DP

NOCYCLE 길찾기

```
//사이클 없을 때 DP를 이용해서 빠르게 탐색 가능
//1937번 욕심쟁이 판다 1949번 우수마을

#include <cstdio>
#include <vector>
using namespace std;
int n, v1, v2, ans, dp[10002][2], w[10002], visited[10002];
vector<vector<int> > e;

void find(int x){
    for(int i = 0; i < e[x].size(); i++){
        int nx = e[x][i];
        if(visited[nx])continue;
        visited[nx] = 1;
        find(nx);
        dp[x][0] += dp[nx][1];
        dp[x][1] += max(dp[nx][0], dp[nx][1]);
    }
    dp[x][0] += w[x];
}

int main(){
    scanf("%d", &n);
    e = vector<vector<int> >(n + 1);
    for(int i = 1; i <= n; i++)scanf("%d", &w[i]);
    for(int i = 0; i < n - 1; i++){
        scanf("%d %d", &v1, &v2);
        e[v1].push_back(v2);
        e[v2].push_back(v1);
    }
    visited[1] = 1;
    find(1);
    ans = max(dp[1][0], dp[1][1]);
    printf("%d", ans);
}
```

LIS

```
#include <cstdio>
#include <vector>
#include <algorithm>
using namespace std;
int ans, n, t, arr[201];
int vector<int> v;
main(){
    scanf("%d", &n);
    for(int i = 1; i <= n; i++){
```

```
        scanf("%d", &t);
        arr[t] = i;
    }
    for(int i = 1; i <= n; i++){
        int r = arr[i];
        if(!ans || v[ans-1] < r)v.push_back(r), ans++;
        else v[upper_bound(v.begin(), v.end(), r)-v.begin()] = r;
    }
    printf("%d", n - ans);
}
```

에레토스테네스의 체

```
#include <cstdio>
#include <vector>

using namespace std;

int main(){
    int n = 0, k = 0, m = 0;
    int check[10000] = {0,};
    scanf("%d", &n);
    vector<int> v;
    while(n--){
        scanf("%d", &k);
        if(k > m) m = k;
        v.push_back(k);
    }
    for(int i = 2; i <= m; i++){
        if(check[i] == 0){
            for(int j = i*2; j <= m; j+=i)check[j] = 1;
        }
    }

    for(int i = 0; i < v.size(); i++){
        for(int j = v[i]/2 ; j < v[i]; j++){
            if(!check[j] && !check[v[i]-j]) {
                printf("%d %d\n", v[i]-j, j);
                break;
            }
        }
    }

}
```

BinarySearch

```
#include <cstdio>
int main(){
    long long x, y, z, ans = -1;
    scanf("%lld %lld", &x, &y);
    z = y*100 / x + 1;
    if(z == 100 || z == 101){
        printf("-1");
        return 0;
    }
    long long l = 1, r = 1000000000;
    while(l <= r){
        long long mid = (l + r) / 2;
        long long val = (y + mid)*100 / (x + mid);
        //현재는 될 수 있는 값 중에 가장 작은 값을 찾는 중
        if(z <= val) r = mid - 1, ans = mid;
        else l = mid + 1;
    }
    printf("%lld", ans);
}
```

Dijkstra

```
#include <cstdio>
#include <vector>
#include <queue>
using namespace std;
#define ll long long
#define pll pair<ll, ll>

ll ans, cnt, n, m, v1, v2, w, src, dst, visited[1001];
vector<vector<pll> > e;
int main(){
    scanf("%lld %lld", &n, &m);
    e = vector<vector<pll> >(n+1);
    for(int i = 0; i < m; i++){
        scanf("%lld %lld %lld", &v1, &v2, &w);
        e[v1].push_back(pll(-w, v2));
    }
    scanf("%lld %lld", &src, &dst);
    priority_queue<pll> pq;
    pq.push(pll(0, src));
    while(!pq.empty()){
        pll top = pq.top(); pq.pop();
        int tv = top.first, tx = top.second;
        visited[tx] = 1, cnt++;
        if(tx == dst){
            ans = tv;
            break;
        }
```

```
        for(int i = 0; i < e[tx].size(); i++){
            pll next = e[tx][i];
            int nw = next.first, nx = next.second;
            if(visited[nx])continue;
            pq.push(pll(tv + nw, nx));
        }
    }
    printf("%lld", -ans);

}
```

Floyd

```
#include <cstdio>
#include <vector>

using namespace std;

int n, m, x, w, v1, v2, M, d[1001][1001];

int main(){
    scanf("%d %d %d", &n, &m, &x);
    for(int i = 1; i < n + 1; i++){
        for(int j = 1; j < n + 1; j++)d[i][j] = 1000000;
        d[i][i] = 0;
    }
    for(int i = 0; i < m; i++){
        scanf("%d %d %d", &v1, &v2, &w);
        d[v1][v2] = w;
    }
    for(int k = 1; k < n + 1; k++){
        for(int i = 1; i < n + 1; i++){
            for(int j = 1; j < n + 1; j++){
                if(d[i][j] > d[i][k] + d[k][j])d[i][j] = d[i][k] + d[k][j];
            }
        }
    }
    for(int i = 1; i < n + 1; i++){
        if(M < d[i][x] + d[x][i])M = d[i][x] + d[x][i];
    }
    printf("%d", M);
}
```

KMP

```
#include <cstdio>
#include <string.h>

char t[1000002], p[1000002];
int ans[1000002], pi[1000002], tn, pn, an;

void make_pi(){
    int j = 0;
    for(int i = 1; i < pn; i++){
        while(j > 0 && p[i] != p[j]) j = pi[j-1];
        if(p[i] == p[j]) pi[i] = ++j;
    }
}

void do_KMP(){
    int j = 0;
    for(int i = 0; i < tn; i++){
        while(t[i] != p[j] && j > 0)j = pi[j - 1];
        if(t[i] == p[j])j++;
        if(j == pn - 1)ans[an++] = i - pn + 3;
    }
}

int main(){
    fgets(t, 1000002, stdin);
    fgets(p, 1000002, stdin);
    tn = strlen(t), pn = strlen(p);
    make_pi();
    do_KMP();
    printf("%d\n", an);
    for(int i = 0; i < an; i++)printf("%d ", ans[i]);
}
```

LCA

```
#include <queue>
#include <vector>
#include <cstdio>
#define INF 1000000000

using namespace std;

vector<vector<int> > ll;
int ac[17][50020], depth[50020];
int n, m, v, w, u;

int find_LCA(int chd, int par){
    int diff = depth[chd] - depth[par];
    for(int i = 16; i >= 0; i--){
        if((1<<i) <= diff){
```

```
            chd = ac[i][chd];
            diff -= (1<<i);
        }
    }
    if(chd == par) return chd;
    for(int i = 16; i >= 0; i--){
        if(ac[i][chd] != ac[i][par]) chd = ac[i][chd], par = ac[i][par];
    }
    return ac[0][chd];
}

void build_tree(){
    queue<int> q;
    depth[1] = 1, ac[0][1] = 1;
    q.push(1);
    while(!q.empty()){
        u = q.front(); q.pop();
        for(int i = 0; i < ll[u].size(); i++){
            v = ll[u][i];
            if(depth[v] == 0){
                depth[v] = depth[u] + 1;
                ac[0][v] = u;
                q.push(v);
            }
        }
    }
}
void find_parent(){
    for(int i = 1; i < 17; i++){
        for(int j = 1; j < n + 1; j++)ac[i][j] = ac[i-1][ac[i-1][j]];
    }
}

int main(){
    scanf("%d", &n);
    ll = vector<vector<int> >(n + 1);
    for(int i = 1; i < n; i++){
        scanf("%d %d", &v, &u);
        ll[v].push_back(u);
        ll[u].push_back(v);
    }
    build_tree();
    find_parent();

    scanf("%d", &m);
    while(m--){
        scanf("%d %d", &v, &u);
        printf("%d\n", depth[v] < depth[u] ? find_LCA(v, u) : find_LCA(u, v));
    }
}
```

Prim

```
#include <cstdio>
#include <queue>
#include <vector>
using namespace std;
#define pii pair<int, int>
#define pip pair<int, pii>
int ans, cnt, n, m, a, b, c, par[100001];

int find(int i){
    if(par[i] == i)return i;
    return par[i] = find(par[i]);
}

int main(){
    scanf("%d %d", &n, &m);
    for(int i = 1; i <= n; i++)par[i] = i;
    priority_queue<pip> pq;
    for(int i = 0; i < m; i++){
        scanf("%d %d %d", &a, &b, &c);
        pq.push(pip(-c, pii(a, b)));
    }
    while(!pq.empty() && cnt != n - 2){
        pip top = pq.top(); pq.pop();
        int tv = top.first, tx = top.second.first, ty = top.second.second;
        int px = find(tx), py = find(ty);
        if(px == py)continue;
        ans += -tv, cnt++;
        par[px] = py;
    }
    printf("%d", ans);
}
```

Segment Tree

```
//구간합, 구간 최소,최대값 찾는데 사용가능
#include <cstdio>
#include <cmath>
#include <vector>
using namespace std;
long long init(vector<long long> &a, vector<long long> &tree, int node, int start,
int end) {
    if (start == end) {
        return tree[node] = a[start];
    } else {
        return tree[node] = init(a, tree, node*2, start, (start+end)/2) + init(a, t
ree, node*2+1, (start+end)/2+1, end);
    }
}
void update(vector<long long> &tree, int node, int start, int end, int index, long
long diff) {
```

```
    if (index < start || index > end) return;
    tree[node] = tree[node] + diff;
    if (start != end) {
        update(tree,node*2, start, (start+end)/2, index, diff);
        update(tree,node*2+1, (start+end)/2+1, end, index, diff);
    }
}
long long sum(vector<long long> &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];
    }
    return sum(tree, node*2, start, (start+end)/2, left, right) + sum(tree, node*2+
1, (start+end)/2+1, end, left, right);
}
int main() {
    int n, m, k;
    scanf("%d %d %d",&n,&m,&k);
    vector<long long> a(n);
    int h = (int)ceil(log2(n));
    int tree_size = (1 << (h+1));
    vector<long long> tree(tree_size);
    m += k;
    for (int i=0; i<n; i++) {
        scanf("%lld",&a[i]);
    }
    init(a, tree, 1, 0, n-1);
    while (m--) {
        int t1,t2,t3;
        scanf("%d",&t1);
        if (t1 == 1) {
            int t2;
            long long t3;
            scanf("%d %lld",&t2,&t3);
            t2-=1;
            long long diff = t3-a[t2];
            a[t2] = t3;
            update(tree, 1, 0, n-1, t2, diff);
        } else if (t1 == 2) {
            int t2,t3;
            scanf("%d %d",&t2,&t3);
            printf("%lld\n",sum(tree, 1, 0, n-1, t2-1, t3-1));
        }
    }
    return 0;
}
```

Topological Sort

```
//구간합, 구간 최소,최대값 찾는데 사용가능
#include <cstdio>
#include <cmath>
#include <vector>
using namespace std;
long long init(vector<long long> &a, vector<long long> &tree, int node, int start,
int end) {
    if (start == end) {
        return tree[node] = a[start];
    } else {
        return tree[node] = init(a, tree, node*2, start, (start+end)/2) + init(a, t
ree, node*2+1, (start+end)/2+1, end);
    }
}
void update(vector<long long> &tree, int node, int start, int end, int index, long
long diff) {
    if (index < start || index > end) return;
    tree[node] = tree[node] + diff;
    if (start != end) {
        update(tree,node*2, start, (start+end)/2, index, diff);
        update(tree,node*2+1, (start+end)/2+1, end, index, diff);
    }
}
long long sum(vector<long long> &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];
    }
    return sum(tree, node*2, start, (start+end)/2, left, right) + sum(tree, node*2+
1, (start+end)/2+1, end, left, right);
}
int main() {
    int n, m, k;
    scanf("%d %d %d",&n,&m,&k);
    vector<long long> a(n);
    int h = (int)ceil(log2(n));
    int tree_size = (1 << (h+1));
    vector<long long> tree(tree_size);
    m += k;
    for (int i=0; i<n; i++) {
        scanf("%lld",&a[i]);
    }
    init(a, tree, 1, 0, n-1);
    while (m--) {
        int t1,t2,t3;
        scanf("%d",&t1);
```

```
        if (t1 == 1) {
            int t2;
            long long t3;
            scanf("%d %lld",&t2,&t3);
            t2-=1;
            long long diff = t3-a[t2];
            a[t2] = t3;
            update(tree, 1, 0, n-1, t2, diff);
        } else if (t1 == 2) {
            int t2,t3;
            scanf("%d %d",&t2,&t3);
            printf("%lld\n",sum(tree, 1, 0, n-1, t2-1, t3-1));
        }
    }
    return 0;
}
```

STL 정리

lower_bound

- 원하는 키값이 **없으면** key 값보다 큰 가장 작은 정수 값
- **있으면** 키값을 가지는 애중에 맨앞 iterator 반환
- ex) arr[6] = {1, 2, 3, 3, 4, 5} 일때, lower_bound 로 3 을 찾으면 arr[2]의 iterator 반환

upper_bound

- 오름차순으로 정렬되어 있어야함
- 원하는 키 값을 초과하는 가장 첫번째 원소의 위치를 반한
- 키 값이 있으면 있을 시에는 키 값을 가지는 애들 바로 다음 iterator 를 반환

트리

Tree 의 지름

임의의 점에서 가장 먼 노드(A)를 찾고 그 노드(A)에서 가장 먼 노드까지의 거리가 트리의 지름입니다.

이진 검색 트리

이진 검색 트리는 root node 의 left subtree 에는 항상 root 보다 작은 값들이 위치하고 right 에는 root 보다 큰 값들이 위치한다.

이진 검색 트리의 후위 순회(Postorder Traversal)

- 후위 순회의 root 는 맨 끝에 있는 값이다.
- 이진 검색 트리에서 left subtree 는 다 root 보다 작기 때문에 root 보다 커지는 첫 지점(upper_bound)를 찾으면 left 와 right 를 나눌 수있다.
- 이를 바탕으로 left - right - root 순으로 재귀를 진행하면 된다.

이진 트리의 중위 순회(Inorder Traversal)

- 중위 순회의 결과는 트리를 중력에 의해서 다 떨겼을 때의 결과와 동일하다.