

Contents

1	DP	1
1.1	LCS	1
1.2	LIS $O(n^2)$	1
1.3	LIS $O(n \log n)$	1
1.4	LIS $O(n \log n)$	1
2	Prime	2
2.1	質數篩 CPP	2
2.2	質數篩 PY	2
2.3	單一質數	2
2.4	egcd CPP	2
2.5	egcd PY	2
3	Graph	2
3.1	Floyd Warshall	2
3.2	Bellman Ford	2
3.3	SPFA	3
3.4	Dijkstra	3
3.5	SecondSP	3
3.6	Kruskal's Algorithm	4
3.7	SecondMST	4
3.8	Prim's Algorithm	5
3.9	LCA	5
3.10	Topological Sort	5
4	RMQ	6
4.1	Segment Tree V1	6
4.2	Segment Tree V2	6
4.3	BIT	7
4.4	Sparse Table	7
5	Uncategorized	8
5.1	快速幂	8
5.2	矩陣快速幂	8
5.3	快速計算費氏數列	8
6	常用小扣	9
6.1	01 背包問題	9
6.2	Topological	9
6.3	常用	9
7	Basic	9
7.1	GCD	9
7.2	LCM	9
7.3	Leap Year	9
8	Trick	9
8.1	求根號和解決 pow 精度問題	9
8.2	查找和二分搜	9
8.3	$a^b \bmod p$	9
8.4	$n! \bmod p$	10
8.5	從 1 ~ n 選 M 個數字	10
8.6	二項式係數 Binomial Coefficient (也可以用在 C 幾取幾)	10
8.7	LICS	10
8.8	m-ary to 10-ary	10
9	Math	10
9.1	ModInverse	10
9.2	CRT	10
9.3	LinearCongruence	11

1 DP

1.1 LCS

```

1 #include <bits/stdc++.h>
2 #define IOS
3   ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 using namespace std;
5 string s1, s2;
6 int dp[505][505];
7 int main(){
8   IOS
9   cin >> s1 >> s2;
10  memset(dp, 0, sizeof(dp));
11  int l1 = s1.size(), l2 = s2.size();
12  for(int i = 1; i <= l1; i++){
13    for(int j = 1; j <= l2; j++){
14      if(s1[i - 1] == s2[j - 1]) dp[i][j] =
15        dp[i - 1][j - 1] + 1;
16      else dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
17    }
18  }

```

```

15   }
16 }
17 cout << dp[l1][l2] << '\n';
18
19 return 0;
20 }

```

1.2 LIS $O(n^2)$

```

1 #include <bits/stdc++.h>
2 #define IOS
3   ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 using namespace std;
5 typedef long long ll;
6 int main(){
7   IOS
8   int arr[100];
9   int n;
10  cin >> n;
11  for(int i = 0; i < n; i++) cin >> arr[i];
12  int dp[100];
13  for(int i = 0; i < n; i++) dp[i] = 1;
14  for(int i = 0; i < n; i++){
15    for(int j = 0; j < i; j++){
16      if(arr[i] > arr[j])
17        dp[i] = max(dp[j] + 1, dp[i]);
18    }
19  }
20  int ans = 1;
21  for(int i = 0; i < n; i++) ans = max(ans, dp[i]);
22  cout << ans << '\n';
23
24 return 0;

```

1.3 LIS $O(n \log n)$

```

1 class Solution {
2 public:
3   int lengthOfLIS(vector<int>& nums) {
4     vector<int> v;
5     int n = nums.size();
6     for(int i = 0; i < n; i++){
7       int p = lower_bound(v.begin(), v.end(),
8         nums[i]) - v.begin();
9       if(p == v.size()) v.push_back(nums[i]);
10      else v[p] = nums[i];
11    }
12    return v.size();
13 };

```

1.4 LIS $O(n \log n)$

```

1 for(int i=0;i<num.size();i++){
2   if(lis.empty()||lis.back()<num[i]){
3     lis.push_back(num[i]);
4     dp[i]=lis.size();
5   }
6   else{
7     auto iter=lower_bound(all(lis),num[i]);
8     dp[i]=iter-lis.begin()+1;
9     *iter=num[i];
10  }
11 }
12 int length=lis.size();
13 for(int i=num.size()-1;i>=0;i--){
14   if(dp[i]==length){
15     ans.push_back(num[i]);
16     length--;
17   }
18 }

```

2 Prime

2.1 質數篩 CPP

```

1 bitset<MAXN> prime_bool;
2 vector<ll> prime;
3 void find_prime(){
4     prime_bool.set();
5     for(int i=2;i<MAXN;i++){
6         if(prime_bool[i]){
7             prime.push_back(i);
8         }
9         for(auto j:prime){
10             if(j*i>=MAXN)
11                 break;
12             prime_bool[j*i]=0;
13             if(i%j==0)
14                 break;
15         }
16     }
17 }

```

2.2 質數篩 PY

```

1 is_prime = n * [1]
2 is_prime[0] = is_prime[1] = 0
3
4 for i in range(2, n):
5     if is_prime[i]:
6         for j in range(2, n):
7             if i * j >= n:
8                 break
9             is_prime[i * j] = 0

```

2.3 單一質數

```

1 bool prime(int n){
2     if(n < 2) return false;
3     if(n <= 3) return true;
4     if(!(n % 2) || !(n % 3)) return false;
5     for(int i = 5; i * i <= n; i += 6)
6         if(!(n % i) || !(n % (i + 2))) return false;
7     return true;
8 }

```

2.4 egcd CPP

```

1 int exgcd(int a,int b,int &x,int &y){
2     if(b==0){
3         x=1,y=0;
4         return a;
5     }
6     int gcd=exgcd(b,a%b,y,x);
7     y-=a/b*x;
8     return gcd;
9 }

```

2.5 egcd PY

```

1 def egcd(a: int, b: int) -> Tuple[int, int, int]:
2     """return (g, x, y) such that a*x + b*y = g =
3         gcd(a, b)"""
4     if a == 0:
5         return (b, 0, 1)
6     else:
7         b_div_a, b_mod_a = divmod(b, a)
8         g, x, y = egcd(b_mod_a, a)
9         return (g, y - b_div_a * x, x)

```

3 Graph

3.1 Floyd Warshall

```

1 int n, rd, l, r, v;
2 cin >> n >> rd;
3 vector<vector<int>> dp(n + 1, vector<int>(n + 1,
4     1e9));
5 for(int i = 0; i < rd; i++){
6     cin >> l >> r >> v;
7     dp[l][r] = dp[r][l] = v;
8     //每條路皆雙向
9 }
10 //以下即 Floyd-Warshall
11 for(int k = 1; k <= n; k++){
12     for(int i = 1; i <= n; i++){
13         for(int j = 1; j <= n; j++){
14             dp[i][j] = min(dp[i][k] + dp[k][j],
15                 dp[i][j]);
16             //窮舉所有鬆弛可能
17         }
18     }
19 }
20 cin >> l >> r;
21 cout << dp[l][r];

```

3.2 Bellman Ford

```

1 #include <bits/stdc++.h>
2 #define IOS
3 ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 #define INF 0x3f3f3f3f
5 using namespace std;
6 typedef long long ll;
7 struct Edge{
8     int x, y, t;
9 };
10 int dis[1005];
11 int main(){
12     IOS
13     int c;
14     cin >> c;
15     while(c--){
16         vector<Edge> edge;
17         int n, m;
18         cin >> n >> m;
19         for(int i = 0; i <= n; i++) dis[i] = INF;
20         dis[0] = 0;
21         for(int i = 0; i < m; i++){
22             int x, y, t;
23             cin >> x >> y >> t;
24             edge.push_back({x, y, t});
25         }
26         for(int i = 0; i < n - 1; i++){
27             for(int j = 0; j < m; j++){
28                 if(dis[edge[j].x] + edge[j].t <
29                     dis[edge[j].y]){
30                     dis[edge[j].y] = dis[edge[j].x] + edge[j].t;
31                 }
32             }
33         }
34         bool judge = true;
35         for(auto e : edge){
36             if(dis[e.x] + e.t < dis[e.y]){
37                 judge = false;
38                 break;
39             }
40         }
41         cout << (judge ? "not possible" : "possible") <<
42             '\n';
43     }
44     return 0;

```

43 }

3.3 SPFA

```

1 #define mem(x) memset(x, 0, sizeof(x))
2 struct road{
3     int r, val;
4 };
5 int main(){
6     int n, e, l, r, v;
7     cin >> n >> e;
8     vector<int> dp(n + 1, 1e9);
9     vector<road> rd[n + 1];
10    for(int i = 0; i < e; i++){
11        cin >> l >> r >> v;
12        rd[l].push_back({r, v});
13        rd[r].push_back({l, v});
14    }
15    cin >> l >> r;
16    dp[l] = 0;
17    queue<int> que;
18    que.push(l);
19    bool check[n + 1]; mem(check);
20    int cnt[n + 1]; mem(cnt);
21    while(!que.empty()){
22        int tmp = que.front(); que.pop();
23        check[tmp] = 0, cnt[tmp]++;
24        if(cnt[tmp] >= n) {cout << "neg cycle\n"; break;}
25        for(auto & i : rd[tmp]){
26            if(dp[i.r] > dp[tmp] + i.val){
27                dp[i.r] = dp[tmp] + i.val;
28                if(!check[i.r]) check[i.r] = 1, que.push(i.r);
29            }
30        }
31    }
32    for(auto & i : dp) cout << i << ' ';
33    return 0;
34 }

```

3.4 Dijkstra

```

1 #include <iostream>
2 #include <algorithm>
3 #include <vector>
4 #include <queue>
5 #define IOS
6     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
7 #define INF 2147483647
8 using namespace std;
9 int n, m;
10 vector<pair<int, int>> adj[100005];
11 bool visited[100005] = {false};
12 priority_queue<pair<int, int>> pq;
13 int dis[100005], parent[100005];
14 void solve(){ // Dijkstra
15     dis[0] = 0;
16     for(int i = 1; i < n; i++) dis[i] = INF;
17     pq.push(make_pair(0, 0));
18     while(!pq.empty()){
19         auto node = pq.top();
20         pq.pop();
21         int v = node.second; // parent
22         if(visited[v]) continue;
23         visited[v] = true;
24         for(auto i : adj[v]){
25             int vertex = i.first, weight = i.second;
26             if(visited[vertex]) continue;
27             if(dis[v] + weight < dis[vertex]){
28                 dis[vertex] = dis[v] + weight;
29                 parent[vertex] = v;
30                 pq.push(make_pair(-dis[vertex],

```

```

31             }
32         }
33     }
34     int maxd = -1, cnt = 0;
35     for(int i = 0; i < n; i++){
36         if(dis[i] < INF){
37             if(dis[i] > maxd) maxd = dis[i];
38             else cnt++;
39         }
40     }
41     cout << maxd << '\n' << cnt << '\n';
42 }
43 int main(){
44     IOS
45     cin >> n >> m;
46     for(int i = 0; i < m; i++){
47         int u, v, w;
48         cin >> u >> v >> w;
49         adj[u].push_back(make_pair(v, w));
50         adj[v].push_back(make_pair(u, w));
51     }
52     solve();
53     return 0;
54 }

```

3.5 SecondSP

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 #define INF 0x3f3f3f3f
5 using namespace std;
6 typedef pair<int, int> pii;
7 typedef long long ll;
8
9 const int MAXN = 1005;
10
11 vector<pii> adj[MAXN]; // adj[u] stores pairs {v,
12 // weight}
13 int dist[MAXN], sec_dist[MAXN]; // shortest and
14 // second shortest distances
15
16 void dijkstra(int s, int n){
17     // Min-heap to store {distance, node}
18     priority_queue<pii, vector<pii>, greater<pii>> pq;
19     fill(dist, dist + n + 1, INF);
20     fill(sec_dist, sec_dist + n + 1, INF);
21
22     dist[s] = 0;
23     pq.push({0, s});
24
25     while(!pq.empty()){
26         int d = pq.top().first;
27         int u = pq.top().second;
28         pq.pop();
29
30         // If we found a path longer than the second
31         // shortest, skip it
32         if (sec_dist[u] < d) continue;
33
34         for (auto &edge : adj[u]){
35             int v = edge.first;
36             int w = edge.second;
37
38             int new_dist = d + w;
39
40             // If this gives a new shortest path to v
41             if(new_dist < dist[v]){
42                 sec_dist[v] = dist[v];
43                 dist[v] = new_dist;
44                 pq.push({dist[v], v});
45                 pq.push({sec_dist[v], v});
46             }
47
48             // If this gives a new second shortest
49             // path to v

```

```

45         else if(new_dist > dist[v] && new_dist <
46             sec_dist[v]){
47             sec_dist[v] = new_dist;
48             pq.push({sec_dist[v], v});
49         }
50     }
51 }
52
53 int main() {
54     IOS
55     int t;
56     cin >> t;
57     while(t--){
58         int n, m, s, d;
59         cin >> n >> m >> s >> d;
60
61         // Reset adjacency list
62         for(int i = 1; i <= n; i++) adj[i].clear();
63
64         for(int i = 0; i < m; i++){
65             int u, v, w;
66             cin >> u >> v >> w;
67             adj[u].push_back({v, w});
68             adj[v].push_back({u, w}); // If the graph
69                                     // is undirected
70         }
71
72         dijkstra(s, n);
73
74         if(sec_dist[d] == INF) cout << -1 << '\n'; //
75         // No second shortest path
76         else cout << sec_dist[d] << '\n';
77     }
78     return 0;

```

3.6 Kruskal's Algorithm

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 using namespace std;
5 int parent[10005];
6 struct Edge{
7     int u, v, w;
8     bool operator < (Edge &b){
9         return w < b.w;
10    };
11 int query(int a){
12     if(parent[a] == -1) return a;
13     return parent[a] = query(parent[a]);
14 }
15 bool merge(int a, int b){
16     int r1 = query(a);
17     int r2 = query(b);
18     if(r1 == r2) return false;
19     if(parent[r1] < parent[r2]) parent[r2] = r1;
20     else parent[r1] = r2;
21     return true;
22 }
23 int main(){
24     IOS
25     int n, m;
26     memset(parent, -1, sizeof(parent));
27     cin >> n >> m;
28     vector<Edge> adj;
29     for(int i = 0; i < m; i++){
30         int u, v, w;
31         cin >> u >> v >> w;
32         adj.push_back({u, v, w});
33     }
34     sort(adj.begin(), adj.end());

```

```

35     // for(int i = 0; i < m; i++) cout << adj[i].w << '
36     ';
37     int cost = 0, n_edge = 0;
38     for(Edge e : adj){
39         if(merge(e.u, e.v)){
40             cost += e.w;
41             n_edge++;
42         }
43     }
44     if(n_edge == n - 1) cout << cost << '\n';
45     else cout << -1 << '\n';
46     return 0;
47 }

```

3.7 SecondMST

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 #define INF 0x3f3f3f3f
5 using namespace std;
6 typedef long long ll;
7 int anc[105], sz[105];
8 struct Edge{
9     int a, b, c;
10 };
11 int Find(int a){
12     return (anc[a] == a ? a : anc[a] = Find(anc[a]));
13 }
14 bool merge(int a, int b){
15     a = Find(a);
16     b = Find(b);
17     if(a == b) return false;
18     if(sz[a] < sz[b]) swap(a, b);
19     anc[b] = a;
20     sz[a] += sz[b];
21     return true;
22 }
23 int main(){
24     IOS
25     int t;
26     cin >> t;
27     while(t--){
28         int n, m;
29         vector<Edge> edge;
30         cin >> n >> m;
31         vector<pair<int, int>> vec[105];
32         for(int i = 0; i < m; i++){
33             int a, b, c;
34             cin >> a >> b >> c;
35             edge.push_back({a, b, c});
36         }
37         for(int i = 1; i <= n; i++){
38             sz[i] = 0;
39             anc[i] = i;
40         }
41         sort(edge.begin(), edge.end(), [&](Edge &u,
42             Edge &v){return u.c < v.c;});
43         int cost1 = 0, cnt = 0;
44         vector<int> mst;
45         for(int i = 0; i < edge.size(); i++){
46             if(merge(edge[i].a, edge[i].b)){
47                 cost1 += edge[i].c;
48                 mst.push_back(i);
49                 if(++cnt == n - 1) break;
50             }
51         }
52         int cost2 = INF;
53         for(int i = 0; i < mst.size(); i++){
54             cnt = 0;
55             int res = 0;
56             for(int i = 1; i <= n; i++) anc[i] = i;
57             for(int j = 0; j < edge.size(); j++){
58                 if(mst[i] == j) continue;
59                 if(merge(edge[j].a, edge[j].b)){

```

```

58         res += edge[j].c;
59         if(++cnt == n - 1){
60             cost2 = min(cost2, res);
61             break;
62         }
63     }
64 }
65 }
66
67 cout << cost1 << ' ' << cost2 << '\n';
68 }
69
70 return 0;
71 }

```

3.8 Prim's Algorithm

```

1 #include <iostream>
2 #include <queue>
3 #include <algorithm>
4 #include <cstring>
5 #define IOS
6     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
7 using namespace std;
8 int n, m, dis[10005], parent[10005];
9 bool visited[10005] = {false};
10 vector<pair<int, int>> adj[10005];
11 int main(){
12     IOS
13     // freopen("input.in", "r", stdin);
14     cin >> n >> m;
15     memset(dis, 0x3f3f3f3f, sizeof(dis));
16     memset(parent, -1, sizeof(parent));
17     for(int i = 0; i < m; i++){
18         int u, v, w;
19         cin >> u >> v >> w;
20         adj[u].push_back({v, w});
21         adj[v].push_back({u, w});
22     }
23     int start = 0;
24     dis[start] = 0;
25     priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
26     pq.push({dis[start], start});
27     while(!pq.empty()){
28         pair<int, int> cur = pq.top();
29         pq.pop();
30         if(visited[cur.second]) continue;
31         visited[cur.second] = true;
32         for(auto i : adj[cur.second]){
33             if(visited[i.first]) continue;
34             if(dis[i.first] > i.second){
35                 dis[i.first] = i.second;
36                 parent[i.first] = cur.second;
37                 pq.push({dis[i.first], i.first});
38             }
39         }
40     }
41     int cost = 0, err = 0;
42     for(int i = 0; i < n; i++){
43         if(dis[i] < 0x3f3f3f3f) cost += dis[i];
44         else err++;
45     }
46     cout << (err ? -1 : cost) << "\n";
47     // for(int i = 0; i < n; i++) cout << dis[i] << ' ';
48     return 0;
49 }

```

3.9 LCA

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);

```

```

3 #define INF 0x3f3f3f3f
4 using namespace std;
5 typedef long long ll;
6 const int N = 2e5 + 5;
7 int n, q;
8 vector<int> vec[N];
9 int p[20][N], in[N], out[N];
10 bool valid(int a, int b){
11     return (in[a] <= in[b] && out[b] <= out[a]);
12 }
13 void dfs(int cur, int par){
14     static int t = 0;
15     p[0][cur] = par;
16     in[cur] = t++;
17     for(auto e : vec[cur]){
18         dfs(e, cur);
19     }
20     out[cur] = t++;
21 }
22 int lca(int a, int b){
23     if(valid(a, b)) return a;
24     for(int i = 19; i >= 0; i--){
25         if(!valid(p[i][a], b)) a = p[i][a];
26     }
27     return p[0][a];
28 }
29 int main(){
30     IOS
31     cin >> n >> q;
32     for(int i = 2; i <= n; i++){
33         int e;
34         cin >> e;
35         vec[e].push_back(i);
36     }
37     dfs(1, 1);
38     for(int i = 1; i < 20; i++){
39         for(int j = 1; j <= n; j++){
40             p[i][j] = p[i - 1][p[i - 1][j]];
41         }
42     }
43     while(q--){
44         int u, v;
45         cin >> u >> v;
46         cout << lca(u, v) << '\n';
47     }
48     return 0;
49 }

```

3.10 Topological Sort

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 using namespace std;
5 typedef long long ll;
6 vector<int> vec[200005];
7 int ind[100005];
8 int main(){
9     IOS
10     int n, m;
11     cin >> n >> m;
12     memset(ind, 0, sizeof(ind));
13     for(int i = 0; i < m; i++){
14         int a, b;
15         cin >> a >> b;
16         ind[b]++;
17         vec[a].push_back(b);
18     }
19     queue<int> q;
20     for(int i = 1; i <= n; i++){
21         if(ind[i] == 0) q.push(i);
22     }
23     vector<int> top;
24     while(!q.empty()){
25         int cur = q.front();

```

```

25     q.pop();
26     top.push_back(cur);
27     for(auto e : vec[cur]){
28         ind[e]--;
29         if(ind[e] == 0){
30             q.push(e);
31         }
32     }
33 }
34 if(top.size() == n){
35     for(auto i : top) cout << i << ' ';
36     cout << '\n';
37 }
38 else cout << "IMPOSSIBLE" << '\n';
39
40 return 0;
41 }

```

4 RMQ

4.1 Segment Tree V1

```

1 #include <bits/stdc++.h>
2 #define IOS
3     ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
4 #define L(x) (x << 1)
5 #define R(x) ((x << 1) | 1)
6 using namespace std;
7 typedef long long ll;
8 ll seg[500005 << 2], lazy[500005 << 2];
9 int n, q;
10 void init(){
11     memset(seg, 0, sizeof(seg));
12     memset(lazy, 0, sizeof(lazy));
13 }
14 void build(int x, int l, int r){
15     if(l == r){
16         cin >> seg[x];
17         return;
18     }
19     int mid = (l + r) >> 1;
20     build(L(x), l, mid);
21     build(R(x), mid + 1, r);
22     seg[x] = seg[L(x)] + seg[R(x)];
23 }
24 void push(int pos, int size){
25     lazy[L(pos)] += lazy[pos];
26     lazy[R(pos)] += lazy[pos];
27     seg[pos] = seg[pos] + lazy[pos] * size;
28     lazy[pos] = 0;
29 }
30 void modify(int x, int l, int r, int ql, int qr, int val){
31     if(lazy[x]) push(x, (r - l) + 1);
32     // seg[x] = seg[L(x)] + (mid - l) * lazy[L(x)] +
33     // seg[R(x)] + (r - mid) * lazy[R(x)];
34     seg[x] += val * (qr - ql + 1);
35     if(ql <= l && qr >= r){
36         lazy[x] += val;
37         return;
38     }
39     int mid = (l + r) >> 1;
40     if(qr <= mid) modify(L(x), l, mid, ql, qr, val);
41     else if(ql > mid) modify(R(x), mid + 1, r, ql, qr, val);
42     else{
43         modify(L(x), l, mid, ql, mid, val);
44         modify(R(x), mid + 1, r, mid + 1, qr, val);
45     }
46 }
47 ll query(int x, int l, int r, int ql, int qr){
48     if(ql <= l && qr >= r) return seg[x] + lazy[x] * (r - l);
49     if(lazy[x]) push(x, (r - l) + 1);
50     int mid = (l + r) >> 1;
51     if(qr <= mid) return query(L(x), l, mid, ql, qr);
52     else if(ql > mid) return query(R(x), mid + 1, r, ql, qr);
53     else return query(L(x), l, mid, ql, mid) + query(R(x), mid + 1, r, mid + 1, qr);
54 }
55 int main(){
56     IOS
57     init();
58     cin >> n;
59     build(1, 1, n);
60     cin >> q;
61     while(q--){
62         int v, x, y, k;
63         cin >> v;
64         if(v == 1){
65             cin >> x >> y >> k;
66             modify(1, 1, n, x, y, k);
67         }
68         else{
69             cin >> x >> y;
70             ll ans = query(1, 1, n, x, y);
71             cout << ans << '\n';
72         }
73     }
74     return 0;
75 }

```

4.2 Segment Tree V2

```

1 struct Node{
2     int left; // 左邊邊界
3     int right; // 右邊邊界
4     int value; // 儲存的值
5     int z; // 區間修改用，如果沒有區間修改就不需要
6 }node[4 * N];
7
8 #define Lson(x) (x << 1) // 左子樹
9 #define Rson(x) ((x << 1) + 1) // 右子樹
10
11 void question(){
12     for(int i = 1; i <= 10; i++) num[i] = i * 123 % 5;
13     // num 為題目產生的一段數列
14     // hash 函數，讓 num 的 i 被隨機打亂
15 }
16
17 void build(int left, int right, int x = 1){
18     // 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
19     node[x].left = left; // 給 x 節點左右邊界
20     node[x].right = right;
21     if(left == right){
22         // 如果左右邊界節點相同，表示這裡是葉節點
23         node[x].value = num[left]; // 把 num 值給 node[x]
24         // 這裡的 num 值表示，我們要在 value 要放的值
25         return; // 向前返回
26     }
27     int mid = (left + right) / 2; // 切半，產生二元樹
28
29     // debug
30     // cout << mid << '\n';
31     // cout << x << ' ' << node[x].left << ' ' << node[x].right << ' ' << '\n';
32
33     build(left, mid, Lson(x)); // 將區間改為 [left, mid] 然後帶給左子樹
34     build(mid + 1, right, Rson(x)); // 將區間改為 [mid+1, right] 然後帶給右子樹
35 }

```

```

34     node[x].value = min(node[Lson(x)].value ,
35         node[Rson(x)].value ) ;
36     //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間
37 }
38 void modify(int position , int value , int x = 1 ){
39     //修改數字
40     if(node[x].left == position && node[x].right ==
41         position ){ //找到葉節點
42         node[x].value = value ; //修改
43         return ; //傳回
44     }
45     int mid = (node[x].left + node[x].right ) / 2 ;
46     //切半，向下修改
47     if(position <= mid )
48         //如果要修改的點在左邊，就往左下角追蹤
49         modify(position , value , Lson(x)) ;
50     if(mid < position )
51         //如果要修改的點在右邊，就往右下角追蹤
52         modify(position , value , Rson(x)) ;
53     node[x].value = min(node[Lson(x)].value ,
54         node[Rson(x)].value ) ;
55     //比較左右子樹哪個值比較小，較小值為此節點的 value
56 }
57
58 void push_down(int x, int add){
59     //將懶人標記往下推，讓下一層子樹進行區間修改
60     int lson = Lson(x) , rson = Rson(x);
61     node[lson].z += add;
62     //給予懶人標記，表示子樹如果要給子樹的子樹區間修改
63     node[rson].z += add;
64     //數值要是多少，左右子樹都需要做
65
66     node[lson].v += add; //更新左右子樹的值
67     node[rson].v += add;
68 }
69
70 void update(int a, int b, int cmd, int x = 1){
71     //a, b 為區間修改的 left and right, cmd 為要增加的數值
72     if(a <= node[x].l && b >= node[x].r){
73         //如果節點的 left and right，跟 a, b
74         //區間是相等，或更小就，只要在這邊修改 cmd，
75         //就可以讓 node[x].v
76         //的值直接變為區間修改後的數值，
77         //之後如果要讓這查詢向子樹進行區間修改，就用
78         //push_down，
79         //我們這邊的懶人標記就會告訴左右子樹要修改的值為多少
80
81         node[x].v += cmd; //區間修改後的 v
82         node[x].z = cmd; //區間修改是要增加多少數值
83         return;
84     }
85     push_down(x); //先將之前的區間查詢修改值，往下給子樹以懶人標記
86     //假如當前的 node[x].z 原本是 3，如果沒有
87     //push_down(x)，那下面的子樹都沒有被 +3，
88     //導致答案不正確。
89
90     int mid = (node[x].l+node[x].r) / 2;
91     //切半，向下修改
92     if(a <= mid) update(a, b, cmd, Lson(x));
93     //如果要修改的點在左邊，就往左下角追蹤
94     if(b > mid) update(a, b, cmd, Rson(x));
95     //如果要修改的點在右邊，就往右下角追蹤
96     node[x].v = node[Lson(x)].v + node[Rson(x)].v;
97     //比較左右子樹哪個值比較小，較小值為此節點的 value
98 }
99 #define INF 0x3f3f3f
100
101 int query(int left , int right , int x = 1 ){
102     if(node[x].left >= left && node[x].right <= right)
103         return node[x].Min_Value ;
104     //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。

```

```

90     // 例如我們要查詢 [2, 8]，我們只需要查詢 [3,
91     4]，不須查詢 [3, 3]、[4, 4]，
92     // [3, 4] 已經做到最小值查詢
93
94     push_down(x); //有區間修改時才需要寫
95     int mid = (node[x].left + node[x].right ) / 2 ;
96     //切半，向下修改
97     int ans = INF ; //一開始先假設答案為最大值
98
99     if( left <= mid )
100         //如果切半後，我們要查詢的區間有在左子樹就向下查詢
101         ans = min(ans , query(left , right ,
102             Lson(x))) ; //更新答案，比較誰比較小
103     if(mid < right )
104         //如果切半後，我們要查詢的區間有在右子樹就向下查詢
105         ans = min(ans , query(left , right ,
106             Rson(x))) ; //更新答案，比較誰比較小
107     return ans ; //回傳答案
108 }

```

4.3 BIT

```

1 // BIT
2 #include <bits/stdc++.h>
3 #include <ext/pb_ds/assoc_container.hpp>
4 #include <ext/pb_ds/tree_policy.hpp>
5 // #include <ext/pb_ds/detail/standard_policies.hpp>
6 #define IOS
7 ios_base::sync_with_stdio(false); cin.tie(0); cout.tie(0);
8 #define INF 0x3f3f3f3f
9 #define lowbit(x) x&(-x)
10 using namespace std;
11 using namespace __gnu_pbds;
12 typedef long long ll;
13 const int N = 2e5 + 5;
14 ll bit[N], n, q;
15 ll query(int idx){
16     ll sum = 0;
17     for(int i = idx; i > 0; i -= lowbit(i))
18         sum += bit[i];
19     return sum;
20 }
21 void update(ll val, int idx){
22     for(int i = idx; i <= n; i += lowbit(i))
23         bit[i] += val;
24 }
25 int main(){
26     IOS
27     cin >> n >> q;
28     for(int i = 1; i <= n; i++){ // 1-based
29         ll in;
30         cin >> in;
31         update(in, i);
32     }
33     while(q--){
34         ll o, a, b;
35         cin >> o >> a >> b;
36         if(o == 1){
37             ll u = query(a) - query(a - 1);
38             update(b - u, a);
39         }
40         else{
41             cout << query(b) - query(a - 1) << '\n';
42         }
43     }
44     return 0;
45 }

```

4.4 Sparse Table

```

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

```



```

2 #define IOS
   ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
3 #define INF 0x3f3f3f3f
4 using namespace std;
5 typedef long long ll;
6 const int N = 5e5 + 5;
7 int n, m, arr[N], dp[35][N];
8 void sparse_table(int n){
9     for(int i = 1; i <= 31; i++){
10         for(int j = 0; (j + (1LL << (i - 1))) < n; j++){
11             dp[i][j] = max(dp[i - 1][j], dp[i - 1][j
               + (1LL << (i - 1))]);
12         }
13     }
14 }
15 int query(int l, int r){
16     int idx = __lg(r - l + 1);
17     return max(dp[idx][l], dp[idx][r - (1LL << idx) +
               1]);
18 }
19 int main(){
20     IOS
21     cin >> n;
22     for(int i = 0; i < n; i++) cin >> arr[i];
23     cin >> m;
24     for(int i = 0; i < n; i++) dp[0][i] = arr[i];
25     sparse_table(n);
26     while(m--){
27         int l, r;
28         cin >> l >> r;
29         if(l > r) swap(l, r);
30         l--, r--;
31         cout << query(l, r) << '\n';
32     }
33
34     return 0;
35 }

```

```

5 typedef long long ll;
6 ll mod = 1000000007;
7 struct Matrix{
8     ll mat[2][2] = {{0}};
9     Matrix operator * (Matrix &inp){
10         Matrix tmp;
11         for(int i = 0; i < 2; i++){
12             for(int j = 0; j < 2; j++){
13                 for(int k = 0; k < 2; k++){
14                     tmp.mat[i][j] = ((tmp.mat[i][j] +
                           (mat[i][k] % mod) *
                           (inp.mat[k][j] % mod)) % mod)
                           % mod;
15                 }
16             }
17         }
18         return tmp;
19     }
20 };
21 Matrix base;
22 Matrix fast_pow(int exp){
23     if(exp == 1) return base;
24     if(exp % 2 == 0){
25         Matrix res = fast_pow(exp >> 1);
26         return res * res;
27     }
28     Matrix res = fast_pow(exp >> 1);
29     return base * res * res;
30 }
31 int main(){
32     IOS
33     base.mat[0][0] = 1;
34     base.mat[0][1] = 4;
35     base.mat[1][0] = 2;
36     base.mat[1][1] = 3;
37     Matrix output = fast_pow(10);
38     for(int i = 0; i < 2; i++){
39         for(int j = 0; j < 2; j++){
40             cout << output.mat[i][j] << ' ';
41         }
42         cout << '\n';
43     }
44
45     return 0;
46 }

```

5 Uncategorized

5.1 快速幂

```

1 #include <bits/stdc++.h>
2 #define IOS
   ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
3 using namespace std;
4 typedef long long ll;
5 ll mod = 1000000007;
6 ll fast_pow(int base, int exp){
7     ll res = 1;
8     while(exp > 0){
9         if(exp & 1) res = res * base % mod;
10        base = base * base % mod;
11        exp >>= 1;
12    }
13    return res;
14 }
15 int main(){
16     IOS
17     int base = 3, exp = 15;
18     cout << fast_pow(base, exp) << '\n';
19
20     return 0;
21 }

```

5.2 矩陣快速幂

```

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

```

5.3 快速計算費氏數列

```

1 #include <bits/stdc++.h>
2 #define IOS
   ios_base::sync_with_stdio(false);cin.tie(0);cout.tie(0);
3 #define INF 0x3f3f3f3f
4 using namespace std;
5 typedef long long ll;
6 ll mod = 1000000007;
7 struct Matrix{
8     ll mat[2][2] = {{0}};
9     Matrix operator * (Matrix &inp){
10         Matrix tmp;
11         for(int i = 0; i < 2; i++){
12             for(int j = 0; j < 2; j++){
13                 for(int k = 0; k < 2; k++){
14                     tmp.mat[i][j] = ((tmp.mat[i][j] +
                           (mat[i][k] % mod) *
                           (inp.mat[k][j] % mod)) % mod)
                           % mod;
15                 }
16             }
17         }
18         return tmp;
19     }
20 };
21 Matrix base;
22 Matrix fast_pow(int n){
23     if(n == 1) return base;
24     if(n % 2 == 0){

```



```

25     Matrix res = fast_pow(n >> 1);
26     return res * res;
27 }
28 Matrix res = fast_pow(n >> 1);
29 return base * res * res;
30 }
31 int main(){
32     IOS
33     base.mat[0][0] = 1;
34     base.mat[0][1] = 1;
35     base.mat[1][0] = 1;
36     base.mat[1][1] = 0;
37     Matrix output = fast_pow(20);
38     cout << output.mat[0][0] << '\n';
39
40     return 0;
41 }

```

6 常用小扣

6.1 01 背包問題

```

1 // 01 backpack
2 for(int i=1;i<=n;i++){
3     for(int j=x;j>=w[i];j--){
4         dp[j]=max(dp[j],dp[j-w[i]]+v[i]);
5     }
6 }

```

6.2 Topological

```

1 //拓鋪
2 int N, M, u, v, deg[MAXN];    // deg[i] 紀錄點 i
    被連入邊數
3 vector<int> edge[MAXN];
4
5 /*----- 1. 計算 indegree -----*/
6 cin >> N >> M;
7 while(M--){
8 {
9     cin >> u >> v;
10    edge[u].push_back(v);
11    ++deg[v];
12 }
13
14 /*----- 2. 將 indegree 為 0 的點放入 queue 中
    -----*/
15 queue<int> q;                // 紀錄待拔點
16 for(int i = 0; i < N; ++i)
17     if(deg[i] == 0)
18         q.push(i);
19
20 /*----- 3. 重複拔點，直到 queue 清空 -----*/
21 while(!q.empty()){
22 {
23     int cur = q.front(); q.pop();
24     cout << cur << "\n";
25     for(int i: edge[cur])
26     {
27         --deg[i];                // 3-1.
            相連點 indegree 減一
28         if(deg[i] == 0) q.push(i);    // 3-2. 若該點
            indegree 減至 0，則放入 queue 中
29     }
30 }

```

6.3 常用

```

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

```

7 Basic

7.1 GCD

```

1 int gcd(int a, int b){
2     if(b == 0) return a;
3     else return gcd(b, a % b);
4 }

```

7.2 LCM

```

1 int lcm(int a, int b){
2     return a * b / gcd(a, b);
3 }

```

7.3 Leap Year

```

1 bool isLeap(int n){
2     if(n % 100 == 0)
3         if(n % 400 == 0) return true;
4         else return false;
5     if(n % 4 == 0) return true;
6     else return false;
7 }

```

8 Trick

8.1 求根號和解決 pow 精度問題

```

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

```

8.2 查找和二分搜

```

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

```

8.3 $a^b \bmod p$

```

1 //  $a^b \bmod p$ 
2
3 long powmod(long base, long exp, long modulus){
4     base %= modulus;
5     long result = 1;
6     while(exp > 0){
7         if(exp & 1) result = (result * base) %
            modulus;
8         base = (base * base) % modulus;
9         exp >> 1;

```

```
10 |     }
11 |     return result;
12 | }
```

8.4 $n! \bmod p$

```
1 // n! mod p
2
3 int factmod(int n,int p){
4     long long res = 1;
5     while(n > 1){
6         res = (res % powmod(p-1 , n/p , p)) % p;
7         for(int i=2;i<=n%p;i++)
8             res = (res * i) % p;
9         n /= p;
10    }
11 }
```

8.5 從 1 到 n 選 M 個數字

```

1 // n >= m number from 1 ~ n
2 void combination(int n, int m){
3     if (n<m) return;
4
5     int a[50] = {0};
6     int k = 0;
7
8     for(int i=1;i<=m;i++) a[i] = i;
9     while(1){
10         for(int i=1;i<=m;i++)
11             cout << a[i] << " ";
12         cout << endl;
13
14         k = m;
15         while((k>0) && (n-a[k] == m-k)) k --;
16         if(k == 0) break;
17         a[k] ++;
18         for(int i=k+1;i<=m;i++){
19             a[i] = a[i-1] + 1;
20         }
21     }
22 }

```

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

```

1 #define MAXN 100 // largest n or m
2 long binomial_coefficient(n, m) // compute n choose m
3 int n, m;{
4     int i, j;
5     long bc[MAXN][MAXN];
6     for(i = 0; i <= n; i++) bc[i][0] = 1;
7     for(j = 0; j <= n; j++) bc[j][j] = 1;
8     for(i = 1; i <= n; i++)
9         for(j = 1; j < i; j++)
10             bc[i][j] = bc[i - 1][j - 1] + bc[i -
11                 1][j];
12     return bc[n][m];
13 }

```

8.7 LICS

```
1 int a[100] = {0};
2 int b[100] = {0};
3 int f[100] = {0};
4 int n = 0, m = 0;
5 int main(void){
6     cin >> n;
7     for(int i = 1; i <= n; i++) cin >> a[i];
```

```

8      cin >> m;
9      for(int i = 1; i <= m; i++) cin >> b[i];
10     for(int i = 1; i <= n; i++){
11         int k = 0;
12         for(int j = 1; j <= m; j++){
13             if(a[i] > b[j] && f[j] > k) k = f[j];
14             else if(a[i] == b[j] && k + 1 > f[j])
15                 f[j] = k + 1;
16         }
17     }

```

8.8 m-ary to 10-ary

```
1 string num = "0123456789ABCDE";
2
3 int mToTen(string n, int m){
4     int multi = 1;
5     int result = 0;
6
7     for(int i = n.size() - 1; i >= 0; i--){
8         result += num.find(n[i]) * multi;
9         multi *= m;
10    }
11    return result;
12 }
```

9 Math

9.1 ModInverse

```

1 // 求 a 在模 m 下的模反元素, m 必須是質數
2 ll modInverse(ll a, ll m) {
3     ll res = 1, exp = m - 2;
4     while (exp > 0) {
5         if (exp % 2 == 1) res = (res * a) % m;
6         a = (a * a) % m;
7         exp /= 2;
8     }
9     return res;
10 }

```

9.2 CRT

```

1 ll extendedGCD(ll a, ll b, ll &x, ll &y){
2     if(b == 0){
3         x = 1, y = 0;
4         return a;
5     }
6     ll x1, y1;
7     ll g = extendedGCD(b, a % b, x1, y1);
8     x = y1;
9     y = x1 - (a / b) * y1;
10    return g;
11 }
12
13 ll chineseRemainder(vector<ll> &n, vector<ll> &a){
14     ll prod = 1;
15     for(ll num : n) prod *= num; // Calculate the product of all moduli
16
17     ll result = 0;
18     for(int i = 0; i < n.size(); i++){
19         ll p = prod / n[i]; // Product divided by current modulus
20         ll x, y;
21         extendedGCD(p, n[i], x, y); // Solve p * x ≡ 1 (mod n[i])
22         result = (result + a[i] * x * p) % prod; // Add current term to result

```

```
23 |     }  
24 |     return (result + prod) % prod; // Return the  
    |         result modulo the product  
25 | }
```

9.3 LinearCongruence

```
1 | ll solveLinearCongruence(ll a, ll b, ll m){  
2 |     ll x, y;  
3 |     ll g = extendedGCD(a, m, x, y); // Solve a * x +  
    |         m * y = gcd(a, m)  
4 |     if (b % g != 0) return -1; // No solution if  
    |         gcd(a, m) does not divide b  
5 |  
6 |     x = (x * (b / g)) % m;  
7 |     if (x < 0) x += m; // Ensure the result is  
    |         positive  
8 |     return x;  
9 | }
```