ICPC template

目 次

1 C++ Preparation **2**

1.1 C++ Compiler 2

1.2 C++ Execution 2

1.3 C++ Template 2

2 データ構造 **2**

2.1 二分探索 2

2.2 Union-Find 2

2.3 BIT 2

1 C++ Preparation

1.1 C++ Compiler

```
g++ -std=c++17 test.cpp -o
```

1.2 C++ Execution

```
./test.out <input> output
```

1.3 C++ Template

```
#include <bits/stdc++.h>
using namespace std;
using ll = long long;
using ull = unsigned long long;
using Graph = vector<vector<int>>;
constexpr int INF = 1e9;
constexpr ll LLINF = 4e18;
#define for_(i,a,b) for(int i=(a);i<(b);++i)
#define rep(i, n) for_(i, 0, n)
#define all(a) (a).begin(), (a).end()
#define rall(a) (a).rbegin(), (a).rend()

//4方向
int dx[4] = {1, 0, -1, 0};
int dy[4] = {0, -1, 0, 1};

//8方向
int ddx[8] = {1,1,1,0,0,-1,-1,-1};
int ddy[8] = {1,0,-1,1,-1,1,0,-1};

int main() {

    return 0;
}
```

2 データ構造

2.1 二分探索

```
vector<int> a = { 1,4,4,7,7,8,8,11,13,19};
// lower_bound:key以上の値が初めて現れる位置
auto iter = lower_bound(all(a),4);
// key以上の最小の値を出力
cout << *iter << endl; // 4
// key以上の最小の値が初めて現れる位置を出力
cout << a.begin() - iter << endl; // 1

// upper_bound:
    keyより大きい値が初めて現れる位置
auto iter1 = upper_bound(all(a), 4);
// keyより大きい最小の値を出力
cout << *iter1 << endl; // 7
// keyより大きい最小の値が初出する位置を出力
cout << a.begin() - iter1 << endl; // 3
```

2.2 Union-Find

```
// Union-Find
// グリッドでUFを使う時,(x,y)に対して使うなら(x-1)*W+(y-1)でハッシュ化できる.
struct UnionFind {
    vector<int> par, rank, siz;
    // 構造体の初期化
    UnionFind(int n) : par(n,-1), rank(n,0),
        siz(n,1) { }
    // 根を求める
    int root(int x) {
    if (par[x]==-1) return x;
    else return par[x] = root(par[x]);
    }
    // x と y が同じグループに属するか (= 根が一致するか)
    bool issame(int x, int y) {
        return root(x)==root(y);
    }
    // x を含むグループと y を含むグループを併合する
    bool unite(int x, int y) {
        int rx = root(x), ry = root(y);
        if (rx==ry) return false;
        // union by rank
        if (rank[rx]<rank[ry]) swap(rx, ry);
        par[ry] = rx; // ry を rx の子とする
        if (rank[rx]==rank[ry]) rank[rx]++;
        siz[rx] += siz[ry];
        return true;
    }
    // x を含む根付き木のサイズを求める
    int size(int x) {
        return siz[root(x)];
    }
};

// union-find木がいくつの連結成分からなるかを返す
long long partial(UnionFind tree){
    long long n = tree.siz.size();
    vector<bool> seen(n, false);
    long long ans = 0;
    for (long long i = 0; i < n; i++){
        if (seen[tree.root(i)]) continue;
        seen[tree.root(i)] = true;
        ans++;
    }
    return ans;
}

// 無向グラフ
    Gがいくつの連結成分からなるかを返す
long long partial(Graph &G){
    long long siz = G.size();
    UnionFind ki(siz);
    for (long long i = 0; i < siz; i++){
        long long siz2 = G[i].size();
        for (long long j = 0; j < siz2; j++){
            ki.unite(i, G[i][j]);
        }
    }
    long long ret = partial(ki);
    return ret;
}
```

2.3 BIT

```
// 数列a(a[0],a[1],…,a[n-1])についての区間和と点更新を扱う
// 区間和,点更新,二分探索はO(log{n})
class BIT {
public:
    //データの長さ
    ll n;
    //データの格納先
    vector<ll> a;
    //コンストラクタ
    BIT(ll n):n(n),a(n+1,0){}

    //a[i]にxを加算する
    void add(ll i,ll x){
        i++;
        if(i==0) return;
        for(ll k=i;k<=n;k+=(k & -k)){
            a[k]+=x;
```

```
        }
    }

    //a[i]+a[i+1]+…+a[j]を求める
    ll sum(ll i,ll j){
        return sum_sub(j)-sum_sub(i-1);
    }

    //a[0]+a[1]+…+a[i]を求める
    ll sum_sub(ll i){
        i++;
        ll s=0;
        if(i==0) return s;
        for(ll k=i;k>0;k-=(k & -k)){
            s+=a[k];
        }
        return s;
    }

    //a[0]+a[1]+…+a[i]>=
        xとなる最小のiを求める(任意のkでa[k
        ]>=0が必要)
    ll lower_bound(ll x){
        if(x<=0){
            //
                xが0以下の場合は該当するものな

            return 0;
        }else{
            ll i=0;ll r=1;
            //最大としてありうる区間の長さを取得する
            //n以下の最小の二乗のべき(
                BITで管理する数列の区間で最大のもの
                )を求める
            while(r<n) r=r<<1;
            //区間の長さは調べるごとに半分になる
            for(int len=r;len>0;len=len>>1) {
                //その区間を採用する場合
                if(i+len<n && a[i+len]<x){
                    x-=a[i+len];
                    i+=len;
                }
            }
            return i;
        }
    }
};
```

Example C++ Code 3

```
#include <map>
#include <string>
using namespace std;

int main() {
    map<string, int> ages;
    ages["Alice"] = 30;
    ages["Bob"] = 25;
    ages["Charlie"] = 35;

    for (const auto &entry : ages) {
        cout << entry.first << ": " << entry.
            second << endl;
    }
    return 0;
}
```

Example Python Code 1

```
def hello_world():
    print("Hello, World!")

if __name__ == "__main__":
    hello_world()
```

Example Python Code 2

```
def sort_descending(numbers):
    return sorted(numbers, reverse=True)

numbers = [1, 2, 3, 4, 5]
sorted_numbers = sort_descending(numbers)
print(sorted_numbers)
```

Example Python Code 3

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def greet(self):
        print(f"Hello, my name is {self.name}
            and I am {self.age} years old.")

alice = Person("Alice", 30)
alice.greet()
```