# Team Notebook

March 28, 2025

## Contents

# 1 estm. efficienty

## 1.1 input-time

| input size | required time complexity |
|---|---|
| n <= 10 | O(n!) |
| n <= 20 | O(2^n) |
| n <= 500 | O( n ) |
| n <= 5000 | O( n ) |
| n <= 10 | O(n log n) or O(n) |
| n is large | O(1) or O(log n) |

# 2 graph

## 2.1 bfs

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

using namespace std;

vector<vector<int>> G;
vector<int> visited;
vector<int> teams;

int bfs(int i) {
    queue<int> q;
    q.push(i);

    teams[i] = 1;

    while (!q.empty()) {
        int c = q.front();
        q.pop();

        if (visited[c])
            continue;

        visited[c] = 1;

        for (int p : G[c]) {
            if (visited[p])
                continue;

            if (teams[p] == teams[c]) {
                cout << "IMPOSSIBLE" << "\n";
                return 0;
            }
            teams[p] = (teams[c] == 1 ? 2 : 1);
            q.push(p);
        }
    }

    return 1;
}

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

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

    G = vector<vector<int>>(n + 1, vector<int>());
    teams = vector<int>(n + 1, 0);
    visited = vector<int>(n + 1, 0);

    int a, b;
    for (int i = 0; i < m; i++) {
        cin >> a >> b;
        G[a].push_back(b);
        G[b].push_back(a);
    }

    for (int i = 1; i <= n; i++) {
        if (teams[i] != 0)
            continue;

        if (!bfs(i))
            return 0;
    }

    for (int i = 1; i <= n; i++)
        cout << teams[i] << ' ';
    cout << '\n';

    return 0;
}
```

## 2.2 bfsGrid

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

using namespace std;

// BFS COM PREVIOUS DO LABYRINTH CSES

vector<vector<char>> G;
vector<vector<int>> visited;
vector<pair<int, int>> moves = {{-1, 0}, {1, 0}, {0, -1},
    {0, 1}};
vector<vector<pair<int, int>>> previous;

void bfs(int i, int j) {
    queue<pair<int, int>> q;
    q.push({i, j});

    while (!q.empty()) {
        pair<int, int> p = q.front();
        q.pop();

        if (visited[p.first][p.second])
            continue;

        if (G[p.first][p.second] == 'B') {
            return;
        }

        visited[p.first][p.second] = 1;

        for (pair<int, int> pp : moves) {
            int I = p.first + pp.first, J = p.second + pp.
                second;

            if (I < 0 || J < 0 || I >= G.size() || J >= G[0].
                size() || visited[I][J] || G[I][J] == '#')
                continue;

            previous[I][J] = {p.first, p.second};
            q.push({I, J});
        }
    }
}

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

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

    G = vector<vector<char>>(n, vector<char>(m));
    visited = vector<vector<int>>(n, vector<int>(m, 0));
    previous = vector<vector<pair<int, int>>>(n, vector<pair<
        int, int>>(m));
```

```
int beginI = 0, beginJ = 0, finalI = 0, finalJ = 0;

for (int i = 0; i < n; i++) {
    for (int j = 0; j < m; j++) {
        cin >> G[i][j];
        if (G[i][j] == 'A') {
            beginI = i;
            beginJ = j;
        }

        if (G[i][j] == 'B') {
            finalI = i;
            finalJ = j;
        }
    }
}

previous[finalI][finalJ] = {INT_MIN, INT_MAX};

bfs(beginI, beginJ);

if (previous[finalI][finalJ].first == INT_MIN && previous
     [finalI][finalJ].second == INT_MAX) {
    cout << "NO" << "\n";
    return 0;
}

cout << "YES\n";

stack<char> ans;

while (finalI != beginI || finalJ != beginJ) {
    pair<int, int> &pre = previous[finalI][finalJ];

    if (finalI < pre.first)
        ans.push('U');
    else if (finalI > pre.first)
        ans.push('D');
    else if (finalJ < pre.second)
        ans.push('L');
    else
        ans.push('R');

    finalI = pre.first;
    finalJ = pre.second;
}

cout << ans.size() << '\n';

while (!ans.empty()) {
        cout << ans.top();
        ans.pop();
    }

    return 0;
}
```

## 2.3 dfs

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

using namespace std;

vector<vector<int>> G;
vector<int> visited;

void dfs(int city) {
    stack<int> st;
    st.push(city);

    while (!st.empty()) {
        int c = st.top();
        st.pop();

        if (visited[c])
            continue;

        visited[c] = 1;

        for (int p : G[c]) {
            if (!visited[p])
                st.push(p);
        }
    }
}

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

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

    G = vector<vector<int>>(n + 1, vector<int>());

    int a, b;
    for (int i = 0; i < m; i++) {
        cin >> a >> b;
        G[a].push_back(b);
        G[b].push_back(a);
    }

    int total = 0;

    string ans;
    visited = vector<int>(n + 1, 0);

    return 0;
}
```

## 2.4 dfsGrid

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

using namespace std;

vector<vector<char>> G;
vector<vector<int>> visited;
vector<pair<int, int>> moves = {{-1, 0}, {0, -1}, {0, 1},
     {1, 0}};

void dfs(int i, int j) {
    if (visited[i][j])
        return;

    visited[i][j] = 1;

    for (pair<int, int> p : moves) {
        int I = i + p.first, J = j + p.second;

        if (I < 0 || J < 0 || I >= G.size() || J >= G[0].size
             ())
            continue;

        dfs(I, J);
    }
}

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

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

    G = vector<vector<char>>(n, vector<char>(m));
    visited =vector<vector<int>>(n, vector<int>(m, 0));
```

```
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            cin >> G[i][j];
        }
    }

    return 0;
}
```

## 2.5 djikstra

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

using namespace std;

vector<vector<pair<int, int>>> G;
vector<int> visited;
vector<int> dist;

void dijkstra(int i) {
    priority_queue<pair<int, int>> q;
    dist[i] = 0;
    q.push({0, i});

    while (!q.empty()) {
        int a = q.top().second;
        q.pop();

        if (visited[a])
            continue;

        visited[a] = 1;

        for (pair<int, int> p : G[a]) {
            int b = p.first, w = p.second;
            if (dist[a] + w < dist[b]) {
                dist[b] = dist[a] + w;
                q.push({ -dist[b], b });
            }
        }
    }
}

int main() {
    ios::sync_with_stdio(0);
    cin.tie(0);
    cout << fixed << setprecision(0); // remove when working
         with floating point

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

    G = vector<vector<pair<int, int>>> (n + 2, vector<pair<
        int ,int>>());
    visited = vector<int> (n + 2, 0);
    dist = vector<int> (n + 2, INFINITY);

    int s, t, b;
    while (m--) {
        cin >> s >> t >> b;
        G[s].push_back({t, b});
        G[t].push_back({s, b});
    }

    dijkstra(0);

    cout << dist[n + 1] << '\n';

    return 0;
}
```

## 2.6 topologicalSort

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

using namespace std;

// TOPOLOGICAL SORT FROM COURSE SCHEDULE (CSES) USING DFS
// IT ALREADY CHECKS IF THERE IS A CYCLE

vector<vector<int>> G;
vector<int> visited;
vector<int> visitedPath;
stack<int> topological;

void dfs(int i, int path, int &ok) {
    if (visited[i])
        return;

    visited[i] = 1;
    visitedPath[i] = path;

    for (int p : G[i]) {
        if (visited[p] && visitedPath[p]) {
            ok = 0;
            return;
        }
        if (!visited[p]) {
            dfs(p, path, ok);
        }
    }

    visitedPath[i] = 0;
    topological.push(i);
}

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

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

    G = vector<vector<int>>(n + 1, vector<int>());
    visited = vector<int>(n + 1, 0);
    visitedPath = vector<int> (n + 1, 0);

    int a, b;
    for (int i = 0; i < m; i++) {
        cin >> a >> b;
        G[a].push_back(b);
    }

    for (int i = 1; i <= n; i++) {
        if (visited[i])
            continue;

        int ok = 1;
        dfs(i, i, ok);

        if (!ok) {
            cout << "IMPOSSIBLE\n";
            return 0;
        }
    }

    while (!topological.empty()) {
        int c = topological.top();
        topological.pop();

        cout << c << ' ';
    }

    return 0;
}
```

# 3 math

## 3.1 divisors

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

using namespace std;

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

   long long n;
   cin >> n;

   for (int i = 1; i <= sqrt(n); i++) {
       if (n % i == 0) {
           if (n / i == i) {
               cout << i << '\n';
           } else {
               cout << i << " " << n / i << '\n';
           }
       }
   }

   return 0;
}
```

## 3.2 gcd

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

using namespace std;

int GCD(int a, int b) {
    if (a == 0)
        return b;

    return GCD(b%a, a);
}
```

## 3.3 $\log_{at}o_b ase_b$

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

using namespace std;

long long log_a_to_base_b(long long a, long long b) {
    return log2(a) / log2(b);
}
```

## 3.4 sieve

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

using namespace std;

vector<bool> primes;

void sieve(long long n) {
    primes = vector<bool>(n + 1, true);
    primes[0] = primes[1] = false;

    for (int i = 2; i <= sqrt(n); i++) {
        if (primes[i]) {
            for (int j = i * i; j <= n; j += i) {
                primes[j] = false;
            }
        }
    }
}

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

    sieve(100);

    return 0;
}
```

# 4 vector

## 4.1 binarySearch

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

using namespace std;

int binarySeach(vector<int> &arr, int val) {
    int l = 0, r = arr.size() - 1;

    while (l <= r) {
        int middle = (l + r) / 2;

        if (arr[middle] == val)
            return middle;

        if (arr[middle] < val)
            l = middle + 1;
        else
            r = middle - 1;
    }

    return -1; // elemento nao encontrado
}
```

## 4.2 kadane

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

using namespace std;

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

   vector<int> arr = {3, -2, 1, 5, -4, 7, 9};

   int MAX = arr[0], sum = arr[0];

   for (int i = 1; i < arr.size(); i++) {
       sum = max(arr[i], sum + arr[i]);
       MAX = max(MAX, sum);
   }

   cout << MAX << '\n';

   return 0;
}
```

## 4.3 prefixSum

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

using namespace std;

int main() {
    ios_base::sync_with_stdio(false);
```

```
cin.tie(0);

int n, q;
cin >> n >> q;

vector<long long> arr(n + 1);
arr[0] = 0;
long long sum = 0, temp;

for (int i = 1; i <= n; i++) {
    cin >> temp;
    sum += temp;
    arr[i] = sum;
}

int a, b;
while (q--) {
    cin >> a >> b;
    cout << arr[b] - arr[a - 1] << '\n';
}
```

```
    return 0;
}
```

## 4.4 rangeXorQueries

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

using namespace std;

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

    int n, q;
    cin >> n >> q;

    vector<long long> arr(n + 1);
    long long sum = 0, temp;

    for (int i = 1; i <= n; i++) {
        cin >> temp;
        sum = sum ^ temp;
        arr[i] = sum;
    }

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

        cout << (arr[b] ^ arr[a - 1]) << '\n';
    }

    return 0;
}
```