新生赛题解

苏州大学 ACM 集训队

日期：December 14, 2021

目录

1 罗峰为什么是神

Problem by **Ryeblablabla**. Tutorial by **Ryeblablabla**.

罗峰为什么是神？在讨论这个问题之前，我们先说说其他选手相较于罗峰究竟差在了哪里？

1.1 题目意思

懂的都懂。

1.2 题解

懂的都懂。

2 Peter 和他的王国 1

Problem by **Peterlits Zo**. Tutorial by **Peterlits Zo**.

道理我都懂，不过为啥这个题目后面有一个数字 1 呢?

2.1 题目意思

给定一个数组 A，第 i 个元素为 A_i。

构造一个集合，即:

$$\{x \mid A_i \in A, A_j \in A, x \mid A_i, x \mid A_j, x \leq l\}$$

求该集合中的最大值。

2.2 题解

我们知道，最大公约数能够被其他所有的公约数整除，那么如果我们求出最大公因数的所有因数，然后二分即可得到在 $[1, l]$ 区间内的公约数。时间复杂度为 $O(\sqrt{P})$。其中分解质因数的时间复杂度是 $O(\sqrt{P})$，而二分查找是 $O(\log \sqrt{P})$，可以忽略不计。

总复杂度为两两组合乘以上面的复杂度，故为 $O(n^2\sqrt{P})$。

3 简单博弈

Problem by **TheNameless**. Tutorial by **TheNameless**.

博弈题。

3.1 题目意思

Aob 和 Bob 进行游戏，其中 Alice 先手。

他们对同一个数字 n 进行操作，在每一个属于自己的回合中，他们可以选择一个数字 $k \in [l, r]$，并令 n 自身减去 k。在自己的回合内，n 被减为负数时，则失败。

双方均采取最优策略。问获胜者。

3.2 题解

从游戏规则中，我们可以发现，无论先手选的数字 k 是多少，后手都可以选数字 $l + r - k$，使得 n 相比两人操作前减少 $l + r$。发现这个规律后，就不难找到必胜的办法。

假设 $n \bmod (l+r) < l$ ($\bmod$ 为取余数操作) , 那么在 Alice 先手选数字 k 后, Bob 只需要选数字 $l+r-k$, 这样轮流操作多次后一定会在 Bob 操作完后使得原始 n 变为 $n \bmod (l + r)$ ，此时无论 Alice 选什么数都会输。Bob 必胜。

假设 $l \le n \bmod (l + r) \le r$ ，那么一开始的时候 Alice 可以选择 $n \bmod (l + r)$。这样操作之后的新的 n 就变为了 $(l + r)$ 的整数倍。在之后的操作中，无论 Bob 取什么 k ，Alice 都可以取 $l + r - k$。多次操作后，n 一定会在 Alice 操作后变为 0，Bob 必输，Alice 必赢。

假设 $n \bmod (l+r) > r$, 那么一开始的时候 Alice 可以选择 r。这样操作之后的新的 n 一定满足 $n \bmod (l+r) < l$， 局面就变成了上面讨论后的情况，只是 Alice 和 Bob 的先后手对调了。因此，Alice 必赢。

综上，只需判断初始时 $n \bmod (l + r)$ 与 l 大小关系就可以知道获胜者是谁了。

4 原来你也玩原神

Problem by **Ryeblablabla**. Tutorial by **dyyyyyyyy**.

大丘丘病了二丘丘瞧，三丘丘采药四丘丘 嗷!

4.1 题目意思

有 5 种圣遗物，每种圣遗物有 4 个词条，比如 HP 生命值，HP Rate 生命值百分比等等。

求在当前 20 个词条的加成下，期望伤害是多少?

期望伤害 = 攻击 × (1 − 暴击率) + 暴击率 × (1 + 暴击伤害率) × 攻击

4.2 题解

模拟题

每次读入一行的数据里需要提取的部分有：

- 词条类型
- 数值大小
- 这两者是以 + 号分割的

另外，对于期望伤害来说我们只需要记录攻击，暴击率，暴击伤害这些词条，其他词条可以不管

按题目所给的计算公式计算答案即可

5 签到题?!

Problem by **dyyyyyyyy**. Tutorial by **dyyyyyyyy**.

动态规划

或许某些 dalao 们一定会想到或许是什么高级的字符串数据结构。

这其实是一道标程 30 行不到的动态规划题。

如果把子序列改成子串的话那确实是后缀自动化上 dp 的题。

5.1 题目意思

给定一个数字组成的字符串。

求其中所有不含 6 的不同的子序列对应的十进制数的和。

5.2 题解

我们先来看一个子问题：如何计算有多少个不同的子序列？

- dp[i] 表示 $s[i]$ 距离上次 $S[i]$ 这个数字出现这段距离内多出现的以 $S[i]$ 结尾的新子序列的个数。
- 他的定义很复杂，但是转移感性理解起来是十分简单的。
- 记 $last[i]$ 为 $s[i]$ 的上一次出现位置，如果没出现即为 0。
- 对于 $last[i]+1, ..., i-1$ 的所有位置对应的序列都是对当前的 $dp[i]$ 有贡献的，不难想对于之前一样的数字，那么已经统计过了，就不需要统计了。

直接看代码可能会更清楚一些。

```
dp[0] = 1;
for (int i = 1; i <= n; i++) {
    for (int j = i - 1; j >= 0; j--) {
        dp[i] += dp[j]; dp[i] %= mod;
        if (s[j] == s[i]) break;
    }
}
```

那么怎么处理带 6 的字符串呢？先预处理掉或者遇到 6 就 `continue`。

那么怎么求解数字和呢？这其实很好实现，另外维护一个数组 ans，和 dp 类似，转移为：

$$ans[i] \leftarrow 10 \times ans[j] + s[i] \times dp[j]$$

时间复杂度：$O(|S||\sum|)$。

主体代码如下：

```
dp[0] = 1;
for (int i = 1; i <= n; i++) {
    if (s[i] == '6') continue;
    for (int j = i - 1; j >= 0; j--) {
        if (s[j] == '6') continue;
        dp[i] += dp[j]; dp[i] %= mod;
        ans[i] += (1ll * 10 * ans[j] + 1ll * (s[i] - '0') * dp[j]) % mod; ans[i] %= mod;
        if (s[j] == s[i]) break;
    }
    res += ans[i]; res %= mod;
}
```

6 xygg 的加乘法

Problem by **Emma194**. Tutorial by **PeterlitsZo**.

什么牛马题?

6.1 题目意思

给定 $n \leq 10$ 个操作，和一个初始值 x。我们可以任意调换操作执行的顺序。

求模 $1 \times 10^9 + 7$ 之后最小的值。

6.2 题解

首先我们不妨找出来一个朴素算法，即枚举所有的排列情况，然后选取最小的即可：

```
int result = 0x3f3f3f3f;
for(; flg; flg = next_permutation(OP, OP + n)) {
    ll sub_res = x;
    for(int i = 0; i < n; i++) {
        if(OP[i].first == '+') {
            sub_res = (sub_res + OP[i].second) % MOD;
        } else {
            sub_res = (sub_res * OP[i].second) % MOD;
        }
    }
    result = min(result, (int)sub_res);
}
```

但是所有的排列数为 $10! \simeq 3 \times 10^6$。而 $T = 1 \times 10^3$，仅仅只有两个相乘便超过了 5×10^8，故肯定会超时。

我们需要寻找一个优化的方法。

我们可以发现，所有的操作均是一段乘法和一段加法。而加法乘法无论怎么搞，其次序都是不影响结果的。我们可以把这个给剪掉，以降低复杂度。

那么最坏的情况不过是五个加法和五个乘法而已。而它的复杂度能证明出来是小于 1×10^6，那么只要常数较小，就可以求解了。

为了让常数较小，我们还需要预处理一下，即如果我们选择一些操作，我们必须得在平均复杂度为 $O(1)$ 的时间内找出来这些操作的和或者积。

```
int get_res(int i, int cur_op) {
    if(ANS[i][cur_op] != 0x3f3f3f3f)
        return ANS[i][cur_op];
    // 乘法的话，单位元是 1，不然是 0。
    int get = (cur_op == 1 ? 1 : 0);
    for(int j = 0; j < 12; j++) {
        if((i >> j) & 1) {
            if (cur_op == 1) get = (get * 1ll * OP[j].second) %MOD;
            else get = (get + OP[j].second) % MOD;
        }
    }
    return ANS[i][cur_op] = get;
}
```

之后基于这些进行 DFS 搜索即可。搜索到叶子节点的时候进行更新。

7 最大异或值

Problem by **TTDragon**. Tutorial by **dyyyyyyyy**.

思维题。

7.1 题目意思

对于 $[1, n]$ 内整数 i 和 j，求 $\max\{i \oplus j\}$。

7.2 题解

先说结论：

- 记 x 表示 n 二进制的位数 (除去前导 0)。
- 答案是 $2^x - 1$，($n = 1$ 的情况需要特判)。

首先证明 $2^x - 1$ 是可行的，因为 n 是一个 x 位二进制数。

对于 $n > 1$ 有：

- $1 \le 2^{x-1} \le n$。
- $1 \le 2^{x-1} - 1 \le n$。
- $2^{x-1} \oplus (2^{x-1} - 1) = 2^x - 1$。

然后我们证明不可能得出 $\geq 2^x$ 的答案。

对于一个 $\geq 2^x$ 的答案来说，其最高位的位数一定是 $> x$ 的，而对于所有在 $[1, n]$ 以内的数，他们 $> x$ 二进制位上的值都为 0，显然无法得出 $\geq 2^x$ 的答案。

8 嘉然只属于我

Problem by **baigeiguai**. Tutorial by **baigeiguai**.

woc 然!

8.1 题目意思

小斌命中的概率是 30%，小雨比他好些，命中率是 50%，最出色的枪手是小 p，他从不失误，命中率是 100%。由于这个显而易见的事实，为公平起见，他们决定这样的出题顺序：小斌先开枪，小雨第二，小 p 最后。然后这样循环，直到他们只剩下一个人。那么这三个人中谁活下来的机会最大呢?

8.2 题解

小斌存活概率最大。

小斌 有三个选择, 空枪, 射击小雨, 射击小 p.

小斌不会选择射击小雨, 因有 30% 概率小雨死亡, 小 p 射击, 小雨必死, 死亡概率 30%;

小斌不会选择射击小 p, 因有 30% 概率小 P 死亡, 小雨回击, 由于小斌的命中率低于小雨，从概率角度，不是良策。

小斌会选择空枪, 因为小雨必然射击小 p, 小 p 死亡概率 50%;

小 P 若不死, 必然射击小雨, 小雨死亡概率 $50\% * 100\% = 50\%$;

小斌死亡概率为 0.

此时, 小雨和小 p 中间必然死亡一人. 小斌可能面对小雨, 可能面对 小 p。

面对小雨, 小斌生存概率 (等比数列求极限)

$$30\% + 70\% * 50\% * 30\% + 70\% * 50\% * 70\% * 50\% * 30\% + ... = \frac{6}{13}$$

面对小 p, 小斌生存概率 30%

这一情况下，小 p 生存概率 70%

汇总生存概率为:

- 小斌: $50\% * 30\% + 50\% * \frac{6}{13} = \frac{99}{260}$
- 小雨: $50\% * \frac{7}{13} = \frac{70}{260}$
- 小 p: $50\% * 70\% = \frac{91}{260}$

因此小斌生存概率最高. 采取方法如上所述。

9 happy 子序列

Problem by **baigeiguai**. Tutorial by **dyyyyyyyy**.

贪心 + 暴力。

9.1 题目意思

求解字符串中是否存在一个子序列，等于给定的字符串。

9.2 题解

这里的匹配串 $T =$ `"Carol"`，我们只需要判断 T 是否在 S 中出现就行。
那么一个很简单的贪心想法是，找到一个出现 T_1 的位置 p_1，再向后找第一个出现 T_2 的位置 $p_2 \cdots\cdots$ 就这样找下去，最后判断有没有找到就行。
时间复杂度：$O(|S||T|)$。
另外可以思考一下 T 作为子序列的出现次数怎么求。

10 Rabbit House 的新菜单

Problem by **lzc2001, KryptonAu, ddlearn**. Tutorial by **dyyyyyyyy**.

经典且比较裸的容斥原理题。

10.1 题目意思

给定 n 个菜品，其中第 i 个有两个属性值 a_i 和 b_i。
我们需要计算满足下列条件的组合数：

1. 选择 3 个不同的菜品。
2. 菜品的 a_i 两两不相同或者 b_i 两两不相同。

10.2 题解

我们需要记录三个东西：

1. `map<pair<int, int>, int> cnt`：`cnt[{x, y}]` 表示点对 (x, y) 的菜品有多少个
2. `map<int, int> cnta`：`cnta[x]` 表示 $a_i = x$ 的 菜品有多少个
3. `map<int, int> cntb`：`cntb[x]` 表示 $b_i = x$ 的 菜品有多少个

如果不考虑约束条件，所有的选择方法为 $\binom{n}{3}$。
先在我们考虑计算不满足约束条件的情况 ans_v。
有哪些情况不满足呢?
对于每一种 (a_i, b_i) 来说，

- 为方便表示，令：$v =$ `cnt[{`a_i`, `b_i`}]`，$x =$ `cnta[`a_i`]`，$y =$ `cntb[`b_i`]`。
- 三个菜品的美味度和观赏度都一样，有 $\binom{v}{3}$ 种。
- 两个菜品的美味度和观赏度都一样，有 $\binom{v}{2} \times (n - v)$ 种。
- 两个菜品的美味度一样，两个菜品的观赏度一样，有 $v \times (x - v) \times (y - v)$ 种。

用总情况减去不满足约束的情况就是满足约束的情况。
时间复杂度：$O(N \log N)$。

11 参考代码

11.1 why_is_lf_god.py

```python
print("lfgg yyds!")
```

11.2 peter_and_his_country.cpp

```cpp
// By Peterlits Zo <peterlitszo@gmail.com>
// ^_^
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

int main() {
    // input l, n and array A
    ll l;
    int n;
    scanf("%lld %d", &l, &n);
    constexpr int LEN = 100 + 16;
    static ll A[LEN];
    for (int i = 0; i < n; i ++) {
        scanf("%lld", &A[i]);
    }

    // try to find the max.
    ll ans = 0;
    for(int i = 0; i < n; i ++) {
        for(int j = i + 1; j < n; j ++) {
            int G = __gcd(A[i], A[j]);

            // get all div
            vector<int> divs;
            for(int k = 1; k * k <= G; k ++) {
                if(G % k == 0) {
                    divs.push_back(k);
                    if(k * k != G) divs.push_back(G / k);
                }
            }

            // get the element
            sort(divs.begin(), divs.end());
            auto ele_ptr = --upper_bound(divs.begin(), divs.end(), l);
            ans = max(ans, (ll)*ele_ptr);
        }
    }
```

```
39
40      printf("%lld", ans);
41      return 0;
42  }
```

11.3 simple_game.cpp

```
1   #include<bits/stdc++.h>
2   using namespace std;
3   typedef long long ll;
4   int main(void) {
5       int T;
6       cin >> T;
7       while (T--) {
8           ll l, r, n;
9           cin >> l >> r >> n;
10          if (n % (l + r) < l) puts("Bob");
11          else puts("Alice");
12      }
13      return 0;
14  }
```

11.4 you_also_play_genshenimpact.py

```
1   atk = 0
2   atk_rate = 0
3   critical_hit = 50
4   ch_rate = 5
5
6   for _ in range(25):
7       s = input().split('+')
8       if s[0] == 'ATK':
9           atk += float(s[1])
10      elif s[0] == 'ATK Rate':
11          atk_rate += float(s[1][:-1])
12      elif s[0] == 'Crit Rate':
13          ch_rate += float(s[1][:-1])
14      elif s[0] == 'Crit DMG Rate':
15          critical_hit += float(s[1][:-1])
```

```
atk = 1500*(1+atk_rate/100)+atk
ch_rate = min(ch_rate, 100)/100
ans = atk*(1-ch_rate)+ch_rate*(1+critical_hit/100)*atk
print(ans)
```

11.5 sign_in_question.cpp

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

const int N = 1e6 + 5, mod = 998244353;
char s[N];
int ans[N], dp[N], res;

int main(void) {
    scanf("%s", s + 1);
    int n = strlen(s + 1);
    dp[0] = 1;
    for (int i = 1; i <= n; i++) {
        if (s[i] == '6') continue;
        for (int j = i - 1; j >= 0; j--) {
            if (s[j] == '6') continue;
            dp[i] += dp[j]; dp[i] %= mod;
            ans[i] += (1ll * 10 * ans[j] + 1ll * (s[i] - '0') * dp[j]) % mod; ans[i] %=
                mod;
            if (s[j] == s[i]) break;
        }
        res += ans[i]; res %= mod;
    }
    cout << res << endl;
    return 0;
}
```

11.6 xygg_add_and_times.cpp

```
#include <bits/stdc++.h>
using namespace std;
using ll=long long;

```

```cpp
constexpr int MOD = 1e9 + 7;
constexpr int LEN = 12;
pair<char, int> OP[LEN];

int n, times_mark, add_mark, ans;

int ANS[1<<LEN][2];

int get_res(int i, int cur_op) {
    if(ANS[i][cur_op] != 0x3f3f3f3f)
        return ANS[i][cur_op];
    // 乘法的话，单位元是 1，不然是 0。
    int get = (cur_op == 1 ? 1 : 0);
    for(int j = 0; j < 12; j++) {
        if((i >> j) & 1) {
            if (cur_op == 1) get = (get * 1ll * OP[j].second) %MOD;
            else get = (get + OP[j].second) % MOD;
        }
    }
    return ANS[i][cur_op] = get;
}

// cur_op: 1 -> times, 0 -> add.
// state: 1 -> able ok to use, 0 -> unable.
void dfs(int bef, int state, int cur_op) {
    // state == 0 -> leaf node
    if(!state) {
        ans = min(ans, bef);
        return;
    }
    // Times. So choose a subset to times.
    if (cur_op == 1) {
        int aim = state & times_mark;
        if(!aim) { /* nothing to choose */ return; }
        for(int i = aim; i; i = (i - 1) & aim) {
            dfs(bef * 1ll * get_res(i, cur_op) % MOD, state & ~i, !cur_op);
        }
    // Add. So choose a subset to add.
    } else {
        int aim = state & add_mark;
```

```
        if(!aim) { /* nothing to choose */ return; }
        for(int i = aim; i; i = (i - 1) & aim) {
            dfs((bef + get_res(i, cur_op)) % MOD, state & ~i, !cur_op);
        }
    }
}

int main(){
    int t; scanf("%d", &t);
    while(t--) {
        // 初始化 ANS。
        memset(ANS, 0x3f, sizeof ANS);
        ans = 0x3f3f3f3f;

        // 读入并处理加操作和乘操作的掩码。
        int x; scanf("%d%d", &n, &x);

        times_mark = 0;
        add_mark = 0;
        for(int i = 0; i < n; i++){
            scanf(" %c%d", &OP[i].first, &OP[i].second);
            if(OP[i].first == '*') times_mark |= (1 << i);
            else add_mark |= (1 << i);
        }

        // 朴素无华的搜索
        dfs(x, (1 << n) - 1, 1);
        dfs(x, (1 << n) - 1, 0);
        printf("%d\n", ans);
    }
    return 0;
}
```

11.7 max_xor_value.cpp

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

int n;

```

```cpp
int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    cin >> n;
    if (n == 1) {
        cout << 0 << endl;
        return 0;
    }
    int cnt = 0;
    while (n) {
        cnt++; n >>= 1;
    }
    cout << (1 << cnt) - 1 << endl;
    return 0;
}
```

11.8 jiaran_only_belong_to_me.py

```python
print("B\n198\n140\n182\n")
```

11.9 happy_subsequence.cpp

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


const int maxn = 1e4 + 5;
char s[maxn], p[] = {'C', 'a', 'r', 'o', 'l'};
int main() {
    scanf("%s", s + 1);
    int n = strlen(s + 1), cur = 0;
    for (int i = 1; i <= n; ++i) {
        if (s[i] == p[cur]) {
            ++cur;
            if (cur == 5) break;
        }
    }
    if (cur == 5) printf("Happy");
    else printf("Emmm");
```

```
    return 0;
}
```

11.10 rabbit_house_new_menu.cpp

```
#include<bits/stdc++.h>
using namespace std;
typedef long long ll;

const int N = 1e5 + 5;
int n;
map<pair<int, int>, int> mp;
int cntx[N], cnty[N];

int main(void) {
    ios::sync_with_stdio(false);
    cin.tie(0); cout.tie(0);
    cin >> n;
    for (int i = 1, x, y; i <= n; i++) {
        cin >> x >> y;
        cntx[x]++; cnty[y]++;
        mp[{x, y}]++;
    }
    ll ans = 1ll * n * (n - 1) * (n - 2) / 6;
    for (auto it: mp) {
        int x = it.first.first, y = it.first.second, cnt = it.second;
        ans -= 1ll * cnt * (cnt - 1) * (cnt - 2) / 6;
        ans -= 1ll * cnt * (cnt - 1) / 2 * (n - cnt);
        ans -= 1ll * cnt * (cntx[x] - cnt) * (cnty[y] - cnt);
    }
    cout << ans << endl;
    return 0;
}
```