Team note for ICPC (2021)

mskim17, firebird, flame623

October 2021

Contents

1 제출 전 확인 사항

- 문제를 제대로 읽었는가
- 컴파일 여부를 확인했는가 (제출 언어 설정이 바른가)
- 매우 작은 입력 중 반례가 있는가
- 중간 결과값이 int 범위를 벗어날 수 있는가
- void가 아니며 return 값을 가지지 않는 함수가 있는가

2 기하

2.1 Point 구조체

```
struct point {
    int x;
    int y;
    point() {}
    point(int x, int y): x(x), y(y) {}

    bool operator<(const point& other) const {
        return (x != other.x) ? x < other.x : y < other.y;
    }
    point operator+(const point& other) const {
        return point(x + other.x, y + other.y);
    }
    point operator-(const point& other) const {
        return point(x - other.x, y - other.y);
    }
};

long long distance(const point& p1, const point& p2) {
    return 1LL * (p1.x - p2.x) * (p1.x - p2.x) + 1LL * (p1.y - p2.y)
     * (p1.y - p2.y);
}
```

2.2 CCW (반시계 방향 판단 알고리즘)

```
int ccw(const point& p1, const point& p2, const point& p3) {
    int x1 = p1.x; int y1 = p1.y;
    int x2 = p2.x; int y2 = p2.y;
    int x3 = p3.x; int y3 = p3.y;

    long long s = 1LL * (x2 - x1)*(y3 - y1) - 1LL * (y2 - y1)*(x3 -
    x1);  // twice the area of a triangle
    if (s > 0) return 1;     // counter-clockwise
    if (s < 0) return -1;    // clockwise
    return 0;                // straight line
}
```

2.3 볼록 껍질 알고리즘

```
vector <point> points;
vector <point> convex_hull;

void get_convex_hull (vector <point> &convex_hull, vector <point> &
    points) {
    for (int i=0; i<points.size(); i++) {
        if (points[i] < points[0]) swap(points[0], points[i]);
    }

    sort(points.begin() + 1, points.end(), [&points](const point &p1
    , const point &p2) {
        int temp = ccw(points[0], p1, p2);
        if (temp == 0) return distance(points[0], p1) < distance(
    points[0], p2);
        return temp > 0;
    });

    for (int i=0; i<points.size(); i++) {
        while (convex_hull.size() > 1) {
            point p1 = convex_hull[convex_hull.size() - 2];
            point p2 = convex_hull[convex_hull.size() - 1];
            point p3 = points[i];

            if (ccw(p1, p2, p3) > 0) break;
            convex_hull.pop_back();
        }
        convex_hull.push_back(points[i]);
    }
}
```

2.4 회전하는 캘리퍼스

```
pair <point, point> rotating_calipers (vector <point> &convex_hull) {

    int m = convex_hull.size();

    int a = 0;
    int c = 1;
    int max_x = convex_hull[c].x;
    for (int i=2; i<convex_hull.size(); i++) {
        if (convex_hull[i].x > max_x) {
            max_x = convex_hull[i].x;
            c = i;
        }
    }

    int max_a, max_c;
    long long max_dist = 0;
    while (true) {
        while (true) {
            long long dist = distance(convex_hull[a], convex_hull[c]);
            if (dist > max_dist) {
                max_a = a;
                max_c = c;
                max_dist = dist;
            }
            int b = (a + 1) % m;
            int d = (c + 1) % m;
            point p1 = convex_hull[b] + convex_hull[d] - convex_hull[c];
            if (ccw(convex_hull[a], convex_hull[b], p1) == -1) {
                a = b;
                break;
            }
            c = d;
        }
        if (a == 0) break;
    }
    return make_pair(convex_hull[max_a], convex_hull[max_c]);

}
```

2.5 선분 교차 판정

2.5.1 단순 교차 판정

```
bool is_intersect(point p1, point p2, point p3, point p4) {
    int ab = ccw(p1, p2, p3) * ccw(p1, p2, p4);
    int cd = ccw(p3, p4, p1) * ccw(p3, p4, p2);
    if (ab != 0 || cd != 0)
        return (ab <= 0 && cd <= 0);
    else {
        int x1 = p1.x; int y1 = p1.y;
        int x2 = p2.x; int y2 = p2.y;
        int x3 = p3.x; int y3 = p3.y;
        int x4 = p4.x; int y4 = p4.y;
        if (x1 > x2) swap(x1, x2);
        if (x3 > x4) swap(x3, x4);
        if (x1 > x3) {
            swap(x1, x3);
            swap(x2, x4);
        }
        if (y1 > y2) swap(y1, y2);
        if (y3 > y4) swap(y3, y4);
        if (y1 > y3) {
            swap(y1, y3);
            swap(y2, y4);
        }
        return (x3 <= x2 && y3 <= y2);
    }
}
```

2.5.2 교차점

```
pair< pair<bool, bool>, point> get_intersection(point p1, point p2, point p3, point p4) {
    if (!is_intersect(p1, p2, p3, p4))
        return make_pair(make_pair(false, false), point(0, 0));
    bool is_only = true;
    point result = point(0, 0);

    if (p1.x == p2.x) {
        result.x = p1.x;
        if (p3.x != p4.x)
            result.y = (double)(p4.y - p3.y)/(p4.x - p3.x) * (p1.x - p3.x) + p3.y;
        else {
            if (p1.y > p2.y) swap(p1, p2);
            if (p3.y > p4.y) swap(p3, p4);
            if (p1.y > p3.y) {
                swap(p1, p3);
                swap(p2, p4);
                if (p2.y == p3.y) result.y = p2.y;
                else is_only = false;
            }
        }
    }
    else if (p1.y == p2.y) {
        result.y = p1.y;
        if (p3.y != p4.y)
            result.x = (p4.x - p3.x)/(p4.y - p3.y) * (p1.y - p3.y) + p3.x;
        else {
            if (p1.x > p2.x) swap(p1, p2);
            if (p3.x > p4.x) swap(p3, p4);
            if (p1.x > p3.x) {
                swap(p1, p3);
                swap(p2, p4);
                if (p2.x == p3.x) result.x = p2.x;
                else is_only = false;
            }
        }
    }
    else if (p3.x == p4.x) {
        result.x = p3.x;
        result.y = (p2.y - p1.y)/(p2.x - p1.x) * (p3.x - p1.x) + p1.y;
    }
    else if (p3.y == p4.y) {
        result.y = p3.y;
        result.x = (p2.x - p1.x)/(p2.y - p1.y) * (p3.y - p1.y) + p1.x;
    }
    else {
        double a = (p2.y - p1.y)/(p2.x - p1.x);
        double b = p1.y - p1.x * (p2.y - p1.y)/(p2.x - p1.x);
        double c = (p4.y - p3.y)/(p4.x - p3.x);
        double d = p3.y - p3.x * (p4.y - p3.y)/(p4.x - p3.x);

        if (a != c) {
            result.x = (d-b) / (a-c);
            result.y = a*result.x + b;
        }
        else {
            if (p1 > p2) swap(p1, p2);
            if (p3 > p4) swap(p3, p4);
            if (p1 > p3) {
                swap(p1, p3);
                swap(p2, p4);
            }
            if (p2.x == p3.x) {
                result.x = p2.x;
                result.y = p2.y;
            }
            else is_only = false;

        }
    }
    return make_pair(make_pair(true, is_only), result);
}
```

2.6 좌표 압축

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

const int max_n = 1000000;
vector <int> arr(max_n);
vector <int> idx;

void initialize(int n) {
    sort(idx.begin(), idx.end());
    idx.erase(unique(idx.begin(), idx.end()), idx.end());
}

int get_idx(int x) {
    return lower_bound(idx.begin(), idx.end(), x) - idx.begin();
}
```

3 문자열

3.1 KMP

```
vector <int> get_pi (string p) {
    int m = (int) p.size();
    vector<int> pi(m);
    pi[0] = 0;
    int j = 0;
    for (int i=1; i<m; i++) {
        while (j > 0 && p[i] != p[j])
            j = pi[j-1];
        if (p[i] == p[j]) {
            j++;
            pi[i] = j;
        }
    }
    return pi;
}

vector <int> kmp(string t, string p) {
    auto pi = get_pi(p);
    vector<int> result;
    int n = (int) t.size();
    int m = (int) p.size();
    int idx = 0;
    for (int i=0; i<n+1; i++) {
        if (idx == m) {
            result.push_back(i+1-m);
            idx = pi[idx - 1];
        }
        if (i == n) break;
        while (idx > 0 && t[i] != p[idx])
            idx = pi[idx - 1];
        if (t[i] == p[idx])
            idx++;
    }
    return result;
}
```

4 그래프

4.1 최단 경로 알고리즘

4.1.1 다익스트라

```
const int INF = 987654321;
int max_size = 20001;
vector <vector <pair<int, int>>> graph(max_size);
priority_queue <pair <int, int> > pq;
vector<int> dist(max_size, INF);

void dijkstra(int start) {
    pq.push({0, start});
    dist[start] = 0;
    while (!pq.empty()) {
        int d = -pq.top().first;
        int node = pq.top().second;
        pq.pop();
        if (d > dist[node]) continue;

        for (auto p: graph[node]) {
            int nxt = p.first;
            int cost = p.second;
            int dd = d + cost;
            if (dist[nxt] > dd) {
                dist[nxt] = dd;
                pq.push({-dd, nxt});
            }
        }
    }
}
```

4.1.2 벨만 포드

```
const long long INF = 987654321;
int max_size = 501;
bool cycle = false;
vector <vector <pair<int, long long>>> graph(max_size);
vector <long long> dist(max_size, INF);

void bellman_ford(int start, int n) {
    dist[start] = 0;
    for (int i=1; i<=n; i++) {
        for (int j=1; j<=n; j++) {
            for (auto &p: graph[j]) {
                if (dist[j] != INF && dist[p.first] > p.second +
    dist[j]) {
                    dist[p.first] = p.second + dist[j];
                    if (i==n) cycle = true;
                }
            }
        }
    }
}
```

4.1.3 플로이드 워셜

```
const int INF = 987654321;
int max_size = 101;
vector <vector <int>> dist(max_size, vector <int> (max_size, INF));

void floyd_warshall(int n) {
    for (int i=0; i<n; i++) dist[i][i] = 0;
    for (int i=0; i<n; i++)
        for (int start=0; start<n; start++)
            for (int end=0; end<n; end++)
                dist[start][end] = min(dist[start][end], dist[start
    ][i] + dist[i][end]);
}
```

4.2 강한 결합 요소

```
#include <iostream>
#include <vector>
#include <set>
#include <stack>
#include <algorithm>
using namespace std;

const int MAX = 20001;

vector <vector <int>> graph(MAX);
vector <vector <int>> graph_rev(MAX);
vector <vector <int>> scc_list;
stack <int> s;
int T, V, E;
char visited[MAX];
int scc_idx[MAX];


void dfs(int node) {
    for (int nxt : graph[node]) {
        if (!visited[nxt]) {
            visited[nxt] = 1;
            dfs(nxt);
        }
    }
    s.push(node);
}

void dfs_rev(vector <int> &scc, int node) {
    scc.push_back(node);
    for (int nxt : graph_rev[node]) {
        if (!visited[nxt]) {
            visited[nxt] = 1;
            dfs_rev(scc, nxt);
        }
    }
}

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

    cin >> V >> E;
    int A, B;
    int notA = 0; int notB = 0;
    for (int i=0; i<E; i++) {
        notA = 0;
        notB = 0;
        cin >> A >> B;
        if (A < 0) {
            notA = 1;
            A *= -1;
        }
        if (B < 0) {
            notB = 1;
            B *= -1;
        }
        int rev_A = ((notA + 1) % 2) * 10000 + A;
        int rev_B = ((notB + 1) % 2) * 10000 + B;
        A = notA * 10000 + A;
        B = notB * 10000 + B;
        // cout << "(" << rev_A << "," << B << ") (" << rev_B << ","
     << A << ")\n";
        graph[rev_A].push_back(B);
        graph_rev[B].push_back(rev_A);
        graph[rev_B].push_back(A);
        graph_rev[A].push_back(rev_B);
    }

    for (int i=1; i<V+1; i++) {
        if (!visited[i]) dfs(i);
    }

    for (int i=0; i<MAX; i++) visited[i] = 0;
    while (!s.empty()) {
        int node = s.top();
        vector <int> scc;
        s.pop();
        if (!visited[node]) {
            visited[node] = 1;
            dfs_rev(scc, node);
```

```
            sort(scc.begin(), scc.end());
            scc_list.push_back(scc);
        }
    }

    int scc_n = scc_list.size();
    vector <int> indegree (scc_n);

    int idx = 1;
    for (auto scc : scc_list) {
        for (int node : scc) {
            scc_idx[node] = idx;
        }
        idx++;
    }

    int result = 1;
    for (int i=1; i<=V; i++) {
        if (scc_idx[i] == scc_idx[10000 + i])
            result = 0;
    }

    cout << result << '\n';

    return 0;
}
```

4.2.1 2-SAT (가능 여부 판별)

```
#include <iostream>
#include <vector>
#include <set>
#include <stack>
#include <algorithm>
using namespace std;

const int MAX = 20001;

vector <vector <int>> graph(MAX);
vector <vector <int>> graph_rev(MAX);
vector <vector <int>> scc_list;
stack <int> s;
int T, V, E;
char visited[MAX];
int scc_idx[MAX];


void dfs(int node) {
    for (int nxt : graph[node]) {
        if (!visited[nxt]) {
            visited[nxt] = 1;
            dfs(nxt);
        }
    }
    s.push(node);
}

void dfs_rev(vector <int> &scc, int node) {
    scc.push_back(node);
    for (int nxt : graph_rev[node]) {
        if (!visited[nxt]) {
            visited[nxt] = 1;
            dfs_rev(scc, nxt);
        }
    }
}

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

    cin >> V >> E;
    int A, B;
    int notA = 0; int notB = 0;
    for (int i=0; i<E; i++) {
        notA = 0;
        notB = 0;
        cin >> A >> B;
        if (A < 0) {
            notA = 1;
            A *= -1;
        }
        if (B < 0) {
            notB = 1;
            B *= -1;
        }
        int rev_A = ((notA + 1) % 2) * 10000 + A;
        int rev_B = ((notB + 1) % 2) * 10000 + B;
        A = notA * 10000 + A;
        B = notB * 10000 + B;
        // cout << "(" << rev_A << "," << B << ") (" << rev_B << ","
     << A << ")\n";
        graph[rev_A].push_back(B);
        graph_rev[B].push_back(rev_A);
        graph[rev_B].push_back(A);
        graph_rev[A].push_back(rev_B);
    }

```

```
    for (int i=1; i<V+1; i++) {
        if (!visited[i]) dfs(i);
    }

    for (int i=0; i<MAX; i++) visited[i] = 0;
    while (!s.empty()) {
        int node = s.top();
        vector <int> scc;
        s.pop();
        if (!visited[node]) {
            visited[node] = 1;
            dfs_rev(scc, node);
            sort(scc.begin(), scc.end());
            scc_list.push_back(scc);
        }
    }

    int scc_n = scc_list.size();
    vector <int> indegree (scc_n);

    int idx = 1;
    for (auto scc : scc_list) {
        for (int node : scc) {
            scc_idx[node] = idx;
        }
        idx++;
    }

    int result = 1;
    for (int i=1; i<=V; i++) {
        if (scc_idx[i] == scc_idx[10000 + i])
            result = 0;
    }

    cout << result << '\n';

    return 0;
}
```

4.3 최대 유량

4.3.1 에드몬드 카프

```
#include <iostream>
#include <vector>
#include <queue>
#include <unordered_set>
#include <algorithm>
using namespace std;

const int SIZE = 52;
const int INF = 1e9;

int n;
vector <unordered_set <int>> graph(SIZE);
vector <vector <int>> c(SIZE, vector<int> (SIZE, 0));
vector <vector <int>> f(SIZE, vector<int> (SIZE, 0));

const int source = 0;
const int sink = SIZE - 1;


int edmonds_karp(int source, int sink) {
    int max_flow = 0;
    while (true) {

        int visited[SIZE];
        for (int i=0; i<SIZE; i++) visited[i] = -1;

        queue <int> q;
        q.push(source);
        while (!q.empty()) {
            int node = q.front();
            q.pop();
            for (auto &nxt: graph[node]) {
                if (c[node][nxt] - f[node][nxt] > 0 && visited[nxt]
    == -1) {
                    q.push(nxt);
                    visited[nxt] = node;
                    if (nxt == sink) break;
                }
            }
        }
        if (visited[sink] == -1) break;
        int curr_flow = INF;
        int node = sink;
        while (node != source) {
            curr_flow = min(curr_flow, c[visited[node]][node] - f[
    visited[node]][node]);
```

```
            node = visited[node];
        }
        node = sink;
        while (node != source) {
            f[visited[node]][node] += curr_flow;
            f[node][visited[node]] -= curr_flow;
            node = visited[node];
        }
        max_flow += curr_flow;

    }
    return max_flow;
}
```

4.3.2 이분 매칭

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

const int SIZE = 1001;
const int INF = 1e9;

int n, m;
vector <vector <int>> graph(SIZE);
bool visited[SIZE];
int work[SIZE];

bool dm_dfs (int start) {
    visited[start] = 1;
    for (auto &i : graph[start]) {
        if (work[i] == 0 || (!visited[work[i]] && dm_dfs(work[i])))
    {
            work[i] = start;
            return 1;
        }
    }
    return 0;
}

int bipartite_matching(int n) {
    int result = 0;
    for (int i=1; i<=n; i++) {
        for (int j=0; j<SIZE; j++) visited[j] = 0;
        if (dm_dfs(i)) result++;
    }
    return result;
}
```

5 트리

5.1 최소 신장 트리

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

const int SIZE = 10001;

vector <vector <int>> edge;
vector <int> parent(SIZE); // must be initialized by parent[i] = i;
vector <int> p_rank(SIZE, 1);

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

void merge(int x, int y) {
    x = find(x);
    y = find(y);
    if (x == y) return;
    if (p_rank[x] > p_rank[y]) swap(x, y);
    if (p_rank[x] == p_rank[y]) p_rank[y]++;
    parent[x] = y;
}


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

    for (int i=0; i<SIZE; i++) parent[i] = i;

    cin >> v >> e;
    for (int i=0; i<e; i++) {
        int a, b, c;
        cin >> a >> b >> c;
        edge.push_back({-c, a, b});
    }
    sort(edge.begin(), edge.end());
    int result = 0;
    while (!edge.empty()) {
        auto ee = edge.back();
        edge.pop_back();
        int c = -ee[0];
```

```
        int a = ee[1];
        int b = ee[2];
        if (find(a) != find(b)) {
            merge(a, b);
            result += c;
        }
    }

    cout << result << endl;


    return 0;
}
```

5.2 세그먼트 트리 (느린 갱신)

```
const int tree_size = 2097152;
long long seg_tree[tree_size];
long long lazy[tree_size];
long long arr[tree_size / 2];


long long generate(int node, int start, int end) {
    if (start == end) {
        return seg_tree[node] = arr[start];
    }
    int mid = (start + end) / 2;
    return seg_tree[node] = generate(node * 2, start, mid) +
    generate(node * 2 + 1, mid + 1, end);
}


void update_lazy(int node, int start, int end) {
    if (lazy[node] != 0) {
        seg_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(int node, int start, int end, int left, int right,
     long long diff) {
    update_lazy(node, start, end);
    if (left > end || right < start) return;
    if (left <= start && end <= right) {
        seg_tree[node] += (end - start + 1) * diff;
        if (start != end) {
            lazy[node * 2] += diff;
            lazy[node * 2 + 1] += diff;
        }
        return;
    }
    int mid = (start + end) / 2;
    update_range(node * 2, start, mid, left, right, diff);
    update_range(node * 2 + 1, mid + 1, end, left, right, diff);
    seg_tree[node] = seg_tree[node * 2] + seg_tree[node * 2 + 1];
}


long long sum(int node, int start, int end, int left, int right) {
    update_lazy(node, start, end);
    if (left > end || right < start) return 0;
    if (left <= start && end <= right) return seg_tree[node];

    int mid = (start + end) / 2;
    return sum(node * 2, start, mid, left, right) + sum(node * 2 +
    1, mid + 1, end, left, right);
}
```

5.3 최소 공통 조상

```cpp
#include <iostream>
#include <cmath>
#include <vector>
#include <queue>
#include <algorithm>
using namespace std;

const int max_n = 100005;
const int max_exp = 17;
vector <int> log_2(max_n);
vector <vector <int>> edge(max_n);
vector <vector <int>> parent(max_n, vector <int> (max_exp));
vector <int> depth(max_n);

void initialize(int n) {
    int power = 0;
    while (pow(2, power) < max_n+1) {
        for (int i=(int) pow(2, power); i<min(max_n+1, (int) pow(2, power+1)); i++)
            log_2[i] = power;
        power++;
    }

    for (int i=0; i<max_n; i++)
        parent[i][0] = -1;

    parent[0][0] = 0;
    parent[1][0] = 0;
    depth[1] = 1;

    queue <int> q;
    q.push(1);
    while (!q.empty()) {
        int node = q.front();
        q.pop();
        for (int &child : edge[node]) {
            if (parent[child][0] != -1) continue;
            depth[child] = depth[node] + 1;
            parent[child][0] = node;
            q.push(child);
        }
    }

    for (int k=1; k<max_exp; k++) {
        for (int i=1; i<n+1; i++) {
            parent[i][k] = parent[ parent[i][k-1] ][k-1];
        }
    }
}

int get_lca(int u, int v) {
    if (depth[u] > depth[v]) swap(u, v);
    int gap = depth[v] - depth[u];
    int m = depth[v];

    for (int i=log_2[m]; i>=0; i--) {
        if (gap >= pow(2, i)) {
            v = parent[v][i];
            gap = depth[v] - depth[u];
        }
    }
    if (u == v) return u;
    while (u != v) {
        int m = depth[v];
        bool terminate = false;
        for (int i=log_2[m]; i>=0; i--) {
            if (parent[u][i] != parent[v][i]) {
                u = parent[u][i];
                v = parent[v][i];
                terminate = true;
                break;
            }
        }
        if (!terminate) {
            u = parent[u][0];
            v = parent[v][0];
            break;
        }
    }
    return u;
}
```

6 기타

6.1 서로소 집합

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

const int SIZE = 100001;

vector <int> parent(SIZE); // must be initialized by parent[i] = i;
vector <int> rank(SIZE, 1);

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

void merge(int x, int y) {
    x = find(x);
    y = find(y);
    if (x == y) return;
    if (rank[x] > rank[y]) swap(x, y);
    if (rank[x] == rank[y]) rank[y]++;
    parent[x] = y;
}
```

6.2 DP 응용

6.2.1 배낭 문제

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

const int SIZE = 100001;
vector <int> dp(SIZE, -1);


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

    dp[0] = 0;
    cin >> n >> k;
    for (int i=0; i<n; i++) {
        int w, v;
        cin >> w >> v;
        for (int j=k; j>=w; j--) {
            if (dp[j-w] == -1) continue;
            dp[j] = max(dp[j], v + dp[j-w]);
        }
    }

    int result = 0;
    for (int i=0; i<SIZE; i++) result = max(result, dp[i]);
    cout << result << endl;

    return 0;
}
```

7 템플릿

7.1 빠른 입출력

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

int main() {
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);
    return 0;
}
```

7.2 string to XXX 예제

```cpp
// Reference: https://blockdmask.tistory.com/333

#include <iostream>
#include <string>
using namespace std;

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

    string str_i = "22";
    string str_li = "2144967290";
    string str_f = "3.4";
    string str_d = "2.11";

    int i       = stoi(str_i);
    long int li = stol(str_li);
    float f     = stof(str_f);
    double d    = stod(str_d);

    //C++ cout
    cout << "stoi : " << i    << endl;
    cout << "stol : " << li   << endl;
    cout << "stof : " << f    << endl;
    cout << "stod : " << d    << endl;
}
```

7.3 여러 input 방법

```cpp
// Reference: https://jhnyang.tistory.com/321
//            https://dbstndi6316.tistory.com/33

#include <iostream>
#include <string>
using namespace std;

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

    string name;
    getline(cin, name);
    cout << name << endl;

    char c;
    c = getchar();
    cout << c << endl;

}
```

7.4 기타 string 관련 함수

```cpp
// Reference: https://psychoria.tistory.com/773
//            https://blockdmask.tistory.com/338

#include <iostream>
#include <string>
using namespace std;

int main()
{
    string numbers = "0123456789";

    cout << numbers.at(6) << endl; //6
    cout << numbers.back() << endl; //9
    cout << numbers.size() << endl; //10


    string resize = "987654321";

    resize.resize(5);
    cout << resize << endl; //98765
    resize.resize(10, '0'); //only char type
    cout << resize << endl; //9876500000


    string full = numbers.substr();
    string sub = numbers.substr(3, 5);

    cout << "Full String: " << full << endl; //0123456789
```

```
    cout << "Sub String: " << sub << endl; //34567


    string replace = "ABCDE";
    replace.replace(3,0, "ZZZ");
    cout << replace << endl; //ABCZZZDE
    replace.replace(0,100, "New Word");
    cout << replace << endl; //New Word


    string find = "abcdbc";
    cout << find.find("bc") << endl; //1, return first matched index

    for (string::iterator iter = find.begin(); iter != find.end(); ++iter){
        cout << *iter << endl;
    }


    string path = "file.txt.png";
    size_t pos = path.rfind('.'); //return last matched index

    string filename = path.substr(0, pos);
    string extension = path.substr(pos + 1);

    cout << "Filename: " << filename << endl;
    cout << "Extension: " << extension << endl;
}
```

7.5 구조체, 연산자 오버로딩을 활용한 우선순위 큐

```
// Reference: https://kbj96.tistory.com/15

#include <iostream>
#include <queue>
#include <functional>    // greater
using namespace std;

struct Student {
    int id;
    int math, eng;
    Student(int num, int m, int e) : id(num), math(m), eng(e) {}
};

struct cmp {
    bool operator()(Student a, Student b) {
        return a.id < b.id;
    }
};

int main() {
    priority_queue<Student, vector<Student>, cmp> pq;


    pq.push(Student(3, 100, 50));
    pq.push(Student(1, 60, 50));
    pq.push(Student(2, 80, 50));
    pq.push(Student(4, 90, 50));
    pq.push(Student(5, 70, 40));

    while (!pq.empty()) {
        Student ts = pq.top(); pq.pop();
        cout << "(id, math, english) : " << ts.id << ' ' << ts.math << ' ' << ts.eng << '\n';
    }

    return 0;
}
```