## 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

```
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

```
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

```
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

```
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

```
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

```
#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];
```

```cpp
  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

```cpp
#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 Kurskal's Algorithm

```cpp
#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;
}
```

## 3.6 Prim's Algorithm

```cpp
#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];
```

```cpp
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;
}
```

### 3.7 LCA

```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;
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;
}
```

### 3.8 Topological Sort

```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;
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

```cpp
#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 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;
}
```

## 4.3 Sparse Table

```
#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 = 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]);
```

```cpp
}
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;
}
```