

# 蓝桥杯选拔赛第一场「部分题解」

## A 题：lrq学长的神奇数列

该题是一道填空题，数据量是  $10^{10}$ ，在蓝桥杯里面优先考虑暴力解法，大概需要运行 10 多分钟出结果

在上一届蓝桥杯中，就是有一道题可以运行 30 多分钟就可以出答案，但是某人却死硬求导正解，导致错失 5 分！**不会的题，尽可能的暴力、或者贪心，就算不能全对，能混分就是赢**

暴力代码：

```
#include <iostream>

int main()
{
    long long mod = 1e9 + 7;
    long long a, b, c, d;
    a = b = c = 1;
    long long n = 1e10;
    for (long long i = 4; i <= n; i++)
    {
        d = (a + c) % mod;
        a = b;
        b = c;
        c = d;
    }
    std::cout << d << '\n';
}
```

最终结果：

191782920

## E 题：你学过素数吗？

判断一个数，是否是素数，一般用  $\sqrt{n}$  算法都可以混到很多分了：

```
#include <bits/stdc++.h>

// 判断 n 是否是素数：
bool isPrim(int n) {
    if (n == 2) return true;
    for (int i = 2; i <= n / i; i++) {
        if (n % i == 0) return false;
    }
    return true;
}
```

```

}

int l, r;

int main() {
    std::cin >> l >> r;
    for (int i = l; i <= r; i++) {
        if (isPrim(i)) std::cout << i << ' ';
    }
    return 0;
}

```

## D 题：寻找 lrq

只需要找到最大值所在的下标就是答案

```

#include <bits/stdc++.h>
long long n;
int main() {
    scanf ("%lld", &n);
    long long ans = 1;
    long long val; scanf ("%lld", &val);
    for (long long i = 2; i <= n; i++) {
        long long t; scanf ("%lld", &t);
        if (val < t) {
            val = t; ans = i;
        }
    }
    printf ("%lld\n", ans);
}

```

## F 题：最大递增区间

枚举每一个下标，从该下标开始向右边尽可能长地扩展的递增区间，不断记录最大的区间首尾差值

```

#include <bits/stdc++.h>

int n;
int a[(int)4e6];

// 从 p 开始的最长递增区间的右端点下标
int right(int p) {
    int ans = p;
    for (int i = p + 1; i <= n; i++) {
        if (a[i] >= a[i - 1]) ans = i;
        else break;
    }
    return ans;
}

```

```

}

int main() {
    int t; scanf ("%d", &t);
    while (t --) {
        scanf ("%d", &n);
        for (int i = 1; i <= n; i ++) scanf ("%d", a + i);

        // 求出递增区间首尾的最大差值
        int max = 0;
        {
            for (int i = 1; i <= n; i ++) {
                int j = right(i);
                max = std::max(max, a[j] - a[i]);
                i = j; // 过滤掉递增区间的中间部分
            }
        }

        // 输出递增区间首位差值是 max 的区间
        {
            for (int i = 1; i <= n; i ++) {
                int j = right(i);
                if (a[j] - a[i] == max) {
                    std::cout << i << ' ' << j << ' ';
                }
                i = j;
            }
        }

        std::cout << '\n';
    }
}

```

## H 题：lrq学长寻找刻晴的圣遗物

经典的暴力广度搜索题，要想在蓝桥杯混高分，一定要学会暴力做题，就算不能全对，只要混到分，就离奖牌更进一步

```

#include <bits/stdc++.h>

int h, w;
int g[22][22];
int on[22][22];

int bfs(int x, int y) {
    if (g[x][y] == 0) return 0;
    std::queue<std::array<int, 3>> qu;
    qu.push({x, y, 0});
    int ans = 0;
    int x_[] = {0, 0, -1, 1};
    int y_[] = {-1, 1, 0, 0};

```

```

while (qu.size()) {
    auto [x, y, d] = qu.front();
    qu.pop();
    if (g[x][y] == 0) continue;
    if (on[x][y]) continue;
    on[x][y] = 1;
    ans = std::max(ans, d);
    for (int i = 0; i < 4; i++) {
        qu.push({x + x_[i], y + y_[i], d + 1});
    }
}
return ans;
}

void solve() {
    int ans = 0;
    for (int i = 1; i <= h; i++) {
        for (int j = 1; j <= w; j++) {
            std::memset(on, 0, sizeof on);
            ans = std::max(ans, bfs(i, j));
        }
    }
    std::cout << ans << '\n';
}

int main() {
    std::cin >> h >> w;
    for (int i = 1; i <= h; i++) {
        std::string s; std::cin >> s;
        for (int j = 1; j <= w; j++) {
            if (s[j - 1] == '.') g[i][j] = 1;
        }
    }
    solve();
    return 0;
}

```

## I 题：计数

非常经典的二分查找题，需要先对身高排序，然后二分查找答案所在的位置，假设是从小到大排序的，且瘦瘦师弟的身高所位于的排名为  $p$ ，则有  $p - 1$  个人比瘦瘦师弟矮，相对的，就有  $n - (p - 1)$  个人不比瘦瘦师弟矮：

```

#include <bits/stdc++.h>

int n, q;
int a[1000000];
int p;

void solve() {
    int ans = n - (std::lower_bound(a + 1, a + 1 + n, p) - a - 1);
}

```

```

        std::cout << ans << '\n';
    }

    int main() {
        std::cin >> n >> q;
        for (int i = 1; i <= n; i++) std::cin >> a[i];
        std::sort(a + 1, a + 1 + n);
        while (q--) {
            std::cin >> p;
            solve();
        }
        return 0;
    }
}

```

## J 题：奇数个 1 的子串个数

考虑动态规划，思路如下：

假设我知道了当前  $1 \sim i$  的子串含有奇数个 1 的子串的个数，又知道了以  $i$  为结尾的且含有奇数个 1 的子串的个数、含有偶数个 1 的子串的个数，又应该如何推导出  $1 \sim i + 1$  的子串含有奇数个 1 的个数呢？

记忆化递归可以询问『刘庭权』！

```

#include <bits/stdc++.h>
typedef long long var;
typedef __int128 hh;

namespace {

    const var N = 5e6;
    var n;
    std::string s;
    var dp_1[N], dp_2[N], dp_0[N];
    var dp[N];
    // 分别代表 1 ~ i 的后缀子串含有奇数个、偶数个 1 的个数
    // dp_0 代表以 i 为末尾的后缀子串连续 0 的长度
    // dp 代表 1 ~ i 含义奇数个 1 的子串个数

    // 求含有奇数个 1 的子串个数
    void solve() {
        std::cin >> n >> s; s = ' ' + s;
        for (var i = 1; i <= n; i++) {
            if (s[i] - '0') {
                if (i - 1 == 0) {
                    dp_1[i] = 1; continue;
                }
                dp_1[i] = dp_2[i - 1] + 1 + dp_0[i - 1];
                dp_2[i] = dp_1[i - 1];
            }
            else {
                dp_1[i] = dp_1[i - 1];
            }
        }
    }
}

```

```

        dp_2[i] = dp_2[i - 1];
        dp_0[i] = dp_0[i - 1] + 1;
    }
}
for (var i = 1; i <= n; i++) {
    dp[i] = dp[i - 1] + dp_1[i];
}
std::cout << dp[n] << '\n';
}

}

// g++ -std=c++20 Main.cpp -o Main && Main
int main() {
    std::ios::sync_with_stdio(0);
    std::cin.tie(0); std::cout.tie(0);
    //int t; std::cin >> t; while (t --)
    solve();
    return 0;
}

```