[j].a, edge[j].b)){
```

```
                    res += edge[j].c;
                    if(++cnt == n - 1){
                        cost2 = min(cost2, res);
                        break;
                    }
                }
            }
        }

        cout << cost1 << ' ' << cost2 << '\n';
    }

    return 0;
}
```

3.8 Prim's Algorithm

```
#include <iostream>
#include <queue>
#include <algorithm>
#include <cstring>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
using namespace std;
int n, m, dis[10005], parent[10005];
bool visited[10005] = {false};
vector<pair<int, int> > adj[100005];
int main(){
    IOS
    // freopen("input.in", "r", stdin);
    cin >> n >> m;
    memset(dis, 0x3f3f3f3f, sizeof(dis));
    memset(parent, -1, sizeof(parent));
    for(int i = 0;i < m;i++){
        int u, v, w;
        cin >> u >> v >> w;
        adj[u].push_back({v, w});
        adj[v].push_back({u, w});
    }
    int start = 0;
    dis[start] = 0;
    priority_queue<pair<int, int>, vector<pair<int,
        int> >, greater<pair<int, int> > > pq;
    pq.push({dis[start], start});
    while(!pq.empty()){
        pair<int, int> cur = pq.top();
        pq.pop();
        if(visited[cur.second]) continue;
        visited[cur.second] = true;
        for(auto i : adj[cur.second]){
            if(visited[i.first]) continue;
            if(dis[i.first] > i.second){
                dis[i.first] = i.second;
                parent[i.first] = cur.second;
                pq.push({dis[i.first], i.first});
            }
        }
    }
    int cost = 0, err = 0;
    for(int i = 0;i < n;i++){
        if(dis[i] < 0x3f3f3f3f) cost += dis[i];
        else err++;
    }
    cout << (err ?  -1 : cost) << "\n";
    // for(int i = 0;i < n;i++) cout << dis[i] << ' ';

    return 0;
}
```

3.9 LCA

```
#include <bits/stdc++.h>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define INF 0x3f3f3f3f
using namespace std;
typedef long long ll;
const int N = 2e5 + 5;
int n, q;
vector<int> vec[N];
int p[20][N], in[N], out[N];
bool valid(int a, int b){
    return (in[a] <= in[b] && out[b] <= out[a]);
}
void dfs(int cur, int par){
    static int t = 0;
    p[0][cur] = par;
    in[cur] = t++;
    for(auto e : vec[cur]){
        dfs(e, cur);
    }
    out[cur] = t++;
}
int lca(int a, int b){
    if(valid(a, b)) return a;
    for(int i = 19;i >= 0;i--){
        if(!valid(p[i][a], b)) a = p[i][a];
    }
    return p[0][a];
}
int main(){
    IOS
    cin >> n >> q;
    for(int i = 2;i <= n;i++){
        int e;
        cin >> e;
        vec[e].push_back(i);
    }
    dfs(1, 1);
    for(int i = 1;i < 20;i++){
        for(int j = 1;j <= n;j++){
            p[i][j] = p[i - 1][p[i - 1][j]];
        }
    }
    while(q--){
        int u, v;
        cin >> u >> v;
        cout << lca(u, v) << '\n';
    }

    return 0;
}
```

3.10 Topological Sort

```
#include <bits/stdc++.h>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
using namespace std;
typedef long long ll;
vector<int> vec[200005];
int ind[100005];
int main(){
    IOS
    int n, m;
    cin >> n >> m;
    memset(ind, 0, sizeof(ind));
    for(int i = 0;i < m;i++){
        int a, b;
        cin >> a >> b;
        ind[b]++;
        vec[a].push_back(b);
    }
    queue<int> q;
    for(int i = 1;i <= n;i++){
        if(ind[i] == 0) q.push(i);
    }
    vector<int> top;
    while(!q.empty()){
        int cur = q.front();
```

```
        q.pop();
        top.push_back(cur);
        for(auto e : vec[cur]){
            ind[e]--;
            if(ind[e] == 0){
                q.push(e);
            }
        }
    }
    if(top.size() == n){
        for(auto i : top) cout << i << ' ';
        cout << '\n';
    }
    else cout << "IMPOSSIBLE" << '\n';

    return 0;
}
```

4 RMQ

4.1 Segment Tree V1

```
#include <bits/stdc++.h>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0)
#define L(x) (x << 1)
#define R(x) ((x << 1) | 1)
using namespace std;
typedef long long ll;
ll seg[500005 << 2], lazy[500005 << 2];
int n, q;
void init(){
    memset(seg, 0, sizeof(seg));
    memset(lazy, 0, sizeof(lazy));
}
void build(int x, int l, int r){
    if(l == r){
        cin >> seg[x];
        return;
    }
    int mid = (l + r) >> 1;
    build(L(x), l, mid);
    build(R(x), mid + 1, r);
    seg[x] = seg[L(x)] + seg[R(x)];
}
void push(int pos, int size){
    lazy[L(pos)] += lazy[pos];
    lazy[R(pos)] += lazy[pos];
    seg[pos] = seg[pos] + lazy[pos] * size;
    lazy[pos] = 0;
}
void modify(int x, int l, int r, int ql, int qr, int
    val){
    if(lazy[x]) push(x, (r - l) + 1);
    // seg[x] = seg[L(x)] + (mid - l) * lazy[L(x)] +
        seg[R(x)] + (r - mid) * lazy[R(x)];
    seg[x] += val * (qr - ql + 1);
    if(ql <= l && qr >= r){
        lazy[x] += val;
        return;
    }
    int mid = (l + r) >> 1;
    if(qr <= mid) modify(L(x), l, mid, ql, qr, val);
    else if(ql > mid) modify(R(x), mid + 1, r, ql,
        qr, val);
    else{
        modify(L(x), l, mid, ql, mid, val);
        modify(R(x), mid + 1, r, mid + 1, qr, val);
    }
}
ll query(int x, int l, int r, int ql, int qr){
    if(ql <= l && qr >= r) return seg[x] + lazy[x] *
        (r - l);
    if(lazy[x]) push(x, (r - l) + 1);
    int mid = (l + r) >> 1;
    if(qr <= mid) return query(L(x), l, mid, ql, qr);
    else if(ql > mid) return query(R(x), mid + 1, r,
        ql, qr);
    else return query(L(x), l, mid, ql, mid) +
        query(R(x), mid + 1, r, mid + 1, qr);
}
int main(){
    IOS
    init();
    cin >> n;
    build(1, 1, n);
    cin >> q;
    while(q--){
        int v, x, y, k;
        cin >> v;
        if(v == 1){
            cin >> x >> y >> k;
            modify(1, 1, n, x, y, k);
        }
        else{
            cin >> x >> y;
            ll ans = query(1, 1, n, x, y);
            cout << ans << '\n';
        }
    }

    return 0;
}
```

4.2 Segment Tree V2

```
struct Node{
    int left; // 左邊邊界
    int right; //右邊邊界
    int value; //儲存的值
    int z; //區間修改用，如果沒有區間修改就不需要
}node[4 * N ];

#define Lson(x) (x << 1) //左子樹
#define Rson(x) ((x << 1) +1) //右子樹

void question(){
    for(int i = 1; i <= 10; i++) num[i] = i * 123 % 5;
    // num 為題目產生的一段數列
    // hash 函數，讓 num 的 i 被隨機打亂
}

void build(int left , int right , int x = 1 ){
    // left 為題目最大左邊界，right
        為題目最大右邊界，圖片最上面的 root
        為第一個節點
    node[x].left = left ; //給 x 節點左右邊界
    node[x].right = right ;
    if(left == right){
        //如果左右邊界節點相同，表示這裡是葉節點
        node[x].value = num[left] ;  //把 num 值給
            node[x]
        //這裡的 num 值表示，我們要在 value 要放的值
        return ; //向前返回
    }
    int mid = (left + right ) / 2 ; //切半，產生二元樹

    //debug
    //cout << mid << '\n' ;
    //cout << x << ' ' << node[x].left << ' ' <<
        node[x].right << ' ' << '\n' ;

    build(left , mid , Lson(x)) ; //將區間改為 [left,
        mid] 然後帶給左子樹
    build(mid + 1 , right , Rson(x)) ; //將區間改為
        [mid+1, right] 然後帶給右子樹
```

```
    node[x].value = min(node[Lson(x)].value  ,
        node[Rson(x)].value ) ;
    //查詢左右子樹哪個數值最小，並讓左右子樹最小值表示此區
}

void modify(int position , int value , int x = 1 ){
    //修改數字
    if(node[x].left == position && node[x].right ==
        position ){ //找到葉節點
       node[x].value = value ; //修改
       return ; //傳回
    }
    int mid = (node[x].left + node[x].right ) / 2 ;
        //切半，向下修改
    if(position <= mid )
        //如果要修改的點在左邊，就往左下角追蹤
        modify(position , value , Lson(x) );
    if(mid < position )
        //如果要修改的點在右邊，就往右下角追蹤
        modify(position , value , Rson(x)) ;
    node[x].value = min(node[Lson(x)].value ,
        node[Rson(x)].value );
    //比較左右子樹哪個值比較小，較小值為此節點的 value

}

void push_down(int x, int add){
    //將懶人標記往下推，讓下一層子樹進行區間修改
    int lson = Lson(x), rson = Rson(x);
    node[lson].z += add;
        //給予懶人標記，表示子樹如果要給子樹的子樹區間修改
    node[rson].z += add;
        //數值要是多少，左右子樹都需要做

    node[lson].v += add; //更新左右子樹的值
    node[rson].v += add;
}

void update(int a, int b, int cmd, int x = 1){
//a, b 為區間修改的 left and right, cmd 為要增加的數值
    if(a <= node[x].l && b >= node[x].r){
        //如果節點的 left and right，跟 a, b
            區間是相等，或更小就，只要在這邊修改 cmd，
        //就可以讓 node[x].v
            的值直接變為區間修改後的數值，
        //之後如果要讓這查詢向子樹進行區間修改，就用
            push_down，
        //我們這邊的懶人標記就會告訴左右子樹要修改的值為多

        node[x].v += cmd; //區間修改後的 v
        node[x].z = cmd; //區間修改是要增加多少數值
        return;
    }
    push_down(x);//先將之前的區間查詢修改值，往下給子樹以
    //假如當前的 node[x].z 原本是 3，如果沒有
        push_down(x)，那下面的子樹都沒有被 +3，
    //導致答案不正確。

    int mid = (node[x].l+node[x].r) / 2;
        //切半，向下修改
    if(a <= mid) update(a, b, cmd, Lson(x));
        //如果要修改的點在左邊，就往左下角追蹤
    if(b > mid) update(a, b, cmd, Rson(x));
        //如果要修改的點在右邊，就往右下角追蹤
    node[x].v = node[Lson(x)].v + node[Rson(x)].v;
    //比較左右子樹哪個值比較小，較小值為此節點的 value
}
#define INF 0x3f3f3f

int query(int left , int right , int x = 1 ){
    if(node[x].left >= left && node[x].right <= right)
        return node[x].Min_Value ;
    //如果我們要查詢的區間比當前節點的區間大，那我們不需再向下查詢直接輸出此答案就好。
    // 例如我們要查詢 [2, 8]，我們只需要查詢 [3,
        4]，不須查詢 [3, 3]、[4, 4]，
    // [3, 4] 已經做到最小值查詢

    push_down(x);//有區間修改時才需要寫
    int mid = (node[x].left + node[x].right ) / 2 ;
        //切半，向下修改
    int ans = INF ; //一開始先假設答案為最大值

    if( left <= mid )
        //如果切半後，我們要查詢的區間有在左子樹就向下查詢
        ans = min(ans , query(left , right ,
            Lson(x))) ; //更新答案，比較誰比較小
    if(mid < right )
        //如果切半後，我們要查詢的區間有在右子樹就向下查詢
        ans = min(ans , query(left , right ,
            Rson(x))) ; //更新答案，比較誰比較小
    return ans ; //回傳答案
}
```

4.3 BIT

```
// BIT
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
// #include <ext/pb_ds/detail/standard_policies.hpp>
#define IOS
    ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define INF 0x3f3f3f3f
#define lowbit(x) x&(-x)
using namespace std;
using namespace __gnu_pbds;
typedef long long ll;
const int N = 2e5 + 5;
ll bit[N], n, q;
ll query(int idx){
    ll sum = 0;
    for(int i = idx;i > 0;i -= lowbit(i))
        sum += bit[i];
    return sum;
}
void update(ll val, int idx){
    for(int i = idx;i <= n;i += lowbit(i))
        bit[i] += val;
}
int main(){
    IOS
    cin >> n >> q;
    for(int i = 1;i <= n;i++){ // 1-based
        ll in;
        cin >> in;
        update(in, i);
    }
    while(q--){
        ll o, a, b;
        cin >> o >> a >> b;
        if(o == 1){
            ll u = query(a) - query(a - 1);
            update(b - u, a);
        }
        else{
            cout << query(b) - query(a - 1) << '\n';
        }
    }

    return 0;
}
```

4.4 Sparse Table

```
#include <bits/stdc++.h>
```

```cpp
#define IOS ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define INF 0x3f3f3f3f
using namespace std;
typedef long long ll;
const int N = 5e5 + 5;
int n, m, arr[N], dp[35][N];
void sparse_table(int n){
    for(int i = 1;i <= 31;i++){
        for(int j = 0;(j + (1LL << (i - 1))) < n;j++){
            dp[i][j] = max(dp[i - 1][j], dp[i - 1][j
                + (1LL << (i - 1))]);
        }
    }
}
int query(int l, int r){
    int idx = __lg(r - l + 1);
    return max(dp[idx][l], dp[idx][r - (1LL << idx) +
        1]);
}
int main(){
    IOS
    cin >> n;
    for(int i = 0;i < n;i++) cin >> arr[i];
    cin >> m;
    for(int i = 0;i < n;i++) dp[0][i] = arr[i];
    sparse_table(n);
    while(m--){
        int l, r;
        cin >> l >> r;
        if(l > r) swap(l, r);
        l--, r--;
        cout << query(l, r) << '\n';
    }

    return 0;
}
```

5 Uncategorized

5.1 快速冪

```cpp
#include <bits/stdc++.h>
#define IOS ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
using namespace std;
typedef long long ll;
ll mod = 1000000007;
ll fast_pow(int base, int exp){
    ll res = 1;
    while(exp > 0){
        if(exp & 1) res = res * base % mod;
        base = base * base % mod;
        exp >>= 1;
    }
    return res;
}
int main(){
    IOS
    int base = 3, exp = 15;
    cout << fast_pow(base, exp) << '\n';

    return 0;
}
```

5.2 矩陣快速冪

```cpp
#include <bits/stdc++.h>
#define IOS ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define INF 0x3f3f3f3f
using namespace std;
typedef long long ll;
ll mod = 1000000007;
struct Matrix{
    ll mat[2][2] = {{0}};
    Matrix operator * (Matrix &inp){
        Matrix tmp;
        for(int i = 0;i < 2;i++){
            for(int j = 0;j < 2;j++){
                for(int k = 0;k < 2;k++){
                    tmp.mat[i][j] = ((tmp.mat[i][j] +
                        (mat[i][k] % mod) *
                        (inp.mat[k][j] % mod)) % mod)
                        % mod;
                }
            }
        }
        return tmp;
    }
};
Matrix base;
Matrix fast_pow(int exp){
    if(exp == 1) return base;
    if(exp % 2 == 0){
        Matrix res = fast_pow(exp >> 1);
        return res * res;
    }
    Matrix res = fast_pow(exp >> 1);
    return base * res * res;
}
int main(){
    IOS
    base.mat[0][0] = 1;
    base.mat[0][1] = 4;
    base.mat[1][0] = 2;
    base.mat[1][1] = 3;
    Matrix output = fast_pow(10);
    for(int i = 0;i < 2;i++){
        for(int j = 0;j < 2;j++){
            cout << output.mat[i][j] << ' ';
        }
        cout << '\n';
    }

    return 0;
}
```

5.3 快速計算費氏數列

```cpp
#include <bits/stdc++.h>
#define IOS ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
#define INF 0x3f3f3f3f
using namespace std;
typedef long long ll;
ll mod = 1000000007;
struct Matrix{
    ll mat[2][2] = {{0}};
    Matrix operator * (Matrix &inp){
        Matrix tmp;
        for(int i = 0;i < 2;i++){
            for(int j = 0;j < 2;j++){
                for(int k = 0;k < 2;k++){
                    tmp.mat[i][j] = ((tmp.mat[i][j] +
                        (mat[i][k] % mod) *
                        (inp.mat[k][j] % mod)) % mod)
                        % mod;
                }
            }
        }
        return tmp;
    }
};
Matrix base;
Matrix fast_pow(int n){
    if(n == 1) return base;
    if(n % 2 == 0){
```

```
        Matrix res = fast_pow(n >> 1);
        return res * res;
    }
    Matrix res = fast_pow(n >> 1);
    return base * res * res;
}
int main(){
    IOS
    base.mat[0][0] = 1;
    base.mat[0][1] = 1;
    base.mat[1][0] = 1;
    base.mat[1][1] = 0;
    Matrix output = fast_pow(20);
    cout << output.mat[0][0] << '\n';

    return 0;
}
```

6 常用小扣

6.1 01 背包問題

```
// 01 backpack
for(int i=1;i<=n;i++){
    for(int j=x;j>=w[i];j--){
        dp[j]=max(dp[j],dp[j-w[i]]+v[i]);
    }
}
```

6.2 Topological

```
//拓鋪
int N, M, u, v, deg[MAXN];    // deg[i] 紀錄點 i 被連入邊數
vector<int> edge[MAXN];

/*------ 1. 計算 indegree ------*/
cin >> N >> M;
while(M--)
{
    cin >> u >> v;
    edge[u].push_back(v);
    ++deg[v];
}

/*------ 2. 將 indegree 為 0 的點放入 queue 中 ------*/
queue<int> q;                 // 紀錄待拔點
for(int i = 0; i < N; ++i)
    if(deg[i] == 0)
        q.push(i);

/*------ 3. 重複拔點，直到 queue 清空 ------*/
while(!q.empty())
{
    int cur = q.front();  q.pop();
    cout << cur << "\n";
    for(int i: edge[cur])
    {
        --deg[i];                              // 3-1. 相連點 indgree 減一
        if(deg[i] == 0) q.push(i);     // 3-2. 若該點 indgree 減至 0，則放入 queue 中
    }
}
```

6.3 常用

```
while(getline(ss,str,'m'))//以m分割
    cout << str << endl;
int a,b; char c;
while(ss>>a>>c>>b)//處理像是 -2/9+9/7
    cout<<a<<" "<<c<<" "<<b<<endl;
//輸出 -2 / 9 endl 9 / 7
```

7 Basic

7.1 GCD

```
int gcd(int a, int b){
    if(b == 0) return a;
    else return gcd(b, a % b);
}
```

7.2 LCM

```
int lcm(int a, int b){
    return a * b / gcd(a, b);
}
```

7.3 Leap Year

```
bool isLeap(int n){
    if(n % 100 == 0)
        if(n % 400 == 0) return true;
        else return false;
    if(n % 4 == 0) return true;
    else return false;
}
```

8 Trick

8.1 求根號和解決 pow 精度問題

```
// when the number is too large, use powl instead of pow
// will provide you more accuracy.
powl(a,b)
(int)round(p,(1.0/n)) //nth root of p
```

8.2 查找和二分搜

```
// will return address of iterator, call result as *iterator;
iterator find(iterator first, iterator last, const T&value);
bool binary_search(iterator first, itee)
```

8.3 $a^b mod p$

```
// a^b mod p

long powmod(long base, long exp , long modulus){
    base %= modulus;
    long result = 1;
    while(exp > 0){
        if(exp & 1) result = (result * base) % modulus;
        base = (base * base) % modulus;
        exp >> 1;
```

```
    }
    return result;
}
```

8.4 $n!modp$

```
// n! mod p

int factmod(int n,int p){
    long long res = 1;
    while(n > 1){
        res = (res % powmod(p-1 , n/p , p)) %p;
        for(int i=2;i<=n%p;i++)
            res = (res * i) % p;
        n /= p;
    }
}
```

8.5 從 1 n 選 M 個數字

```
// n >=m choose m number from 1 ~ n
void combination(int n, int m){
    if (n<m) return;

    int a[50] = {0};
    int k = 0;

    for(int i=1;i<=m;i++) a[i] = i;
    while(1){
        for(int i=1;i<=m;i++)
            cout << a[i] << " ";
        cout << endl;

        k = m;
        while((k>0) && (n-a[k] == m-k)) k --;
        if(k == 0) break;
        a[k] ++;
        for(int i=k+1;i<=m;i++){
            a[i] = a[i-1] + 1;
        }
    }
}
```

8.6 二項式係數 Binomial Coefficient（也可以用在 C 幾取幾）

```
#define MAXN 100 // largest n or m
long binomial_coefficient(n, m) // compute n choose m
int n, m;{
    int i, j;
    long bc[MAXN][MAXN];
    for(i = 0;i <= n;i++) bc[i][0] = 1;
    for(j = 0;j <= n;j++) bc[j][j] = 1;
    for(i = 1;i <= n;i++)
        for(j = 1;j < i;j++)
            bc[i][j] = bc[i - 1][j - 1] + bc[i -
                1][j];
    reutrn bc[n][m];
}
```

8.7 LICS

```
int a[100] = {0};
int b[100] = {0};
int f[100] = {0};
int n = 0, m = 0;
int main(void){
    cin >> n;
    for(int i = 1;i <= n;i++) cin >> a[i];
    cin >> m;
    for(int i = 1;i <= m;i++) cin >> b[i];
    for(int i = 1;i <= n;i++){
        int k = 0;
        for(int j = 1;j <= m;j++){
            if(a[i] > b[j] && f[j] > k) k = f[j];
            else if(a[i] == b[j] && k + 1 > f[j])
                f[j] = k + 1;
        }
    }
}
```

8.8 m-ary to 10-ary

```
string num = "0123456789ABCDE";

int mToTen(string n, int m){
    int multi = 1;
    int result = 0;

    for(int i = n.size() - 1;i >= 0;i--){
        result += num.find(n[i]) * multi;
        multi *= m;
    }
    return result;
}
```

9 Math

9.1 ModInverse

```
// 求 a 在模 m 下的模反元素，m 必須是質數
ll modInverse(ll a, ll m) {
    ll res = 1, exp = m - 2;
    while (exp > 0) {
        if (exp % 2 == 1) res = (res * a) % m;
        a = (a * a) % m;
        exp /= 2;
    }
    return res;
}
```

9.2 CRT

```
ll extendedGCD(ll a, ll b, ll &x, ll &y){
    if(b == 0){
        x = 1, y = 0;
        return a;
    }
    ll x1, y1;
    ll g = extendedGCD(b, a % b, x1, y1);
    x = y1;
    y = x1 - (a / b) * y1;
    return g;
}

ll chineseRemainder(vector<ll> &n, vector<ll> &a){
    ll prod = 1;
    for(ll num : n) prod *= num; // Calculate the
        product of all moduli

    ll result = 0;
    for(int i = 0; i < n.size(); i++){
        ll p = prod / n[i]; // Product divided by
            current modulus
        ll x, y;
        extendedGCD(p, n[i], x, y); // Solve p * x
            1 (mod n[i])
        result = (result + a[i] * x * p) % prod; //
            Add current term to result
```

```
    }
    return (result + prod) % prod; // Return the
        result modulo the product
}
```

9.3 LinearCongruence

```
ll solveLinearCongruence(ll a, ll b, ll m){
    ll x, y;
    ll g = extendedGCD(a, m, x, y); // Solve a * x +
        m * y = gcd(a, m)
    if (b % g != 0) return -1; // No solution if
        gcd(a, m) does not divide b

    x = (x * (b / g)) % m;
    if (x < 0) x += m; // Ensure the result is
        positive
    return x;
}
```