

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	1
2.1	質數篩 CPP	1
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	
3.1	Floyd Warshall	
3.2	Bellman Ford	
3.3	SPFA	
3.4	Dijkstra	
3.5	Kruskal's Algorithm	
3.6	Prim's Algorithm	
3.7	LCA	
3.8	Topological Sort	

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    }
19    cout << dp[l1][l2] << '\n';
20    return 0;
21 }
```

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 }
```

```

20 for(int i = 0; i < n; i++) ans = max(ans, dp[i]);
21 cout << ans << '\n';
22
23 return 0;
24 }
```

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    }
14 }
```

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 #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 bitset<MAXN> prime_bool;
6 vector<ll> prime;
7 void find_prime(){
8     prime_bool.set();
9     for(int i=2;i<MAXN;i++){
10        if(prime_bool[i]){
11            prime.push_back(i);
12        }
13        for(auto j:prime){
14            if(j*i>=MAXN)
15                break;
16            prime_bool[j*i]=0;
17            if(i%j==0)
18                break;
19        }
20    }
21 }
```

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     """x % y"""
5     if a == 0:
6         return (b, 0, 1)
7     else:
8         b_div_a, b_mod_a = divmod(b, a)
9         g, x, y = egcd(b_mod_a, a)
10        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 cin >> l >> r;
20 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;
45 }

```

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

```

```

51 solve();
52
53 return 0;
54 }

```

3.5 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 };
12 int query(int a){
13     if(parent[a] == -1) return a;
14     return parent[a] = query(parent[a]);
15 }
16 bool merge(int a, int b){
17     int r1 = query(a);
18     int r2 = query(b);
19     if(r1 == r2) return false;
20     if(parent[r1] < parent[r2]) parent[r2] = r1;
21     else parent[r1] = r2;
22     return true;
23 }
24 int main(){
25     IOS
26     int n, m;
27     memset(parent, -1, sizeof(parent));
28     cin >> n >> m;
29     vector<Edge> adj;
30     for(int i = 0; i < m; i++){
31         int u, v, w;
32         cin >> u >> v >> w;
33         adj.push_back({u, v, w});
34     }
35     sort(adj.begin(), adj.end());
36     // for(int i = 0; i < m; i++) cout << adj[i].w << '
37     ';
38     int cost = 0, n_edge = 0;
39     for(Edge e : adj){
40         if(merge(e.u, e.v)){
41             cost += e.w;
42             n_edge++;
43         }
44     }
45     if(n_edge == n - 1) cout << cost << '\n';
46     else cout << -1 << '\n';
47     return 0;
48 }

```

3.6 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[100005];
11 int main(){
12     IOS
13     // freopen("input.in", "r", stdin);
14     cin >> n >> m;

```

```

14  memset(dis, 0x3f3f3f3f, sizeof(dis));
15  memset(parent, -1, sizeof(parent));
16  for(int i = 0; i < m; i++){
17      int u, v, w;
18      cin >> u >> v >> w;
19      adj[u].push_back({v, w});
20      adj[v].push_back({u, w});
21  }
22  int start = 0;
23  dis[start] = 0;
24  priority_queue<pair<int, int>, vector<pair<int,
    int>, greater<pair<int, int> > > pq;
25  pq.push({dis[start], start});
26  while(!pq.empty()){
27      pair<int, int> cur = pq.top();
28      pq.pop();
29      if(visited[cur.second]) continue;
30      visited[cur.second] = true;
31      for(auto i : adj[cur.second]){
32          if(visited[i.first]) continue;
33          if(dis[i.first] > i.second){
34              dis[i.first] = i.second;
35              parent[i.first] = cur.second;
36              pq.push({dis[i.first], i.first});
37          }
38      }
39  }
40  int cost = 0, err = 0;
41  for(int i = 0; i < n; i++){
42      if(dis[i] < 0x3f3f3f3f) cost += dis[i];
43      else err++;
44  }
45  cout << (err ? -1 : cost) << "\n";
46  // for(int i = 0; i < n; i++) cout << dis[i] << ' ';
47
48  return 0;
49 }

```

3.7 LCA

```

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 = 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
49     return 0;
50 }

```

3.8 Topological Sort

```

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  vector<int> vec[200005];
6  int ind[100005];
7  int main(){
8     IOS
9     int n, m;
10    cin >> n >> m;
11    memset(ind, 0, sizeof(ind));
12    for(int i = 0; i < m; i++){
13        int a, b;
14        cin >> a >> b;
15        ind[b]++;
16        vec[a].push_back(b);
17    }
18    queue<int> q;
19    for(int i = 1; i <= n; i++){
20        if(ind[i] == 0) q.push(i);
21    }
22    vector<int> top;
23    while(!q.empty()){
24        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 }

```