

リーゼんと

テンプレ

ソースコード 1 テンプレ

```
1 #include<bits/stdc++.h>
2 using namespace std;
3 #define int long long
4 const int INF = 1e18;
5 const int MOD = 1e9 + 7;
```

ソースコード 2 ループ

```
1 #define _overload(_1, _2, _3, name, ...) name
2 #define _rep(i, n) repi(i, 0, n)
3 #define repi(i, a, b) for(int i = (int)(a); i < (int)(b); i++)
4 #define rep(...) _overload(__VA_ARGS__, repi, _rep)(__VA_ARGS__)
5 #define _rev(i, n) revi(i, n, 0)
6 #define revi(i, a, b) for(int i = (int)(a - 1); i >= (int)(b); i--)
7 #define rev(...) _overload(__VA_ARGS__, revi, _rev)(__VA_ARGS__)
8 #define each(i, n) for(auto&& i: n)
```

データ構造

ソースコード 3 UnionFind

```
1 struct UnionFind {
2     vector<int> t;
3     UnionFind(int size): t(size, -1) {}
4     int root(int x) {
5         return t[x] < 0 ? x : t[x] = root(t[x]);
6     }
7     int size(int x) {
8         return -t[root(x)];
9     }
10    bool isSame(int x, int y) {
11        return root(x) == root(y);
12    }
13    bool unite(int x, int y) {
14        x = root(x), y = root(y);
15        if(x != y) {
16            if(t[y] < t[x]) swap(x, y);
17            t[x] += t[y], t[y] = x;
18        }
19    }
20 }
```

```

19         return x != y;
20     }
21 };

```

ソースコード 4 セグ木

```

1 struct SegmentTree {
2     int n = 1;
3     vector<int> t;
4     SegmentTree(vector<int> v) {
5         while(n < v.size()) n *= 2;
6         t.resize(2 * n - 1, INF);
7         rep(i, v.size()) t[i + n - 1] = v[i];
8         rev(i, n - 1) t[i] = min(t[2 * i + 1], t[2 * i + 2]); // 区間最小
9     }
10    void update(int x, int val) {
11        x += n - 1;
12        t[x] = val;
13        while(x > 0){
14            x--; x /= 2;
15            t[x] = min(t[2 * x + 1], t[2 * x + 2]); // 区間最小
16        }
17    }
18    int query(int a, int b, int now = 0, int l = 0, int r = -1) {
19        if(r < 0) r = n;
20        if(r <= a || b <= l) return INF; // の単位元 min
21        if(a <= l && r <= b) return t[now];
22        int c1 = query(a, b, 2 * now + 1, l, (l + r) / 2);
23        int c2 = query(a, b, 2 * now + 2, (l + r) / 2, r);
24        return min(c1, c2);
25    }
26 };

```

数学

ソースコード 5 素因数分解

```

1 map<int, int> factorization(int n) {
2     map<int, int> res;
3     rep(i, 2, sqrt(n) + 1) {
4         if(i > 3) i++;
5         while(!(n%i)) {
6             res[i]++;
7             n /= i;
8         }
9     }

```

```

10     if(n > 1) {
11         res[n]++;
12     }
13     return res;
14 }

```

ソースコード 6 約数列挙

```

1 vector<int> divisor(int n) {
2     vector<int> res;
3     rep(i, 1, sqrt(n) + 1) {
4         if(!(n % i)) {
5             res.push_back(i);
6             if(i * i < n) res.push_back(n / i);
7         }
8     }
9     return res;
10 }

```

ソースコード 7 最小公倍数

```

1 int lcm(int a, int b) {return a / gcd(a, b) * b;}

```

ソースコード 8 累乗

```

1 int factorial(int a) {return a < 2 ? 1 : factorial(a - 1) * a;}

```

ソースコード 9 総和

```

1 int summation(int a) {return a < 1 ? 0 : (a * a + a) / 2;}

```

ソースコード 10 組み合わせ

```

1 int comb(int n, int r) {
2     int res = 1;
3     rep(i, 1, r + 1) {
4         res *= n--;
5         res /= i;
6     }
7     return res;
8 }

```

ソースコード 11 素数判定

```

1 bool isPrime(int n) {
2     rep(i, 2, sqrt(n) + 1) {
3         if(i > 3) i++;
4         if(!(n % i)) return false;
5     }

```

```
6   return true;
7 }
```

bound

ソースコード 12 bound

```
1 template<class T>
2 int lb(vector<T>& x,T n){return lower_bound(all(x) , n) - x.begin();}
3 template<class T>
4 int ub(vector<T>& x,T n){return upper_bound(all(x) , n) - x.begin();}
```

やの

テンプレ

ソースコード 13 テンプレ

```
1 #include<iostream>
2 #include<vector>
3 #include<algorithm>
4 #include<string>
5 #include<map>
6 #include<set>
7 #include<stack>
8 #include<queue>
9 #include<math.h>
10 using namespace std;
11 #define int long long
12 #define INF 1000000000
13 #define LLINF 9223372036854775807
14 #define mod 1000000007
```

ソースコード 14 alias

```
1 typedef vector<int> VI;
2 typedef pair<int, int> pii;
3 typedef priority_queue<int> PQ;
4 #define SORT(c) sort((c).begin(),(c).end())
5 #define rSORT(c) sort((c).rbegin(),(c).rend())
```

ソースコード 15 ループ

```
1 #define fore(i,a) for(auto &i:a)
2 #define REP(i,n) for(int i=0;i<n;i++)
3 #define eREP(i,n) for(int i=0;i<=n;i++)
```

```
4 #define FOR(i,a,b) for(int i=(a);i<(b);++i)
5 #define eFOR(i,a,b) for(int i=(a);i<=(b);++i)
```

データ構造

ソースコード 16 UnionFind

```
1 class UnionFind {
2     public:
3         vector<int> Parent;
4         UnionFind(int N) {
5             Parent = vector<int>(N, -1);
6         }
7         int root(int A) {
8             if (Parent[A] < 0) return A;
9             return Parent[A] = root(Parent[A]);
10        }
11        int size(int A) {return -Parent[root(A)];}
12
13        bool connect(int A, int B) {
14            A = root(A);
15            B = root(B);
16            if (A == B) return false;
17
18            if (size(A) < size(B)) swap(A, B);
19
20            Parent[A] += Parent[B];
21            Parent[B] = A;
22
23            return true;
24        }
25 };
```

ソースコード 17 セグ木

```
1 struct segtree {
2     public:
3         const int SIZE = 1 << 18;
4         VI seg, lazy;
5         misegtree() :seg(SIZE * 2, INF), lazy(SIZE * 2, INF) {}
6         void lazy_evaluate(int k, int l, int r) {
7             if (lazy[k] != 0) {
8                 //seg[k] += lazy[k];
9                 seg[k] = min(seg[k], lazy[k]);
10                if (r - l > 1) {
11                    //lazy[k * 2 + 1] += lazy[k];
```

```

12         //lazy[k * 2 + 2] += lazy[k];
13         //lazy[k * 2 + 1] = min(lazy[k * 2 + 1], lazy[k]);
14         //lazy[k * 2 + 2] = min(lazy[k * 2 + 2], lazy[k]);
15         //lazy[k * 2 + 1] = max(lazy[k * 2 + 1], lazy[k]);
16         //lazy[k * 2 + 2] = max(lazy[k * 2 + 2], lazy[k]);
17     }
18     lazy[k] = LLINF;
19 }
20 }
21 void update(int a, int b, int k, int l, int r, int x) {
22     lazy_evaluate(k, l, r);
23     if (r <= a || b <= l) return;
24     if (a <= l && r <= b) {
25         //lazy[k] += x;
26         //lazy[k] = min(lazy[k], x);
27         //lazy[k] = max(lazy[k], x);
28         lazy_evaluate(k, l, r);
29     } else {
30         update(a, b, k * 2 + 1, l, (l + r) / 2, x);
31         update(a, b, k * 2 + 2, (l + r) / 2, r, x);
32         //seg[k] = seg[k * 2 + 1] + seg[k * 2 + 2];
33         //seg[k] = min(seg[k * 2 + 1], seg[k * 2 + 2]);
34         //seg[k] = max(seg[k * 2 + 1], seg[k * 2 + 2]);
35     }
36 }
37 int query(int a, int b, int k, int l, int r) {
38     lazy_evaluate(k, l, r);
39     if (r <= a || b <= l) return LLINF;
40     if (a <= l && r <= b) return seg[k];
41     int x = query(a, b, k * 2 + 1, l, (l + r) / 2);
42     int y = query(a, b, k * 2 + 2, (l + r) / 2, r);
43     //return x + y;
44     //return min(x, y);
45     //return max(x, y);
46 }
47 void update(int a, int b, int x) { update(a, b, 0, 0, SIZE, x); }
48 int query(int a, int b) {
49     return query(a, b, 0, 0, SIZE);
50 }
51 };

```

グラフ

```

1  const int MAX_N = 100;
2  int G[100][100];
3  int N;
4  VI dist(110);
5
6  void dikstra(int s) {
7      priority_queue<pii, vector<pii>, greater<pii> >que;
8      dist[s] = 0;
9      que.push(pii(0, s));
10     while (!que.empty()) {
11         pii p = que.top();
12         que.pop();
13         int v = p.second;
14         if (dist[v] < p.first) continue;
15         REP(i, N) {
16             if (G[v][i] == LLINF) continue;
17             if (dist[i] > dist[v] + G[v][i]) {
18                 dist[i] = dist[v] + G[v][i];
19                 que.push(pii(dist[v], i));
20             }
21         }
22     }
23 }

```

bound

ソースコード 19 bound

```

1 #define LB(x,a) lower_bound((x).begin(),(x).end(),(a))
2 #define UB(x,a) upper_bound((x).begin(),(x).end(),(a))

```
