Team Note of Knu_idle Team

Park Juhyung, Kim Minhwan, Jeong Seunghyeon

Compiled on October 15, 2023

Contents

1 자료구조 **1**
- 1.1 2차원 구간합 . . . 1
- 1.2 에라토스테네스의 체 . . . 1
- 1.3 유니온 파인드 . . . 1
- 1.4 세그먼트 트리 . . . 2
- 1.5 바텀업 레이지 세그먼트 트리 . . . 2

2 DP **3**
- 2.1 LIS . . . 3
- 2.2 LCS . . . 3
- 2.3 냅색 . . . 3
- 2.4 비트마스크 dp . . . 3
- 2.5 TSP . . . 4

3 문자열 알고리즘 **4**
- 3.1 라빈 카프 알고리즘 . . . 4
- 3.2 kmp 알고리즘 . . . 4
- 3.3 매내처 . . . 5

4 그래프 **5**
- 4.1 DFS, BFS . . . 5
- 4.2 다익스트라 우선순위 큐 알고리즘 . . . 5
- 4.3 벨만포드 알고리즘 . . . 6
- 4.4 플로이드 워셜 . . . 6
- 4.5 크루스칼 알고리즘 . . . 6

1 자료구조

1.1 2차원 구간합

Usage:
Time Complexity: 2차원 구간합 구현, O(nm)

```
#include <bits/stdc++.h>
using namespace std;

int main() {
   ios::sync_with_stdio(false);
   cin.tie(NULL); cout.tie(NULL);

   cin >> n >> m;
   for (int i = 1; i <= n; i++) {
      for (int j = 1; j <= m; j++) {
         cin >> arr[i][j];
         dp[i][j] = dp[i - 1][j] + dp[i][j - 1] - dp[i - 1][j
         - 1] + arr[i][j];
      }
   }
   cin >> k;
   for (int i = 0; i < k; i++) {
      int x1, y1, x2, y2;
      cin >> x1 >> y1;
      cin >> x2 >> y2;
      ans = dp[x2][y2] - dp[x1 - 1][y2] - dp[x2][y1 - 1] +
      dp[x1 - 1][y1 - 1];
      cout << ans << "\n";
   }
   return 0;
}
```

1.2 에라토스테네스의 체

Usage:
Time Complexity: 에라토스테네스의 체 구현, 100만 까지, false 인 경우 소수

```
#include <bits/stdc++.h>
using namespace std;

bool sosu[1000007] = {0};

int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    //확실한 생각과 계획 코드
    sosu[1] = true;
    for (int i = 2; i * i <= 1000000; i++){
        for (int j = i + i; j <= 1000000; j += i){
            sosu[j] = true;
        }
    }
    for (int i = 0; i < 100; i++){
        if (!sosu[i]) cout << i << ' ';
    }
    return 0;
}
```

1.3 유니온 파인드

Usage:
Time Complexity: 유니온 파인드 구현, 경로 압축 버전

```
#include <iostream>
#include <vector>
using namespace std;

int parent[1000001];

int Find(int a){
    if (parent[a] == a){
        return a;
    }
    return parent[a] = Find(parent[a]);
}

void Union(int a, int b){
    a = Find(a);
    b = Find(b);
    if (a != b){
        parent[b] = a;
    }
}

bool check(int a, int b){
    a = Find(a);
    b = Find(b);
    if (a != b){
        return false;
    }else{
        return true;
    }
}

int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    for (int i = 0; i < 1000001; i++){
        parent[i] = i;
    }
    int n,m;
    cin >> n >> m;
    for (int i = 0; i < m; i++){
        int a;
        cin >> a;
        if (a == 0){
```

```
            int b,c;
            cin >> b >> c;
            Union(b,c);
        }else{
            int b,c;
            cin >> b >> c;
            if (check(b,c)){
                cout << "YES" << '\n';
            }else{
                cout << "NO" << '\n';
            }
        }
    }
    return 0;
}
```

1.4 세그먼트 트리

Usage:
Time Complexity: 세그먼트 트리 구현, 구간합 logn 처리

```
#include <iostream>
#include <vector>
using namespace std;
typedef long long int ll;

constexpr int sizen = 100007;

ll segment_tree[4*sizen+1];

ll arr[sizen+1];
ll number = 1;
void init(int n){
    for (int i = 0; i < n; i++){
        segment_tree[n+i] = arr[i];
    }

    for (int i = n-1; i > 0; i--){
        segment_tree[i] = segment_tree[i << 1] +
        segment_tree[i << 1 | 1];
    }
}
void update(int index, ll x, int n){
    segment_tree[n+index] = x;

    for (int i = (n+index)/2; i > 0; i >>= 1){
        segment_tree[i] = segment_tree[i << 1] +
        segment_tree[i << 1 | 1];
    }
}

ll find(int l, int r, int n){
    ll ans_max = 0;
    int templ = l;
    int tempr = r;
    for (l += n, r += n; l != r; l >>= 1, r >>= 1){
        if (l & 1) ans_max += segment_tree[l++];
        if (r & 1) ans_max += segment_tree[--r];
    }
    if (templ % 2 == 1){
        return -ans_max;
    }else{
        return ans_max;
    }
}
void solve(){
    int n,q;
    cin >> n >> q;
    for (int i = 0; i < n; i++){
        cin >> arr[i];
        if (i % 2 == 1){
            arr[i] = -arr[i];
        }
    }
    init(n);
    cout << "#" << number << ' ';
    number++;
    for (int i = 0; i < q; i++){
        int a;
        cin >> a;
        if (a == 0){
            int k,x;
            cin >> k >> x;
            if (k % 2 == 1){
                update(k,-x,n);
            }else{
                update(k,x,n);
            }
        }else if (a == 1){
            int l,r;
            cin >> l >> r;
            cout << find(l,r,n) << ' ';
        }
    }
    cout << '\n';
}
int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    freopen("sample_in.txt","r",stdin);
    int t;
    cin >> t;
    while(t--) solve();
    return 0;
}
```

1.5 바텀업 레이지 세그먼트 트리

Usage:
Time Complexity: $\mathcal{O}(n)$ for constructor, $\mathcal{O}(\log n)$ for query

```
template <class S, S (*op)(S, S), S (*e)(),
  class F, S (*mapping)(F, S), F (*composition)(F, F), F
  (*id)()>
class LazySeg {
  int N, log;
  vector<S> d;
  vector<F> lz;

  void pull(int k) { d[k] = op(d[2 * k], d[2 * k + 1]); }
  void put(int k, F f) {
    d[k] = mapping(f, d[k]);
    if (k < N) lz[k] = composition(f, lz[k]);
  }
  void push(int k) {
    put(2 * k, lz[k]);
    put(2 * k + 1, lz[k]);
    lz[k] = id();
  }
public:
  LazySeg() : LazySeg(0) {}
  explicit LazySeg(int n) : LazySeg(vector<S>(n, e())) {}
  explicit LazySeg(const vector<S> &v) {
    log = 31 - __builtin_clz(v.size() | 1);
    N = 1 << log;
    d = vector<S>(2 * N, e());
    lz = vector<F>(N, id());
    for (int i = 0; i < (int)v.size(); i++) d[N + i] = v[i];
    for (int i = N - 1; i >= 1; i--) pull(i);
  }
  void set(int p, S x) {
    p += N;
    for (int i = log; i >= 1; i--) push(p >> i);
    d[p] = x;
    for (int i = 1; i <= log; i++) pull(p >> i);
  }
  S prod(int l, int r) {
    if (l == r) return e();
    l += N, r += N;
    for (int i = log; i >= 1; i--) {
      if (((l >> i) << i) != l) push(l >> i);
      if (((r >> i) << i) != r) push((r - 1) >> i);
    }
    S sml = e(), smr = e();
    while (l < r) {
      if (l & 1) sml = op(sml, d[l++]);
      if (r & 1) smr = op(d[--r], smr);
      l >>= 1, r >>= 1;
    }
    return op(sml, smr);
  }
```

```
  S all_prod() { return d[1]; }
  void apply(int l, int r, F f) {
    if (l == r) return;
    l += N, r += N;
    for (int i = log; i >= 1; i--) {
      if (((l >> i) << i) != l) push(l >> i);
      if (((r >> i) << i) != r) push((r - 1) >> i);
    }
    int l2 = l, r2 = r;
    while (l < r) {
      if (l & 1) put(l++, f);
      if (r & 1) put(--r, f);
      l >>= 1, r >>= 1;
    }
    l = l2, r = r2;
    for (int i = 1; i <= log; i++) {
      if (((l >> i) << i) != l) pull(l >> i);
      if (((r >> i) << i) != r) pull((r - 1) >> i);
    }
  }
};
```

2 DP

2.1 LIS

Usage:
Time Complexity: LIS 구현, 간단한 버전

```
#include <stdio.h>

int a[1001];
int d[1001] = {0};

int main(){
    int n;
    int max = 0;
    scanf("%d",&n);
    for (int i = 0; i < n; i++){
        scanf("%d",&a[i]);
    }
    for (int i = 0; i < n; i++){
        d[i] = 1;
        for (int j = 0; j < i; j++){
            if (a[i] > a[j] && d[i] < d[j] + 1){
                d[i] = d[j] + 1;
            }
        }
    }
    for (int i = 0; i < n; i++){
        if (d[i] > max){
            max = d[i];
        }
    }
    printf("%d\n",max);

    return 0;
}
```

2.2 LCS

Usage:
Time Complexity: LCS 구현, 간단한 버전

```
#include <bits/stdc++.h>
using namespace std;

int dp[1001][1001] = {0};
string a,b;

int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    //확실한 생각과 계획 코드
    cin >> a >> b;
    for (int i = 1; i <= a.size(); i++){
        for (int j = 1; j <= b.size(); j++){
            if (a[i-1] == b[j-1]){
                dp[i][j] = dp[i-1][j-1]+1;
            }
            dp[i][j] = max(dp[i][j],dp[i-1][j]);
            dp[i][j] = max(dp[i][j],dp[i][j-1]);
        }
    }
    cout << dp[a.size()][b.size()] << '\n';
    return 0;
}
```

2.3 냅색

Usage:
Time Complexity: 냅색 구현, 간단한 버전

```
#include <bits/stdc++.h>
using namespace std;

int dp[101][100001] = {0};
vector <array<int,3> > v1;

int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    //확실한 생각과 계획 코드
    int n,k;
    cin >> n >> k;
    for (int i = 1; i <= n; i++){
        int w,v,K;
        cin >> w >> v >> K;
        v1.push_back({w,v,K});
    }
    sort(v1.begin(),v1.end());
    for (int i = 1; i <= n; i++){
        for (int j = 1; j <= k; j++){
            if (j - v1[i-1][0] >= 0){
                int t = j / v1[i-1][0];
                if (t > v1[i-1][2]){
                    dp[i][j] = max(dp[i-1][j],
                    dp[i-1][j-v1[i-1][0]*v1[i-1][2]] +
                    v1[i-1][1]*v1[i-1][2]);
                }else{
                    dp[i][j] = max(dp[i-1][j],
                    dp[i-1][j-v1[i-1][0]*t] + v1[i-1][1]*t);
                }
            }else{
                dp[i][j] = dp[i-1][j];
            }
        }
    }
    cout << dp[n][k] << '\n';
    return 0;
}
```

2.4 비트마스크 dp

Usage:
Time Complexity: 백준 계단수 문제 코드, 비트 dp의 시작점

```
#include <bits/stdc++.h>
using namespace std;
typedef long long int ll;

ll dp[107][11][2000] = {0};
ll MOD = 1000000000;

int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    //확실한 생각과 계획 코드
    ll n;
    cin >> n;
    for (ll i = 0; i <= 9; i++){
        dp[1][i][(1 << i)] = 1;
    }
    dp[1][0][1] = 0;
    for (int i = 2; i <= n; i++){
        for (int j = 0; j <= 9; j++){
            for (int k = 0; k <= 1023; k++){
                if (dp[i-1][j+1][k] > 0 || dp[i-1][j-1][k] >
                0){
                    dp[i][j][k | (1 << j)] += (dp[i-1][j+1][k]
                    + dp[i-1][j-1][k]) % MOD;
```

```
                    dp[i][j][k | (1 << j)] %= MOD;

                }

            }
        }
    }
    ll ans = 0;
    // for (int i = 1; i <= n; i++){
    //     for (ll j = 0; j <= 9; j++){
    //         ans += dp[i][j][1023];

    //     }
    // }
    for (ll i = 0; i <= 9; i++){
        ans += (dp[n][i][1023]) % MOD;
        ans %= MOD;

    }

    cout << ans << '\n';
    return 0;
}
```

2.5 TSP

Usage:
Time Complexity: 외판원 순회, 구현

```
#include <iostream>
#include <cstring>
using namespace std;

#define INF 987654321;

int n,map[16][16];
int dp[16][1<<16]; //각 마을이 방문한 도시를 2진법으로 저장

int dfs(int cur, int visit){

    if (visit == (1<<n)-1){ //탐색 완료
        if(map[cur][0] == 0) //이동불가능
            return INF;
        return map[cur][0];
    }

    if (dp[cur][visit] != -1) //이미 탐색했으면
        return dp[cur][visit];

    dp[cur][visit] = INF;

    for (int i=0; i<n; i++){
        if (map[cur][i]==0) //길 X
            continue;
        if ((visit & (1<<i)) == (1<<i)) //이미 방문
            continue;
        dp[cur][visit] = min(dp[cur][visit], map[cur][i] +
        dfs(i, visit | 1<<i));
    }

    return dp[cur][visit];
}

int main(int argc, const char * argv[]) {
    ios::sync_with_stdio(0), cin.tie(0), cout.tie(0);

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

    memset(dp, -1, sizeof(dp)); //dp배열 -1로 초기화
    cout<<dfs(0,1);

    return 0;
}
```

3 문자열 알고리즘

3.1 라빈 카프 알고리즘

Usage:
Time Complexity: 라빈 카프 알고리즘 구현, 예시

```
// Algorithm Analysis
// 라빈-카프 알고리즘 (Rabin-Karp)

#include <iostream>

using namespace std;

// ws : 검색 대상 문자열 , ps : 검색할 패턴 문자열
void findString(string ws, string ps) {
  int wsSize = ws.size();
  int psSize = ps.size();

  int wsHash = 0;
  int psHash = 0;
  int power = 1; // 제곱수

  for (int i = 0; i <= wsSize - psSize; i++) {
    if (i == 0) {
      for (int j = 0; j < psSize; j++) {
        wsHash += ws[psSize - 1 - j] * power;
        psHash += ps[psSize - 1 - j] * power;
        if (j < psSize - 1) power *= 3;
      }
    }
    else {
      wsHash = 3 * (wsHash - ws[i - 1] * power) + ws[psSize -
      1 + i];
    }
    if (wsHash == psHash) {
      bool isFind = true;
      // 우연히 해시값이 겹친 경우 위해 문자열 일치 여부 검사
      for (int j = 0; j < psSize; j++) {
        if (ws[i + j] != ps[j]) {
          isFind = false;
          break;
        }
      }
      if (isFind) {
        cout << wsHash << " " << psHash << endl;
        printf("%d번째에서 발견\n", i + 1);
      }
    }
  }
}
int main() {
  string ws = "AABDCDABD";
  string ps = "ABD";
  findString(ws, ps);
}
```

시간 복잡도 : O(N)
라빈-카프 알고리즘의 시간 복잡도는 문자열 길이(N) 만큼의 복잡도를 가집니다.

3.2 kmp 알고리즘

Usage:
Time Complexity: kmp 알고리즘 구현, 백준 찾기와 동일

```
#include <iostream>
#include <vector>
#include <string>

using namespace std;

int lps[1000007] = {0};

void kmp(string total, string search){
    vector <int> v;
    int j = 0;
    for (int i = 1; i < search.size(); i++){
        while(j > 0 && search[i] != search[j]){
            j = lps[j-1];
        }
        if (search[i] == search[j]){
```

```
            lps[i] = ++j;
        }
    }
    int ans = 0;
    j = 0;
    int size = search.size();
    for (int i = 0; i < total.size(); i++){
        while(j > 0 && total[i] != search[j]){
            j = lps[j-1];
        }
        if (total[i] == search[j]){
            if (j == search.size()-1){
                ans++;
                v.push_back(i-j+1);
                j = lps[j];
            }else{
                j++;
            }
        }
    }
    cout << ans << '\n';
    for (int i = 0; i < v.size(); i++) cout << v[i] << ' ';
    cout << '\n';
}
int main(void){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    string a,b;
    getline(cin,a);
    getline(cin,b);
    kmp(a,b);
}
```

3.3 매내처

Usage: Returns palindromic radius of S. To calculate even length palindromes, insert $ between each character.
Time Complexity: $\mathcal{O}(N)$

```
vector<int> M(const vector<int>& S) {
    int N = S.size();
    vector<int> M(N);
    int L = 0, R = 0;
    for(int i = 0; i < N; i++) {
        if(i < R && i+M[2*L-i] < R) M[i] = M[2*L-i];
        else {
            L = i, R = max(R, i);
            while(R < N && 2*i-R >=0 && S[R] == S[2*i-R]) ++R;
            M[i] = R-i;
        }
    }
    return M;
}
```

4 그래프

4.1 DFS, BFS

Usage:
Time Complexity: DFS와 BFS, 간단 구현

```
#include <iostream>
#include <queue>
#include <algorithm>
#include <vector>

using namespace std;

int arr[1001][1001] = {0};
int visited[10001] = {0};
int visited2[10001] = {0};
void dfs(int v){
    visited[v] = 1;
    cout << v << ' ';
    for (int i = 1; i <= 1000; i++){
        if (visited[i] == 0 && arr[v][i] == 1) {
            visited[i] = 1;
            dfs(i);
        }
    }
}
void bfs(int v){
    queue <int> q;
    q.push(v);
    visited2[v] = 1;

    while(!q.empty()){
        int x = q.front();
        q.pop();
        cout << x << ' ';
        for (int i = 1; i <= 1000; i++){
            if (arr[x][i] == 1 && visited2[i] == 0){
                q.push(i);
                visited2[i] = 1;
            }
        }
    }

}
int main(void){
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int n,m,v;
    cin >> n >> m >> v;
    for (int i = 0; i < m; i++){
        int a,b;
        cin >> a >> b;
        arr[a][b] = 1;
        arr[b][a] = 1;
    }
    dfs(v);
    cout << '\n';
    bfs(v);
    cout << '\n';

}
```

4.2 다익스트라 우선순위 큐 알고리즘

Usage:
Time Complexity: 다익스트라 알고리즘 구현, 우선순위 큐 사용버전

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

using namespace std;
int V, E, K;
int dist[20005]; // 가중치
const int INF = 987654321;
vector<pair<int, int>> adj[20005];
priority_queue<pair<int, int>, vector<pair<int, int>>,
greater<pair<int, int>>> pq;

int main() {
   ios::sync_with_stdio(false);
   cin.tie(NULL); cout.tie(NULL);

   cin >> V >> E;
   cin >> K;

   fill(dist, dist + 20005, INF); // 무한대 값으로 초기화

   for (int i = 0; i < E; i++) {
      int u, v, w; // u에서 v까지 가중치 w
      cin >> u >> v >> w;
      adj[u].push_back(make_pair(w, v));
   }

   pq.push(make_pair(0, K)); // 가중치, 시작 위치
   dist[K] = 0;
   while (!pq.empty()) {
      int here = pq.top().second;
      int here_dist = pq.top().first;
      pq.pop();
      if (dist[here] != here_dist) continue; // 마지막 정점인가
      for (pair<int, int> there : adj[here]) {
         int _dist = there.first;
         int _there = there.second;
         if (dist[_there] > dist[here] + _dist) { // 더 작으면
         갱신
```

```
            dist[_there] = dist[here] + _dist;
            pq.push(make_pair(dist[_there], _there));
        }
    }
}

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

return 0;
}
```

4.3 벨만포드 알고리즘

Usage:
Time Complexity: 벨만포드 알고리즘 구현, 간단한 틀

```
void Bellman_Ford()
{
    Dist[1] = 0;
    for (int i = 1; i <= N - 1; i++)
    {
        for (int j = 0; j < Edge.size(); j++)
        {
            int From = Edge[j].first.first;
            int To = Edge[j].first.second;
            int Cost = Edge[j].second;

            if (Dist[From] == INF) continue;
            if (Dist[To] > Dist[From] + Cost) Dist[To] =
            Dist[From] + Cost;
        }
    }

    for (int i = 0; i < Edge.size(); i++)
    {
        int From = Edge[i].first.first;
        int To = Edge[i].first.second;
        int Cost = Edge[i].second;

        if (Dist[From] == INF) continue;
        if (Dist[To] > Dist[From] + Cost)
        {
            cout << "음의 사이클이 존재하는 그래프입니다." <<
            endl;
            return;
        }
    }
    cout << "음의 사이클이 존재하지 않는, 정상적인 그래프 입니다."
    << endl;
}
```

4.4 플로이드 워셜

다익스트라 여러개 쓰는 걸로 대체

4.5 크루스칼 알고리즘

Usage:
Time Complexity: 크루스칼 알고리즘 구현, 유니온 파인드 경로 압축 버전 사용

```
#include <iostream>
#include <vector>
#include <array>
#include <algorithm>

using namespace std;
typedef long long int ll;

ll parent[10001];

ll Find(ll a){
    if (parent[a] == a){
        return a;
    }
    return parent[a] = Find(parent[a]);
}

int Union(ll a, ll b){
    a = Find(a);
    b = Find(b);
    if (a != b){
        parent[b] = a;
        return 1;
    }else{
        return 0;
    }
}
vector <array <ll,3> > v;

int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    cout.tie(nullptr);
    int V,E;
    cin >> V >> E;
    for (int i = 0; i < 10001; i++){
        parent[i] = i;
    }
    for (int i = 0; i < E; i++){
        ll a,b,c;
        cin >> a >> b >> c;
        v.push_back({c,a,b});
    }
    sort(v.begin(), v.end());
    ll ans = 0;
    for (auto w : v){
        if (Union(w[1], w[2])){
            ans += w[0];
        }
    }
    cout << ans << '\n';
    return 0;
}
```