Team Note of jwpassion1

Yoo Jaewon (jwpassion1)

Compiled on January 15, 2026

Contents

1 Data Structure

1.1 PBDS

Time Complexity: $\mathcal{O}(logN)$

```
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace __gnu_pbds;

//중복 비허용 int형 pbds
#define ordered_set tree<int, null_type, less<>, rb_tree_tag,
tree_order_statistics_node_update>
//중복 허용 int형 pbds
#define ordered_set tree<int, null_type, less_equal<>,
rb_tree_tag, tree_order_statistics_node_update>

ordered_set OS; //pbds OS 선언
OS.insert(val); // OS에 val 삽입
OS.erase(val); // OS에서 val 삭제 (중복 비허용일때)
OS.order_of_key(val); // OS에서 val보다 작은 원소의 개수 리턴
OS.find_by_order(K); // OS[K] 리턴 (0-based-index)

void erase(ordered_set &OS, int val){ // OS에서 val 삭제 (중복 허용일
때)
    auto it = OS.find_by_order(OS.order_of_key(val));
    if (*it == val) OS.erase(it);
}
```

2 Graph & Flow

2.1 이분 매칭(Kuh, 느림)

Time Complexity: $\mathcal{O}(N \log N)$

```
vector<int> graph[total_graph_sz];
int from[right_graph_sz], visited[left_graph_sz];
int visitco; //매번 dfs 돌때마다 값 1씩 증가

bool dfs(int node){
    if (visited[node] == visitco) return false;
    visited[node] = visitco;
    for (int i : graph[node]){
        if (from[i] == -1){
            from[i] = node;
            return true;
        }
        if (dfs(from[i])){
            from[i] = node;
            return true;
        }
    }
    return false;
}
```

3 Fast Fourier Transform

3.1 Fast Fourier Transform

Time Complexity: $\mathcal{O}(N \log N)$

```
typedef complex<long double> cld;

// 곱할 두 벡터를 isfft true로 하고 각각 fft 돌린뒤에 자릿수별로 곱해주고
isfft false로 하고 fft 돌리기
vector<cld> fft(vector<cld>& a, bool isfft){
    int n = 1, isf = isfft ? 2 : -2;
    while (n < a.size()) n <<= 1;
    vector<cld> ret(n);

    for (int i = 0; i < n; i++){
        int tmp = 0, j = 1, rj = n >> 1;
        for (; j < n; j <<= 1, rj >>= 1){
            if (i & j) tmp |= rj;
        }
        if (i < a.size()) ret[tmp] = a[i];
    }

    for (int i = 2; i <= n; i <<= 1){
        cld w = exp(cld(0, isf * M_PI / i));
        int ord = i >> 1;
        for (int j = 0; j < n; j += i){
            cld wn = cld(1, 0);
            for (int k = j; k < (j | ord); k++, wn *= w){
                cld tmp = ret[k | ord];
                ret[k | ord] = ret[k] - wn * tmp;
                ret[k] += wn * tmp;
            }
        }
    }

    if (!isfft) for (cld& i : ret) i /= n;

    return ret;
}
```

4 ETC

4.1 Template

```
#pragma GCC optimize("Ofast")
#pragma GCC optimize("unroll-loops")
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
typedef pair<int, int> pii;

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


}
```