

Team note for ICPC (2021)

mskim17, firebird, flame623

October 2021

Contents

1	제출 전 확인 사항	2	4.2	강한 결합 요소	6
2	기하	2	4.2.1	2-SAT (가능 여부 판별)	7
2.1	Point 구조체	2	4.3	최대 유량	8
2.2	CCW (반시계 방향 판단 알고리즘)	2	4.3.1	에드몬드 카프	8
2.3	볼록 껍질 알고리즘	2	4.3.2	이분 매칭	9
2.4	회전하는 캘리퍼스	3	5	트리	9
2.5	선분 교차 판정	3	5.1	최소 신장 트리	9
2.5.1	단순 교차 판정	3	5.2	세그먼트 트리 (노린 갱신)	10
2.5.2	교차점	4	5.3	최소 공통 조상	11
2.6	좌표 압축	4	6	기타	12
3	문자열	5	6.1	서로소 집합	12
3.1	KMP	5	6.2	DP 응용	12
4	그래프	5	6.2.1	배낭 문제	12
4.1	최단 경로 알고리즘	5	7	템플릿	13
4.1.1	다익스트라	5	7.1	빠른 입출력	13
4.1.2	벨만 포드	5	7.2	string to XXX 예제	13
4.1.3	플로이드 워셜	6	7.3	여러 input 방법	13
			7.4	기타 string 관련 함수	13
			7.5	구조체, 연산자 오버로딩을 활용한 우선순위 큐	14

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

```
1 struct point {
2     int x;
3     int y;
4     point() {}
5     point(int x, int y): x(x), y(y) {}
6
7     bool operator<(const point& other) const {
8         return (x != other.x) ? x < other.x : y < other.y;
9     }
10    point operator+(const point& other) const {
11        return point(x + other.x, y + other.y);
12    }
13    point operator-(const point& other) const {
14        return point(x - other.x, y - other.y);
15    }
16 };
17
18 long long distance(const point& p1, const point& p2) {
19     return 1LL * (p1.x - p2.x) * (p1.x - p2.x) + 1LL * (p1.y - p2.y)
20         * (p1.y - p2.y);
21 }
```

2.2 CCW (반시계 방향 판단 알고리즘)

```
1 int ccw(const point& p1, const point& p2, const point& p3) {
2     int x1 = p1.x; int y1 = p1.y;
3     int x2 = p2.x; int y2 = p2.y;
4     int x3 = p3.x; int y3 = p3.y;
5
6     long long s = 1LL * (x2 - x1)*(y3 - y1) - 1LL * (y2 - y1)*(x3 -
7     x1); // twice the area of a triangle
8     if (s > 0) return 1; // counter-clockwise
9     if (s < 0) return -1; // clockwise
10    return 0; // straight line
11 }
```

2.3 볼록 껍질 알고리즘

```
1 vector <point> points;
2 vector <point> convex_hull;
3
4 void get_convex_hull (vector <point> &convex_hull, vector <point> &
5 points) {
6     for (int i=0; i<points.size(); i++) {
7         if (points[i] < points[0]) swap(points[0], points[i]);
8     }
9
10    sort(points.begin() + 1, points.end(), [&points](const point &p1
11    , const point &p2) {
12        int temp = ccw(points[0], p1, p2);
13        if (temp == 0) return distance(points[0], p1) < distance(
14        points[0], p2);
15        return temp > 0;
16    });
17
18    for (int i=0; i<points.size(); i++) {
19        while (convex_hull.size() > 1) {
20            point p1 = convex_hull[convex_hull.size() - 2];
21            point p2 = convex_hull[convex_hull.size() - 1];
22            point p3 = points[i];
23
24            if (ccw(p1, p2, p3) > 0) break;
25            convex_hull.pop_back();
26        }
27        convex_hull.push_back(points[i]);
28    }
29 }
```

2.4 회전하는 캘리퍼스

```
1 pair <point, point> rotating_calipers (vector <point> &convex_hull)
2 {
3     int m = convex_hull.size();
4
5     int a = 0;
6     int c = 1;
7     int max_x = convex_hull[c].x;
8     for (int i=2; i<convex_hull.size(); i++) {
9         if (convex_hull[i].x > max_x) {
10             max_x = convex_hull[i].x;
11             c = i;
12         }
13     }
14
15     int max_a, max_c;
16     long long max_dist = 0;
17     while (true) {
18         while (true) {
19             long long dist = distance(convex_hull[a], convex_hull[c
20 ]);
21             if (dist > max_dist) {
22                 max_a = a;
23                 max_c = c;
24                 max_dist = dist;
25             }
26             int b = (a + 1) % m;
27             int d = (c + 1) % m;
28             point p1 = convex_hull[b] + convex_hull[d] - convex_hull
29 [c];
30             if (ccw(convex_hull[a], convex_hull[b], p1) == -1) {
31                 a = b;
32                 break;
33             }
34             c = d;
35         }
36         if (a == 0) break;
37     }
38     return make_pair(convex_hull[max_a], convex_hull[max_c]);
39 }
```

2.5 선분 교차 판정

2.5.1 단순 교차 판정

```
1 bool is_intersect(point p1, point p2, point p3, point p4) {
2     int ab = ccw(p1, p2, p3) * ccw(p1, p2, p4);
3     int cd = ccw(p3, p4, p1) * ccw(p3, p4, p2);
4     if (ab != 0 || cd != 0)
5         return (ab <= 0 && cd <= 0);
6     else {
7         int x1 = p1.x; int y1 = p1.y;
8         int x2 = p2.x; int y2 = p2.y;
9         int x3 = p3.x; int y3 = p3.y;
10        int x4 = p4.x; int y4 = p4.y;
11        if (x1 > x2) swap(x1, x2);
12        if (x3 > x4) swap(x3, x4);
13        if (x1 > x3) {
14            swap(x1, x3);
15            swap(x2, x4);
16        }
17        if (y1 > y2) swap(y1, y2);
18        if (y3 > y4) swap(y3, y4);
19        if (y1 > y3) {
20            swap(y1, y3);
21            swap(y2, y4);
22        }
23        return (x3 <= x2 && y3 <= y2);
24    }
25 }
```

2.5.2 교차점

```
1 pair< pair<bool, bool>, point> get_intersection(point p1, point p2,
2 point p3, point p4) {
3     if (!is_intersect(p1, p2, p3, p4))
4         return make_pair(make_pair(false, false), point(0, 0));
5     bool is_only = true;
6     point result = point(0, 0);
7
8     if (p1.x == p2.x) {
9         result.x = p1.x;
10        if (p3.x != p4.x)
11            result.y = (double)(p4.y - p3.y)/(p4.x - p3.x) * (p1.x -
12 p3.x) + p3.y;
13        else {
14            if (p1.y > p2.y) swap(p1, p2);
15            if (p3.y > p4.y) swap(p3, p4);
16            if (p1.y > p3.y) {
17                swap(p1, p3);
18                swap(p2, p4);
19                if (p2.y == p3.y) result.y = p2.y;
20                else is_only = false;
21            }
22        }
23    }
24    else if (p1.y == p2.y) {
25        result.y = p1.y;
26        if (p3.y != p4.y)
27            result.x = (p4.x - p3.x)/(p4.y - p3.y) * (p1.y - p3.y) +
28 p3.x;
29        else {
30            if (p1.x > p2.x) swap(p1, p2);
31            if (p3.x > p4.x) swap(p3, p4);
32            if (p1.x > p3.x) {
33                swap(p1, p3);
34                swap(p2, p4);
35                if (p2.x == p3.x) result.x = p2.x;
36                else is_only = false;
37            }
38        }
39    }
40    else if (p3.x == p4.x) {
41        result.x = p3.x;
42        result.y = (p2.y - p1.y)/(p2.x - p1.x) * (p3.x - p1.x) + p1.
43 y;
44    }
45    else if (p3.y == p4.y) {
46        result.y = p3.y;
47        result.x = (p2.x - p1.x)/(p2.y - p1.y) * (p3.y - p1.y) + p1.
48 x;
```

```
44 }
45 else {
46     double a = (p2.y - p1.y)/(p2.x - p1.x);
47     double b = p1.y - p1.x * (p2.y - p1.y)/(p2.x - p1.x);
48     double c = (p4.y - p3.y)/(p4.x - p3.x);
49     double d = p3.y - p3.x * (p4.y - p3.y)/(p4.x - p3.x);
50
51     if (a != c) {
52         result.x = (d-b) / (a-c);
53         result.y = a*result.x + b;
54     }
55     else {
56         if (p1 > p2) swap(p1, p2);
57         if (p3 > p4) swap(p3, p4);
58         if (p1 > p3) {
59             swap(p1, p3);
60             swap(p2, p4);
61         }
62         if (p2.x == p3.x) {
63             result.x = p2.x;
64             result.y = p2.y;
65         }
66         else is_only = false;
67     }
68 }
69 }
70 return make_pair(make_pair(true, is_only), result);
71 }
```

2.6 좌표 압축

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 const int max_n = 1000000;
7 vector <int> arr(max_n);
8 vector <int> idx;
9
10 void initialize(int n) {
11     sort(idx.begin(), idx.end());
12     idx.erase(unique(idx.begin(), idx.end()), idx.end());
13 }
14
15 int get_idx(int x) {
16     return lower_bound(idx.begin(), idx.end(), x) - idx.begin();
17 }
```

3 문자열

3.1 KMP

```
1 vector<int> get_pi (string p) {
2     int m = (int) p.size();
3     vector<int> pi(m);
4     pi[0] = 0;
5     int j = 0;
6     for (int i=1; i<m; i++) {
7         while (j > 0 && p[i] != p[j])
8             j = pi[j-1];
9         if (p[i] == p[j]) {
10             j++;
11             pi[i] = j;
12         }
13     }
14     return pi;
15 }
16
17 vector<int> kmp(string t, string p) {
18     auto pi = get_pi(p);
19     vector<int> result;
20     int n = (int) t.size();
21     int m = (int) p.size();
22     int idx = 0;
23     for (int i=0; i<n+1; i++) {
24         if (idx == m) {
25             result.push_back(i+1-m);
26             idx = pi[idx - 1];
27         }
28         if (i == n) break;
29         while (idx > 0 && t[i] != p[idx])
30             idx = pi[idx - 1];
31         if (t[i] == p[idx])
32             idx++;
33     }
34     return result;
35 }
```

4 그래프

4.1 최단 경로 알고리즘

4.1.1 다익스트라

```
1 const int INF = 987654321;
```

```
2 int max_size = 20001;
3 vector<vector<pair<int, int>>> graph(max_size);
4 priority_queue<pair<int, int>> pq;
5 vector<int> dist(max_size, INF);
6
7 void dijkstra(int start) {
8     pq.push({0, start});
9     dist[start] = 0;
10    while (!pq.empty()) {
11        int d = -pq.top().first;
12        int node = pq.top().second;
13        pq.pop();
14        if (d > dist[node]) continue;
15
16        for (auto p: graph[node]) {
17            int nxt = p.first;
18            int cost = p.second;
19            int dd = d + cost;
20            if (dist[nxt] > dd) {
21                dist[nxt] = dd;
22                pq.push({-dd, nxt});
23            }
24        }
25    }
26 }
```

4.1.2 벨만 포드

```
1 const long long INF = 987654321;
2 int max_size = 501;
3 bool cycle = false;
4 vector<vector<pair<int, long long>>> graph(max_size);
5 vector<long long> dist(max_size, INF);
6
7 void bellman_ford(int start, int n) {
8     dist[start] = 0;
9     for (int i=1; i<=n; i++) {
10        for (int j=1; j<=n; j++) {
11            for (auto &p: graph[j]) {
12                if (dist[j] != INF && dist[p.first] > p.second +
13                    dist[p.first] = p.second + dist[j];
14                    if (i==n) cycle = true;
15            }
16        }
17    }
18 }
19 }
```

4.1.3 플로이드 워셜

```
1 const int INF = 987654321;
2 int max_size = 101;
3 vector <vector <int>> dist(max_size, vector <int> (max_size, INF));
4
5 void floyd_warshall(int n) {
6     for (int i=0; i<n; i++) dist[i][i] = 0;
7     for (int i=0; i<n; i++)
8         for (int start=0; start<n; start++)
9             for (int end=0; end<n; end++)
10                 dist[start][end] = min(dist[start][end], dist[start
11 ][i] + dist[i][end]);
12 }
```

4.2 강한 결합 요소

```
1 #include <iostream>
2 #include <vector>
3 #include <set>
4 #include <stack>
5 #include <algorithm>
6 using namespace std;
7
8 const int MAX = 20001;
9
10 vector <vector <int>> graph(MAX);
11 vector <vector <int>> graph_rev(MAX);
12 vector <vector <int>> scc_list;
13 stack <int> s;
14 int T, V, E;
15 char visited[MAX];
16 int scc_idx[MAX];
17
18 void dfs(int node) {
19     for (int nxt : graph[node]) {
20         if (!visited[nxt]) {
21             visited[nxt] = 1;
22             dfs(nxt);
23         }
24     }
25     s.push(node);
26 }
27
28 void dfs_rev(vector <int> &scc, int node) {
29     scc.push_back(node);
30     for (int nxt : graph_rev[node]) {
31         if (!visited[nxt]) {
32             visited[nxt] = 1;
33             dfs_rev(scc, nxt);
34         }
35     }
36 }
```

```
33         visited[nxt] = 1;
34         dfs_rev(scc, nxt);
35     }
36 }
37
38 int main() {
39     ios_base::sync_with_stdio(false);
40     cin.tie(NULL);
41     cout.tie(NULL);
42
43     cin >> V >> E;
44     int A, B;
45     int notA = 0; int notB = 0;
46     for (int i=0; i<E; i++) {
47         notA = 0;
48         notB = 0;
49         cin >> A >> B;
50         if (A < 0) {
51             notA = 1;
52             A *= -1;
53         }
54         if (B < 0) {
55             notB = 1;
56             B *= -1;
57         }
58         int rev_A = ((notA + 1) % 2) * 10000 + A;
59         int rev_B = ((notB + 1) % 2) * 10000 + B;
60         A = notA * 10000 + A;
61         B = notB * 10000 + B;
62         // cout << "(" << rev_A << "," << B << ")" (" << rev_B << ","
63         << A << ")\n";
64         graph[rev_A].push_back(B);
65         graph_rev[B].push_back(rev_A);
66         graph[rev_B].push_back(A);
67         graph_rev[A].push_back(rev_B);
68     }
69
70     for (int i=1; i<V+1; i++) {
71         if (!visited[i]) dfs(i);
72     }
73
74     for (int i=0; i<MAX; i++) visited[i] = 0;
75     while (!s.empty()) {
76         int node = s.top();
77         vector <int> scc;
78         s.pop();
79         if (!visited[node]) {
80             visited[node] = 1;
81             dfs_rev(scc, node);
82         }
83     }
84 }
```

```

82         sort(scc.begin(), scc.end());
83         scc_list.push_back(scc);
84     }
85 }
86
87 int scc_n = scc_list.size();
88 vector<int> indegree (scc_n);
89
90 int idx = 1;
91 for (auto scc : scc_list) {
92     for (int node : scc) {
93         scc_idx[node] = idx;
94     }
95     idx++;
96 }
97
98 int result = 1;
99 for (int i=1; i<=V; i++) {
100     if (scc_idx[i] == scc_idx[10000 + i])
101         result = 0;
102 }
103
104 cout << result << '\n';
105
106 return 0;
107 }

```

4.2.1 2-SAT (가능 여부 판별)

```

1 #include <iostream>
2 #include <vector>
3 #include <set>
4 #include <stack>
5 #include <algorithm>
6 using namespace std;
7
8 const int MAX = 20001;
9
10 vector<vector<int>> graph(MAX);
11 vector<vector<int>> graph_rev(MAX);
12 vector<vector<int>> scc_list;
13 stack<int> s;
14 int T, V, E;
15 char visited[MAX];
16 int scc_idx[MAX];
17
18 void dfs(int node) {
19     for (int nxt : graph[node]) {
20

```

```

21         if (!visited[nxt]) {
22             visited[nxt] = 1;
23             dfs(nxt);
24         }
25     }
26     s.push(node);
27 }
28
29 void dfs_rev(vector<int> &scc, int node) {
30     scc.push_back(node);
31     for (int nxt : graph_rev[node]) {
32         if (!visited[nxt]) {
33             visited[nxt] = 1;
34             dfs_rev(scc, nxt);
35         }
36     }
37 }
38
39 int main() {
40     ios_base::sync_with_stdio(false);
41     cin.tie(NULL);
42     cout.tie(NULL);
43
44     cin >> V >> E;
45     int A, B;
46     int notA = 0; int notB = 0;
47     for (int i=0; i<E; i++) {
48         notA = 0;
49         notB = 0;
50         cin >> A >> B;
51         if (A < 0) {
52             notA = 1;
53             A *= -1;
54         }
55         if (B < 0) {
56             notB = 1;
57             B *= -1;
58         }
59         int rev_A = ((notA + 1) % 2) * 10000 + A;
60         int rev_B = ((notB + 1) % 2) * 10000 + B;
61         A = notA * 10000 + A;
62         B = notB * 10000 + B;
63         // cout << "(" << rev_A << "," << B << ")" (" << rev_B << ","
64         << A << ")\n";
65         graph[rev_A].push_back(B);
66         graph_rev[B].push_back(rev_A);
67         graph[rev_B].push_back(A);
68         graph_rev[A].push_back(rev_B);
69     }

```

```

70     for (int i=1; i<V+1; i++) {
71         if (!visited[i]) dfs(i);
72     }
73
74     for (int i=0; i<MAX; i++) visited[i] = 0;
75     while (!s.empty()) {
76         int node = s.top();
77         vector<int> scc;
78         s.pop();
79         if (!visited[node]) {
80             visited[node] = 1;
81             dfs_rev(scc, node);
82             sort(scc.begin(), scc.end());
83             scc_list.push_back(scc);
84         }
85     }
86
87     int scc_n = scc_list.size();
88     vector<int> indegree (scc_n);
89
90     int idx = 1;
91     for (auto scc : scc_list) {
92         for (int node : scc) {
93             scc_idx[node] = idx;
94         }
95         idx++;
96     }
97
98     int result = 1;
99     for (int i=1; i<=V; i++) {
100         if (scc_idx[i] == scc_idx[10000 + i])
101             result = 0;
102     }
103
104     cout << result << '\n';
105
106     return 0;
107 }

```

4.3 최대 유량

4.3.1 에드몬드 카프

```

1  #include <iostream>
2  #include <vector>
3  #include <queue>
4  #include <unordered_set>
5  #include <algorithm>
6  using namespace std;
7
8  const int SIZE = 52;
9  const int INF = 1e9;
10
11 int n;
12 vector<unordered_set<int>> graph(SIZE);
13 vector<vector<int>> c(SIZE, vector<int>(SIZE, 0));
14 vector<vector<int>> f(SIZE, vector<int>(SIZE, 0));
15
16 const int source = 0;
17 const int sink = SIZE - 1;
18
19 int edmonds_karp(int source, int sink) {
20     int max_flow = 0;
21     while (true) {
22
23         int visited[SIZE];
24         for (int i=0; i<SIZE; i++) visited[i] = -1;
25
26         queue<int> q;
27         q.push(source);
28         while (!q.empty()) {
29             int node = q.front();
30             q.pop();
31             for (auto &nxt: graph[node]) {
32                 if (c[node][nxt] - f[node][nxt] > 0 && visited[nxt]
33 == -1) {
34                     q.push(nxt);
35                     visited[nxt] = node;
36                     if (nxt == sink) break;
37                 }
38             }
39         }
40         if (visited[sink] == -1) break;
41         int curr_flow = INF;
42         int node = sink;
43         while (node != source) {
44             curr_flow = min(curr_flow, c[visited[node]][node] - f[
visited[node]][node]);

```



```

45     node = visited[node];
46 }
47 node = sink;
48 while (node != source) {
49     f[visited[node]][node] += curr_flow;
50     f[node][visited[node]] -= curr_flow;
51     node = visited[node];
52 }
53 max_flow += curr_flow;
54
55 }
56 return max_flow;
57 }

```

4.3.2 이분 매칭

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  const int SIZE = 1001;
7  const int INF = 1e9;
8
9  int n, m;
10 vector <vector <int>> graph(SIZE);
11 bool visited[SIZE];
12 int work[SIZE];
13
14 bool dm_dfs (int start) {
15     visited[start] = 1;
16     for (auto &i : graph[start]) {
17         if (work[i] == 0 || (!visited[work[i]] && dm_dfs(work[i])))
18         {
19             work[i] = start;
20             return 1;
21         }
22     }
23     return 0;
24 }
25
26 int bipartite_matching(int n) {
27     int result = 0;
28     for (int i=1; i<=n; i++) {
29         for (int j=0; j<SIZE; j++) visited[j] = 0;
30         if (dm_dfs(i)) result++;
31     }
32     return result;
33 }

```

5 트리

5.1 최소 신장 트리

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4  using namespace std;
5
6  const int SIZE = 10001;
7
8  vector <vector <int>> edge;
9  vector <int> parent(SIZE); // must be initialized by parent[i] = i;
10 vector <int> p_rank(SIZE, 1);
11
12 int find(int x) {
13     if (x == parent[x]) return x;
14     return parent[x] = find(parent[x]);
15 }
16
17 void merge(int x, int y) {
18     x = find(x);
19     y = find(y);
20     if (x == y) return;
21     if (p_rank[x] > p_rank[y]) swap(x, y);
22     if (p_rank[x] == p_rank[y]) p_rank[y]++;
23     parent[x] = y;
24 }
25
26
27 int main() {
28     int v, e;
29     cin.tie(NULL);
30     cout.tie(NULL);
31     ios::sync_with_stdio(false);
32
33     for (int i=0; i<SIZE; i++) parent[i] = i;
34
35     cin >> v >> e;
36     for (int i=0; i<e; i++) {
37         int a, b, c;
38         cin >> a >> b >> c;
39         edge.push_back({-c, a, b});
40     }
41     sort(edge.begin(), edge.end());
42     int result = 0;
43     while (!edge.empty()) {
44         auto ee = edge.back();
45         edge.pop_back();
46         int c = -ee[0];

```

```

47     int a = ee[1];
48     int b = ee[2];
49     if (find(a) != find(b)) {
50         merge(a, b);
51         result += c;
52     }
53 }
54
55 cout << result << endl;
56
57
58 return 0;
59 }

```

5.2 세그먼트 트리 (느린 갱신)

```

1  const int tree_size = 2097152;
2  long long seg_tree[tree_size];
3  long long lazy[tree_size];
4  long long arr[tree_size / 2];
5
6
7  long long generate(int node, int start, int end) {
8      if (start == end) {
9          return seg_tree[node] = arr[start];
10     }
11     int mid = (start + end) / 2;
12     return seg_tree[node] = generate(node * 2, start, mid) +
13         generate(node * 2 + 1, mid + 1, end);
14 }
15
16 void update_lazy(int node, int start, int end) {
17     if (lazy[node] != 0) {
18         seg_tree[node] += (end - start + 1) * lazy[node];
19         if (start != end) {
20             lazy[node * 2] += lazy[node];
21             lazy[node * 2 + 1] += lazy[node];
22         }
23         lazy[node] = 0;
24     }
25 }
26
27
28 void update_range(int node, int start, int end, int left, int right,
29     long long diff) {
30     update_lazy(node, start, end);
31     if (left > end || right < start) return;
32     if (left <= start && end <= right) {

```

```

32         seg_tree[node] += (end - start + 1) * diff;
33         if (start != end) {
34             lazy[node * 2] += diff;
35             lazy[node * 2 + 1] += diff;
36         }
37         return;
38     }
39     int mid = (start + end) / 2;
40     update_range(node * 2, start, mid, left, right, diff);
41     update_range(node * 2 + 1, mid + 1, end, left, right, diff);
42     seg_tree[node] = seg_tree[node * 2] + seg_tree[node * 2 + 1];
43 }
44
45
46 long long sum(int node, int start, int end, int left, int right) {
47     update_lazy(node, start, end);
48     if (left > end || right < start) return 0;
49     if (left <= start && end <= right) return seg_tree[node];
50
51     int mid = (start + end) / 2;
52     return sum(node * 2, start, mid, left, right) + sum(node * 2 +
53         1, mid + 1, end, left, right);
54 }

```

5.3 최소 공통 조상

```
1 #include <iostream>
2 #include <cmath>
3 #include <vector>
4 #include <queue>
5 #include <algorithm>
6 using namespace std;
7
8 const int max_n = 100005;
9 const int max_exp = 17;
10 vector <int> log_2(max_n);
11 vector <vector <int>> edge(max_n);
12 vector <vector <int>> parent(max_n, vector <int> (max_exp));
13 vector <int> depth(max_n);
14
15 void initialize(int n) {
16     int power = 0;
17     while (pow(2, power) < max_n+1) {
18         for (int i=(int) pow(2, power); i<min(max_n+1, (int) pow(2,
19             power+1)); i++)
20             log_2[i] = power;
21         power++;
22     }
23
24     for (int i=0; i<max_n; i++)
25         parent[i][0] = -1;
26
27     parent[0][0] = 0;
28     parent[1][0] = 0;
29     depth[1] = 1;
30
31     queue <int> q;
32     q.push(1);
33     while (!q.empty()) {
34         int node = q.front();
35         q.pop();
36         for (int &child : edge[node]) {
37             if (parent[child][0] != -1) continue;
38             depth[child] = depth[node] + 1;
39             parent[child][0] = node;
40             q.push(child);
41         }
42     }
43
44     for (int k=1; k<max_exp; k++) {
45         for (int i=1; i<n+1; i++) {
46             parent[i][k] = parent[ parent[i][k-1] ][k-1];
47         }
48     }
```

```
48 }
49
50 int get_lca(int u, int v) {
51     if (depth[u] > depth[v]) swap(u, v);
52     int gap = depth[v] - depth[u];
53     int m = depth[v];
54
55     for (int i=log_2[m]; i>=0; i--) {
56         if (gap >= pow(2, i)) {
57             v = parent[v][i];
58             gap = depth[v] - depth[u];
59         }
60     }
61     if (u == v) return u;
62     while (u != v) {
63         int m = depth[v];
64         bool terminate = false;
65         for (int i=log_2[m]; i>=0; i--) {
66             if (parent[u][i] != parent[v][i]) {
67                 u = parent[u][i];
68                 v = parent[v][i];
69                 terminate = true;
70                 break;
71             }
72         }
73         if (!terminate) {
74             u = parent[u][0];
75             v = parent[v][0];
76             break;
77         }
78     }
79     return u;
80 }
```

6 기타

6.1 서로소 집합

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 const int SIZE = 100001;
7
8 vector <int> parent(SIZE); // must be initialized by parent[i] = i;
9 vector <int> rank(SIZE, 1);
10
11 int find(int x) {
12     if (x == parent[x]) return x;
13     return parent[x] = find(parent[x]);
14 }
15
16 void merge(int x, int y) {
17     x = find(x);
18     y = find(y);
19     if (x == y) return;
20     if (rank[x] > rank[y]) swap(x, y);
21     if (rank[x] == rank[y]) rank[y]++;
22     parent[x] = y;
23 }
```

6.2 DP 응용

6.2.1 배낭 문제

```
1 #include <iostream>
2 #include <vector>
3 #include <algorithm>
4 using namespace std;
5
6 const int SIZE = 100001;
7 vector <int> dp(SIZE, -1);
8
9
10 int main() {
11     int n, k;
12     cin.tie(NULL);
13     cout.tie(NULL);
14     ios::sync_with_stdio(false);
15
16     dp[0] = 0;
```

```
17     cin >> n >> k;
18     for (int i=0; i<n; i++) {
19         int w, v;
20         cin >> w >> v;
21         for (int j=k; j>=w; j--) {
22             if (dp[j-w] == -1) continue;
23             dp[j] = max(dp[j], v + dp[j-w]);
24         }
25     }
26
27     int result = 0;
28     for (int i=0; i<SIZE; i++) result = max(result, dp[i]);
29     cout << result << endl;
30
31     return 0;
32 }
```

7 템플릿

7.1 빠른 입출력

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5     ios_base::sync_with_stdio(false);
6     cin.tie(NULL);
7     return 0;
8 }
```

7.2 string to XXX 예제

```
1 // Reference: https://blockdmask.tistory.com/333
2
3 #include <iostream>
4 #include <string>
5 using namespace std;
6
7 int main() {
8     ios_base::sync_with_stdio(false);
9     cin.tie(NULL);
10
11     string str_i = "22";
12     string str_li = "2144967290";
13     string str_f = "3.4";
14     string str_d = "2.11";
15
16     int i      = stoi(str_i);
17     long int li = stol(str_li);
18     float f     = stof(str_f);
19     double d    = stod(str_d);
20
21     //C++ cout
22     cout << "stoi : " << i      << endl;
23     cout << "stol : " << li     << endl;
24     cout << "stof : " << f      << endl;
25     cout << "stod : " << d      << endl;
26 }
```

7.3 여러 input 방법

```
1 // Reference: https://jhnyang.tistory.com/321
2 //           https://dbstndi63i6.tistory.com/33
```

```
3
4 #include <iostream>
5 #include <string>
6 using namespace std;
7
8 int main() {
9     ios_base::sync_with_stdio(false);
10    cin.tie(NULL);
11
12    string name;
13    getline(cin, name);
14    cout << name << endl;
15
16    char c;
17    c = getchar();
18    cout << c << endl;
19
20 }
```

7.4 기타 string 관련 함수

```
1 // Reference: https://psychoria.tistory.com/773
2 //           https://blockdmask.tistory.com/338
3
4 #include <iostream>
5 #include <string>
6 using namespace std;
7
8 int main()
9 {
10     string numbers = "0123456789";
11
12     cout << numbers.at(6) << endl; //6
13     cout << numbers.back() << endl; //9
14     cout << numbers.size() << endl; //10
15
16
17     string resize = "987654321";
18
19     resize.resize(5);
20     cout << resize << endl; //98765
21     resize.resize(10, '0'); //only char type
22     cout << resize << endl; //9876500000
23
24
25     string full = numbers.substr();
26     string sub = numbers.substr(3, 5);
27
28     cout << "Full String: " << full << endl; //0123456789
```

```

29     cout << "Sub String: " << sub << endl; //34567
30
31
32     string replace = "ABCDE";
33     replace.replace(3,0, "ZZZ");
34     cout << replace << endl; //ABCZZZDE
35     replace.replace(0,100, "New Word");
36     cout << replace << endl; //New Word
37
38
39     string find = "abcdbc";
40     cout << find.find("bc") << endl; //1, return first matched index
41
42     for (string::iterator iter = find.begin(); iter != find.end();
43 ++iter){
44         cout << *iter << endl;
45     }
46
47     string path = "file.txt.png";
48     size_t pos = path.rfind('.'); //return last matched index
49
50     string filename = path.substr(0, pos);
51     string extension = path.substr(pos + 1);
52
53     cout << "Filename: " << filename << endl;
54     cout << "Extension: " << extension << endl;
55 }

```

```

19
20 int main() {
21     priority_queue<Student, vector<Student>, cmp> pq;
22
23
24     pq.push(Student(3, 100, 50));
25     pq.push(Student(1, 60, 50));
26     pq.push(Student(2, 80, 50));
27     pq.push(Student(4, 90, 50));
28     pq.push(Student(5, 70, 40));
29
30     while (!pq.empty()) {
31         Student ts = pq.top(); pq.pop();
32         cout << "(id, math, english) : " << ts.id << ' ' << ts.math
33 << ' ' << ts.eng << '\n';
34     }
35
36     return 0;
37 }

```

7.5 구조체, 연산자 오버로딩을 활용한 우선순위 큐

```

1 // Reference: https://kbj96.tistory.com/15
2
3 #include <iostream>
4 #include <queue>
5 #include <functional> // greater
6 using namespace std;
7
8 struct Student {
9     int id;
10    int math, eng;
11    Student(int num, int m, int e) : id(num), math(m), eng(e) {}
12 };
13
14 struct cmp {
15     bool operator()(Student a, Student b) {
16         return a.id < b.id;
17     }
18 };

```