Team Note of SWJY

Compiled on October 7, 2022

Contents

1 Graph

1.1 Dijkstra

```
//백준 1753<최단거리> 소스코드
#define X first
#define Y second

priority_queue<pair<int, int>, vector<pair<int, int>>, greater<pair<int, int>>> pq;
vector<pair<int, int>> stage[20005];
int dist[20005];
int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int v, e, k;
    fill(dist, dist+20005, 1e9+10);
    cin >> v >> e >> k;

    while(e--)
    {
        int a, b, cost;
        cin >> a >> b >> cost;
        stage[a].push_back({cost, b});
    }

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

    while(!pq.empty())
    {
        int cost, cur;
        tie(cost, cur) = pq.top(); pq.pop();

        if(dist[cur] != cost)
            continue;
        for(auto nxt : v[cur])
        {
            if(dist[nxt.Y] > dist[cur] + nxt.X)
            {
                dist[nxt.Y] = dist[cur] + nxt.X;
                pq.push({dist[nxt.Y], nxt.Y});
            }
        }
    }
```

```
    for(int i = 1; i <= v; i++)
    {
        if(dist[i] == 1e9+10)
            cout << "INF\n";
        else
            cout << dist[i] << "\n";
    }
}
```

1.2 Dijkstra BackTracking

```
int temp = to;
while(temp != from)
{
    result_route.push_back(temp);
    temp = route[temp];
}
result_route.push_back(from);
cout << result_route.size() << "\n";
reverse(result_route.begin(), result_route.end());

for(auto x : result_route)
    cout << x << " ";
cout << "\n";
```

1.3 Union-Find (Disjoint-set)

```
int find(int x)
{
    if(parent[x] < 0)
        return x;
    parent[x] = find(parent[x]);
    return parent[x];
}

void comb(int a, int b)
{
    a = find(a);
    b = find(b);

    if(a == b)
        return;

    if(parent[a] > parent[b])
        swap(a, b);
    parent[a] += parent[b];
    parent[b] = a;
    return;
}
```

1.4 MST (Kruskal ver.)

```
//NOTICE! using Union-Find
int find(int x)
{
    if(parent[x] < 0)
        return x;
    return parent[x] = find(parent[x]);
}

bool is_diff_group(int a, int b)
{
    a = find(a);
    b = find(b);
    if(a == b)
        return 0;
    if(parent[a] > parent[b])
        swap(a, b);

    parent[a] += parent[b];
    parent[b] = a;
    return 1;
}

sort(stage, stage+e);
tuple<int, int, int> stage[];
for(int i = 0; i < e; i++)
{
    int a, b, cost;
    tie(cost, a, b) = stage[i];
    if(!is_diff_group(a, b))
        continue;
    result += cost;
    cnt++;
    if(cnt == v-1)
        break;
}
cout << reuslt << "\n";
```

1.5 Bellman-Ford

```
//백준 11657<타임머신> 소스코드
#define INF 1e+18
#define ll long long
vector<pair<int, ll>> stage[501];
ll cost[501];

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int n, m;
    cin >> n >> m;

    while(m--)
    {
        int a, b; ll c;
        cin >> a >> b >> c;

        stage[a].push_back({b, c});
    }

    fill(cost, cost+n+1, INF);
    cost[1] = 0;
    bool flag = 0;

    for(int k = 0; k < n; k++)
        for(int i = 1; i <= n; i++)
```

```
            for(auto nxt : stage[i])
                if(cost[i] != INF && cost[nxt.first] > cost[i] + nxt.second)
                {
                    cost[nxt.first] = cost[i] + nxt.second;
                    if(k == n-1)
                        flag = 1;   //Is it cycle?
                }

    if(flag == 1)
        cout << "-1\n";
    else
        for(int i = 2; i <= n; i++)
            cout << (cost[i] != INF ? cost[i] : -1) << "\n";
}
```

1.6 SCC (Strongly Connected Component)

```
\\백준 2150<Strongly Connected Component> 소스코드
int cnt = 1, scc_num;
vector<vector<int>> scc_result;
vector<int> stage[10005];
vector<bool> finished(10005, 0);
int dfsn[10005];
stack<int> s;

int dfs(int node)
{
    dfsn[node] = cnt++;
    s.push(node);

    int result = dfsn[node];
    for(int nxt : stage[node])
    {
        if(dfsn[nxt] == 0)
            result = min(result, dfs(nxt));
        else if(!finished[nxt])
            result = min(result, dfsn[nxt]);
    }

    if(result == dfsn[node])
    {
        vector<int> cur_scc;
        while(1)
        {
            int k = s.top();
            finished[k] = true;
            cur_scc.push_back(k);
            s.pop();

            if(k == node)
                break;
        }

        sort(cur_scc.begin(), cur_scc.end());
        scc_result.push_back(cur_scc);
        scc_num++;
    }
    return result;
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    int v, e;
    cin >> v >> e;

    while(e--)
    {
        int a, b;
        cin >> a >> b;

        stage[a].push_back(b);
    }

    for(int i = 1; i <= v; i++)
        if(dfsn[i] == 0)
            dfs(i);

    cout << scc_num << "\n";

    sort(scc_result.begin(), scc_result.end());
    for(auto& party : scc_result)
    {
        for(int x : party)
            cout << x << " ";
        cout << "-1\n";
    }
}
```

2 Segment Tree

2.1 Original Segment Tree (Sum ver.)

```
#define ll long long
void init(vector<ll>& a, vector<ll>& tree, int node, int start, int end)
{
    if(start == end)
        tree[node] = a[start];
    else
    {
        init(a, tree, node*2, start, (start+end)/2);
        init(a, tree, node*2+1, (start+end)/2+1, end);
        tree[node] = tree[node*2] + tree[node*2+1];
    }
}

ll query(vector<ll>& tree, int node, int start, int end, int left ,int right)
{
    if(left > end || right < start)
        return 0;
    if(left <= start && end <= right)
        return tree[node];
```

```
    ll lsum = query(tree, node*2, start, (start+end)/2, left, right);
    ll rsum = query(tree, node*2+1, (start+end)/2+1, end, left, right);
    return lsum + rsum;
}

void update(vector<ll>& arr, vector<ll>& tree, int node, int start, int end, int index, ll
val)
{
    if(index < start || index > end)
        return;
    if(start == end)
    {
        a[index] = val;
        tree[node] = val;
        return;
    }

    update(arr, tree, node*2, start, (start+end)/2, index, val);
    update(arr, tree, node*2+1, (start+end)/2+1, end, index, val);
    tree[node] = tree[node*2] + tree[node*2+1];
}
```

2.2 Lazy Segment Tree (Sum ver.)

```
#include <cmath>
#define ll long long
void init(vector<ll>& a, vector<ll>& tree, int node, int start, int end)
{
    if(start == end)
        tree[node] = a[start];
    else
    {
        init(a, tree, node*2, start, (start+end)/2);
        init(a, tree, node*2+1, (start+end)/2+1, end);
        tree[node] = tree[node*2] + tree[node*2+1];
    }
}

void update_lazy(vector<ll>& tree, vector<ll>& lazy, int node, int start, int end)
{
    if(lazy[node] != 0)
    {
        tree[node] += (end-start+1) * lazy[node];
        if(start != end)
        {
            lazy[node*2] += lazy[node];
            lazy[node*2+1] += lazy[node];
        }
        lazy[node] = 0;
    }
}

void update_range(vector<ll>& tree, vector<ll>& lazy, int node, int start, int end, int
left, int right, ll diff)
{
    update_lazy(tree, lazy, node, start, end);
    if(left > end || right < start)
        return;
    if(left <= start && end <= right)
    {
        tree[node] += (end-start+1) * diff;
        if(start != end)
        {
            lazy[node*2] += diff;
            lazy[node*2+1] += diff;
        }
        return;
    }

    update_range(tree, lazy, node*2, start, (start+end)/2, left, right, diff);
    update_range(tree, lazy, node*2+1, (start+end)/2+1, left, right, diff);
    tree[node] = tree[node*2] + tree[node*2+1];
}

ll query(vector<ll>& tree, vector<ll>& lazy, int node, int start, int end, int left, int
right)
{
    update_lazy(tree, lazy, node, start, end);
    if(left > end || right < start)
        return 0;
    if(left <= start && end <= right)
        return tree[node];

    ll lsum = query(tree, lazy, node*2, start, (start+end)/2, left, right);
    ll rsum = query(tree, lazy, node*2+1, (start+end)/2+1, end, left, right);
    return lsum + rsum;
}
```

2.3 Finding k-th number with Segment Tree

```
#define ll long long
//init -> original segment tree
//update -> original segment tree

ll query(vector<ll>& tree, int node, int start, int end, int q)
{
    tree[node]--;
    if(start == end)
        return start;
    if(q <= tree[node*2])
        return query(tree, node*2, start, (start+end)/2, q);
    else
        return query(tree, node*2+1, (start+end/2+1, end, q - tree[node*2]);
}
```

3 Strings

3.1 KMP

```
vector<int> failure(string& s)
{
    vector<int> f(s.size());
    int j = 0;
    for(int i = 1; i < s.size(); i++)
    {
        while(j > 0 && s[i] != s[j])
            j = f[j-1];
        if(s[i] == s[j])
```

```
            f[i] = ++j;
    }

    return f;
}

string s, p;
vector<int> f = failure(p);
int j = 0;
for(int i = 0; i < s.size(); i++)
{
    while(j > 0 && s[i] != p[j])
        j = f[j-1];
    if(s[i] == p[j])
        j++;
    if(j == p.size())
    {
        cout << 1 << "\n";
        return 0;
    }
}
cout << 0 << "\n";
```

4 DP

4.1 LIS(O(N^2))

```
int n;
cin >> n;
for(int i = 0; i < n; i++)
    cin >> arr[i];

dp[0] = 1;
for(int i = 1; i < n; i++)
{
    for(int j = i-1; j >= 0; j--)
        if(arr[j] < arr[i] && dp[i] < dp[j] + 1)
            dp[i] = dp[j] + 1;
}
```

4.2 LIS(O(NlogN))

```
int n;
cin >> n;
for(int i = 0; i < n; i++)
    cin >> arr[i];

vector<int> result;
result.push_back(0); //NOTICE!!
for(int i = 0; i < n; i++)
{
    int k = lower_bound(result.begin(), result.end(), arr[i]) - result.begin();
    if(k == result.size() + 1)
        result.push_back(arr[i]);
    else
        result[k] = arr[i];
}

cout << result.size() - 1 << "\n";
```

4.3 LCS

```
string s1, s2;
int result = 0;
for(int i = 0; i < s2.size(); i++)
{
    for(int j = 0; j < s1.size(); j++)
    {
        if(s2[i] == s1[j])
            dp[i+1][j+1] = dp[i][j] + 1;
        else
            dp[i+1][j+1] = max(dp[i][j+1], dp[i+1][j]);
        result = max(result, dp[i+1][j+1]);
    }
}
cout << result << "\n";
```

4.4 LCS Backtracking

```
//First, LCS algorithm needed
string s; //result string
int x = b.size();
int y = a.size();

while(dp[x][y] > 0)
{
    if(dp[x][y] == dp[x-1][y])
        x--;
    else if(dp[x][y] == dp[x][y-1])
        y--;
    else if(dp[x][y] - 1 == dp[x-1][y-1])
    {
        s.push_back(a[y-1]);
        x--;
        y--;
    }
}

for(int i = s.size()-1; i >= 0; i--)
    cout << s[i];
```

4.5 Knapsack

```
int n, k;
cin >> n >> k;
for(int i = 1; i <= n; i++)
    cin >> w[i] >> v[i];

for(int i = 1; i <= n; i++)
{
    for(int j = 1; j <= k; j++)
    {
        if(w[i] > j)
            dp[i][j] = dp[i-1][j];
        else
            dp[i][j] = max(dp[i-1][j], dp[i-1][j-w[i]] + v[i]);
    }
}
cout << dp[n][k] << "\n";
```

5 Extra

5.1 Sparse Table

```
//백준 17435<합성함수와 쿼리> 소스코드
//next[i][j] = 정점 i에서 2^j번 이동한 후의 정점
//initiating
for(int i = 1; i <= M; i++)
    cin >> next[i][0];

//i에서 2^(j+1)번 이동한 후의 정점은 i에서 2^j번*2번 이동하는 것
//next[i][j+1] = next[ next[i][j] ][j]
for(int j = 1; j < MAX_D; j++)
    for(int i = 1; i <= M; i++)
        next[i][j] = next[ next[i][j-1] ][j-1];

//processing query
while(q--)
{
    int n, x;
    cin >> n >> x;

    for(int j = MAX_D - 1; j >= 0; j--)
    {
        if(n >= (1<<j))
        {
            n -= (1<<j);
            x = next[x][j];
        }
    }

    cout << x << "\n";
}
```

5.2 Prefix Sum(Two-dimensinal)

```
int n, m;
cin >> n >> m;

for(int i = 1; i <= n; i++)
    for(int j = 1; j <= n; j++)
        cin >> stage[i][j];

for(int i = 1; i <= n; i++)
    for(int j = 1; j <= n; j++)
        dp[i][j] = dp[i-1][j] + dp[i][j-1] - dp[i-1][j-1] + stage[i][j];

while(m--)
{
    int x1, y1, x2, y2;
    cin >> x1 >> y1 >> x2 >> y2;

    cout << dp[x2][y2] - dp[x2][y1-1] - dp[x1-1][y2] + dp[x1-1][y1-1] << "\n";
}
```

5.3 Matrix Pow

```
//NOTICE! Max_size == 5
//NOTICE! mod == 1000

int n;
int stage[5][5];
int temp[5][5];
int result[5][5];

void multiple(int arr[5][5], int brr[5][5])
{
    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
            temp[i][j] = 0;

    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
        {
            for(int p = 0; p < n; p++)
                temp[i][j] += (arr[i][p]*brr[p][j]) % 1000;
            temp[i][j] %= 1000;
        }

    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
            result[i][j] = temp[i][j];
}

void solve(int arr[5][5], long long k)
{
    if(k == 1)
    {
        multiple(result, arr);
        return;
    }

    solve(arr, k/2);
    multiple(result, result);
    if(k % 2 == 0)
        return;
    else
        multiple(result, arr);
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    long long b;
    cin >> n >> b;

    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
        {
            cin >> stage[i][j];
            if(i == j)
                result[i][j] = 1;
        }

    solve(stage, b);
```

```
    for(int i = 0; i < n; i++)
    {
        for(int j = 0; j < n; j++)
            cout << result[i][j] << " ";
        cout << "\n";
    }
}
```

6 Tips/Tools

6.1 Time Complexity

```
1. Binary Search Tree(set, map)
insert, erase, find, update -> O(lg N)
2. Priority Queue
insert, erase, push -> O(lg N) (faster than set/map)
3. MST(Kruskal ver.) -> O(ElgE)
4. Floyd - O(V^3)
5. Dijkstra - O(ElgE) or O(ElgV)
6. KMP - O(s1.size() + s2.size())
```

6.2 Bitmasking

```
1. k번 비트가 1인지 0인지 확인 - status & (1<<k) == (1<<k)
2. k번 비트를 1로 만들기 - status |= (1<<k)
3. k번 비트를 0으로 만들기 - status &= ~(1<<k)
4. 모든 비트가 0인지 확인 - status == 0
5. 모든 비트가 1인지 확인 - status == (1<<n)-1

ex) 백준 2098<외판원 순회> 소스코드
```

```
#define INF 1e9+10

int n;
int stage[20][20];
int dp[20][(1<<16)+1];

int tsp(int cur, int status)
{
    int& ret = dp[cur][status];
    if(ret != -1)
        return ret;

    if(status == (1 << n)-1)
    {
        if(stage[cur][0] != 0)
            return stage[cur][0];
        return INF;
    }

    ret = INF;
    for(int i = 0; i < n; i++)
    {
        if(status & (1 << i) || stage[cur][i] == 0)
            continue;
        ret = min(ret, tsp(i, status | (1 << i)) + stage[cur][i]);
    }

    return ret;
}

int main()
{
    ios::sync_with_stdio(0);
    cin.tie(0);

    cin >> n;

    for(int i = 0; i < n; i++)
        for(int j = 0; j < n; j++)
            cin >> stage[i][j];

    memset(dp, -1, sizeof(dp));

    cout << tsp(0, 1) << "\n";
}
```

6.3 Memo

```
<bits/stdc++.h> 대체 헤더파일들
<iostream>
<string>
<algorithm>
<cmath>
<vector>
<string>
<set>
<stack>
<queue>
<sstream>
<iomanip>
<map>

#둘의 차이라고 한다면 getline(cin, str);는 string형의 객체인 str에 입력을 받는 것이고
cin.getline(input, 101);은 char형 배열인 input에 최대 101자를 입력받는다는 것 뿐입니다.
#sort -> a 뒤 b
#gcd -> (a < b)에서 (remain == 0 종료 / a = remain, b = a)
#전위순회(preorder) - 루트/왼/오
#중위순회(inorder) - 왼/루트/오
#후위순회(postorder) - 왼/오/루트
#(피보나치) 피사노주기 - 주기는 M = 10^k (k>2)일 때 15*10^(k-1)
```