Standard Code Library

Final Fantasy

Zhejiang University City College

November 6, 2021

Contents

一切的开始

宏定义

- 需要 C++11

```cpp
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;
typedef pair<int, int> pii;
typedef pair<ll, ll> pll;

#define dbg(x...) \
    do {          \
        cout << #x << " -> "; \
        err(x);   \
    } while (0)

void err() {
    cout << endl;
}

template<class T, class... Ts>
void err(T arg, Ts... args) {
    cout << arg << ' ';
    err(args...);
}

void read() {}

template<class T, class... Ts>
void read(T &x, Ts &... xs) {
    T f = 1;
    char ch;
    x = 0;
    for (ch = getchar(); ch < '0' || ch > '9'; ch = getchar()) {
        if (ch == '-') f = -1;
    }
    for (; ch >= '0' && ch <= '9'; ch = getchar()) x = x * 10 + ch - '0';
    x *= f;
    read(xs...);
}

mt19937 mt(chrono::steady_clock::now().time_since_epoch().count());
ll rng(ll l, ll r) {
    uniform_int_distribution<ll> uni(l, r);
    return uni(mt);
}

template <class T>
void myHash(T a[], int n, T w[] = nullptr) {
    set<T> st;
    for (int i = 0; i < n; i++) {
        st.insert(a[i]);
    }
    int tot = 0;
    map<T, int> mp;
    for (T x : st) {
        mp[x] = ++tot;
    }
    for (int i = 0; i < n; i++) {
        a[i] = mp[a[i]];
    }
    if (w != nullptr) {
        for (pair<T, int> p : mp) {
            w[p.second] = p.first;
        }
    }
}

```

```cpp
template<class T>
void unique(vector<T> &v) {
    sort(v.begin(), v.end());
    v.erase(unique(v.begin(), v.end()), v.end());
}

const int maxn = 1e5 + 7;
const int inf = 0x3f3f3f3f;
const ll INF = 0x3f3f3f3f3f3f3f3f;
const int mod = 1e9 + 7;

void run() {

}

/*

*/

int main() {

    int T = 1;
    read(T);
    while (T--) {
        run();
    }

    return 0;
}
```

STL

bitset

头文件

```
#include <bitset>
```

指定大小

```
bitset<1000> bs;  // a bitset with 1000 bits
```

构造函数

- `bitset()`：每一位都是 `false`。
- `bitset(unsigned long val)`：设为 `val` 的二进制形式。
- `bitset(const string& str)`：设为串 `str`。

运算符

- `operator []`：访问其特定的一位。
- `operator ==/!=`：比较两个 `bitset` 内容是否完全一样。
- `operator &/&=/|/|  =/^/^=/~`：进行按位与/或/异或/取反操作。**`bitset` 只能与 `bitset` 进行位运算**，若要和整型进行位运算，要先将整型转换为 `bitset`。
- `operator <>/<<=/>>=`：进行二进制左移/右移。
- `operator <>`：流运算符，这意味着你可以通过 `cin/cout` 进行输入输出。

成员函数

- `count()`：返回 `true` 的数量。
- `size()`：返回 `bitset` 的大小。
- `test(pos)`：它和 `vector` 中的 `at()` 的作用是一样的，和 `[]` 运算符的区别就是越界检查。

- `any()`：若存在某一位是 `true` 则返回 `true`，否则返回 `false`。
- `none()`：若所有位都是 `false` 则返回 `true`，否则返回 `false`。
- `all()`：**C++11**，若所有位都是 `true` 则返回 `true`，否则返回 `false`。
- 1. `set()`：将整个 `bitset` 设置成 `true`。
  2. `set(pos, val = true)`：将某一位设置成 `true` / `false`。
- 1. `reset()`：将整个 `bitset` 设置成 `false`。
  2. `reset(pos)`：将某一位设置成 `false`。相当于 `set(pos, false)`。
- 1. `flip()`：翻转每一位。（相当于异或一个全是的 `bitset`）
  2. `flip(pos)`：翻转某一位。
- `to_string()`：返回转换成的字符串表达。
- `to_ulong()`：返回转换成的 `unsigned long` 表达（`long` 在 NT 及 32 位 POSIX 系统下与 `int` 一样，在 64 位 POSIX 下与 `long long` 一样）。
- `to_ullong()`：**C++11**，返回转换成的 `unsigned long long` 表达。

一些文档中没有的成员函数：

- `_Find_first()`：返回 `bitset` 第一个 `true` 的下标，若没有 `true` 则返回 `bitset` 的大小。
- `_Find_next(pos)`：返回 `pos` 后面（下标严格大于 `pos` 的位置）第一个 `true` 的下标，若 `pos` 后面没有 `true` 则返回 `bitset` 的大小。

vector

- 几乎在所有容器中，pop 和 $clear$ 只清除元素，不清除内存
- $vector$ 中释放内存的方法

```
vector<int>().swap(vec);
```

deque

- $deque$ 的内部实现是预先占用多个连续的存储块，添加元素并不需要重新分配空间
- 因此，向 $deque$ 两端添加/删除元素的时间复杂度很小，但是同时 $deque$ 初始化时便会占用较大的额外空间
- 非常不推荐同时使用多个 $deque$，如果一定要用（如多个单调队列），建议使用 $vector$ 加头尾指针模拟，使用 $vector.resize$ 初始化空间
- 以下写法在 hdu 一道 $528mb$ 的题中 MLE 了

```
deque<dqNode> dq[maxn]; //maxn = 1e6 + 5
```

博弈

SG 函数

- 所有后继状态的 MEX

多个游戏组合

- 前提：多个公平游戏且游戏之间相互独立
- 结论：每个游戏的 SG 值异或和为 0 则先手必败，反之先手必胜

阶梯 NIM

题面

N 堆石子，两人轮流操作，一次操作为挑选一堆石子 i，将至少 1 个石子移动至 $i-1$ 位置（$i=1$ 则被移出游戏），不能操作者输。

结论

相当于对所有奇数位置上的石子堆做 NIM 游戏。即 $a_1⊠a_3⊠...=0$，则先手必败

斐波那契博弈

题面

有一堆个数为 $n(n \geq 2)$ 的石子，游戏双方轮流取石子，规则如下

- 先手不能在第一次把所有石子取完，至少取 1 颗
- 之后每次取石子数范围 $[1, 2 * a_{i-1}]$，a_{i-1} 表示对手上一轮取石子的数量
- 取走最后一个石子的为赢家

结论

当 n 为 $Fibonacci$ 数时，先手必败。

图论

LCA

- 倍增

```
void dfs(int u, int fa) {
    pa[u][0] = fa; dep[u] = dep[fa] + 1;
    FOR (i, 1, SP) pa[u][i] = pa[pa[u][i - 1]][i - 1];
    for (int& v: G[u]) {
        if (v == fa) continue;
        dfs(v, u);
    }
}

int lca(int u, int v) {
    if (dep[u] < dep[v]) swap(u, v);
    int t = dep[u] - dep[v];
    FOR (i, 0, SP) if (t & (1 << i)) u = pa[u][i];
    FORD (i, SP - 1, -1) {
        int uu = pa[u][i], vv = pa[v][i];
        if (uu != vv) { u = uu; v = vv; }
    }
    return u == v ? u : pa[u][0];
}
```

计算几何

二维几何：点与向量

```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
```

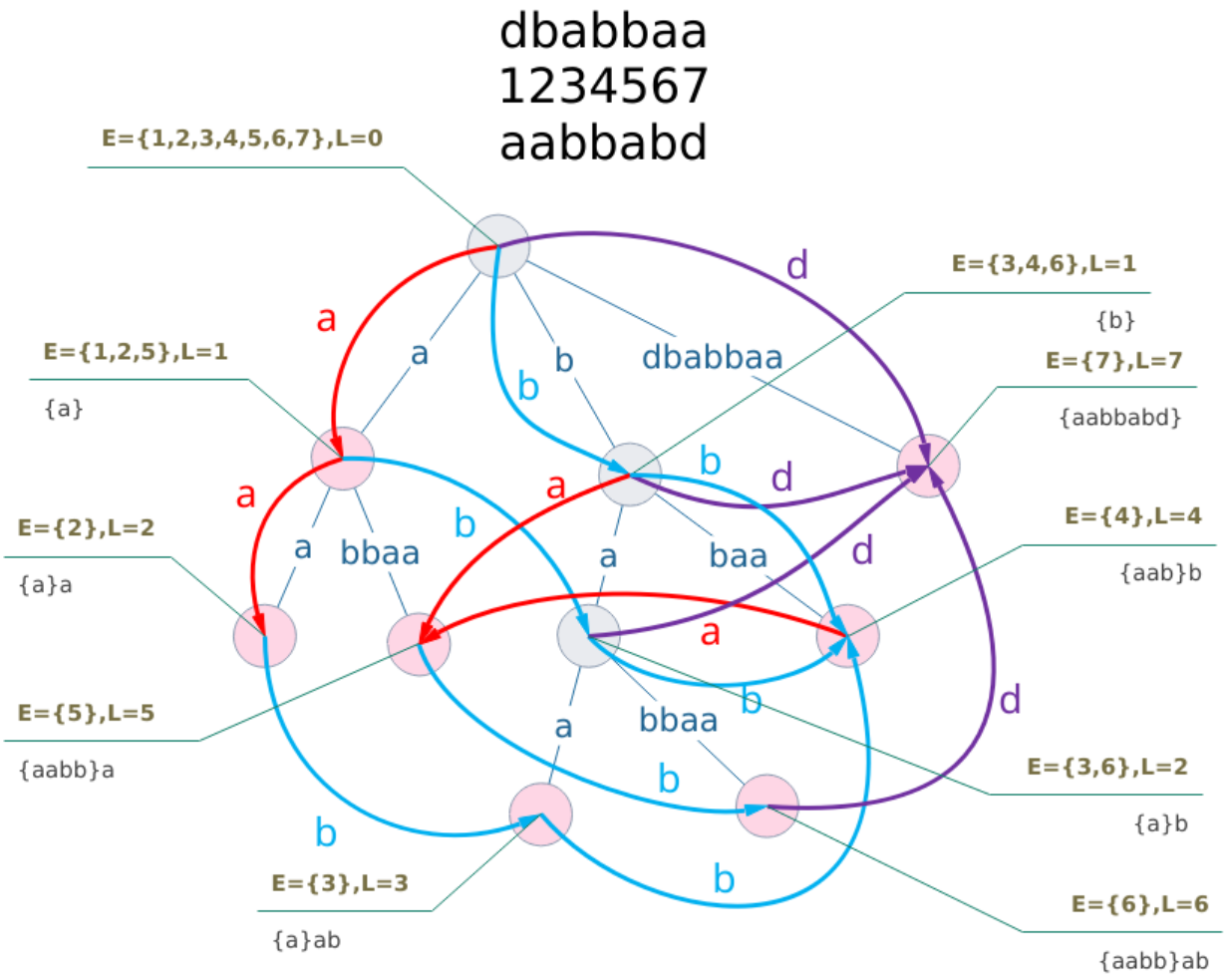
```cpp
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// ------------------------------------------------
```

字符串

后缀自动机



杂项

STL

- copy

```
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```