模拟赛讲评

disangan233
中国人民大学
上海洛谷网络科技有限公司

洛谷
www.luogu.com.cn

选择题

1. 在断电后，其中的数据不会变化的是？（B）

RAM 存储器在断电后数据会变化，ROM 存储器在断电后数据不会变化

寄存器与高速缓存是 CPU 结构，断电后数据会变化

选择题

2. 用来全面管理计算机硬件和软件资源的软件叫？（A）

用来全面管理计算机硬件和软件资源的软件叫操作系统

中央处理器（CPU）是硬件

选择题

3. 下列协议中与文件传输有关的是？（C）

A. HTTP：Hypertext Transfer Protocol 超文本传输协议
B. SMTP：Simple Mail Transfer Protocol 简单邮件传输协议
C. FTP： File Transfer Protocol 文件传输协议
D. TCP：Transmission Control Protocol 传输控制协议

选择题

4. 定义两个字符串 a, b 本质不同，当且仅当长度不相等或 a 的第 i 个字符不等于 b 的第 i 个字符。字符串 $S =$ abaab 本质不同的非空子串（不含本身）数为？（D）

长度为 1：a，b

长度为 2：aa，ab，ba

长度为 3：aab，aba，baa

长度为 4：abaa，baab

长度为 5：abaab

共 11 个非空子串，不含本身有 10 个非空子串

选择题

5. 以下排序算法中，最坏复杂度为 $O(n \log n)$ 的是？（C）

A. 快速排序：期望 $O(n \log n)$
B. 桶排序：稳定 $O(n)$
C. 归并排序：稳定 $O(n \log n)$
D. 基数排序：稳定 $O(n)$

选择题

6. 一棵具有 32 层的完全二叉树中结点数至少为？（A）

完全二叉树最后一层要求向左对齐，最少的情况下只需要有 1 个

于是结点数至少为

$$2^0 + 2^1 + \cdots + 2^{30} + 1 = 2^{31}$$

选择题

7. 链表具有的特点是？（D）

链表只能顺序访问，不需要事先估计存储空间（new）
插入删除元素只需要更改相邻元素的指针，不需要移动元素

选择题

8. 对于一个八进制数 13.64 转换成四进制应该是多少？（B）

$(13.64)_8=(001\ 011.110\ 100)_2=(10\ 11.11\ 01)_2=(23.31)_4$

选择题

9. int 型整数 x 的后 k 位二进制位组成的数是？（C）

将 x 按位与上 0x111…11（共 k 个 1）即可

1<<k 第 1 位是 1，后 k 位是 0，减去 1 即可得到

所以答案是 x&(1<<k)-1，注意运算符优先级最高的是减法

选择题

10. 将 32 位无符号整型强制转换为 32 位有符号整型后，不可能？（B）

可能会溢出成负数，但是负数再次强制转换时会变成原数

不可能的是大于原数

选择题

11. 有一种用六位数表示日期的方法，如 230807 表示 2023 年的 8 月 7 日，也就是从左到右第 1,2 位表示年，第 3,4 位表示月，第 5,6 位表示日。如果用这种方法表示 2023 年的日期，那么全年中 6 个数字都不相同的日期共有多少天？（A）

月只有 0,1 两种开头，日只有 0,1,2,3 四种开头，且日无法取到 2,3 作为开头

如果月以 0 开头，日就只能以 1 开头，此时剩下 6 种数字可以自由选择，方案为 $C_6^5 = 30$

如果月以 1 开头，会发现 10,11,12 三个月都不能取，所以月只能以 0 开头

于是共有 30 天

选择题

12. 二叉树 T，已知其中序遍历是 7 8 1 4 5 2 3 6（数字为结点的编号，以下同），后序遍历是 8 7 1 5 3 6 2 4，则该二叉树的先序遍历是？（A）

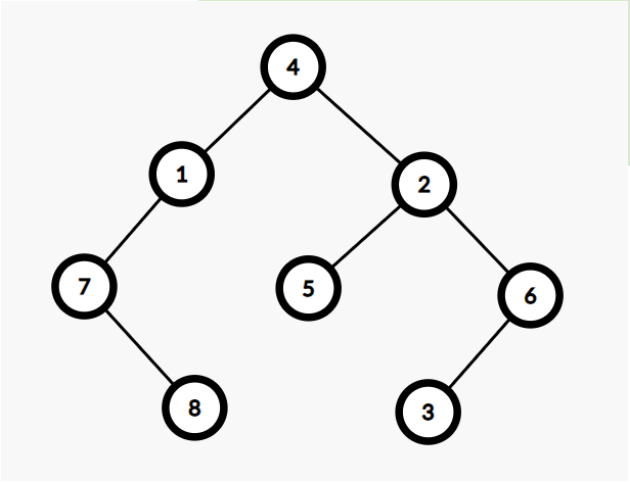
后序遍历可以确定根结点是 4

中序遍历得出左子树中序为 7 8 1，右子树中序为 5 2 3 6

再次通过后序遍历确定根节点，中序遍历划分子树，递归操作即可

于是画一下图就可以求出原二叉树

先序遍历为 4 1 7 8 2 5 6 3



选择题

13. 分辨率为 1920×1080 、 32 位色的位图，存储图像信息所需的空间约为？（A）

bmp 格式下位深为 8，所以是除以 8

$$1920 \times 1080 \times 32 \div 8 = 8{,}294{,}400\ \mathrm{B} = 7.91\ \mathrm{MiB}$$

选择题

14. 在调用 bits/stdc++.h 头文件后，以下哪种输出方式一定无法将类型为 double 的浮点数 x 四舍五入保留六位小数输出？（C）

%f,%lf 都可以用来输出 double，默认保留六位小数

cout<<fixed<<setprecision(6)<<x 也是一种输出方法，在 iomanip 中

但是用 printf("%.6Lf",x) 不会输出正确结果

跟不能用 %lld 输出 int 是一样的

选择题

15.在 CSP-J 2022 初赛第 17 题中，考察了 `numeric_limits<int>::max()` 的使用，这个模板类包含在哪个头文件中？（D）

包含在 `limits.h` 中

阅读程序 1

原题：P9160

给定一个多重集合（可重集合）S，求一个所有元素之和最大的多重集合 T，满足 T 是 S 的一个真子集，且对于 $\forall x \in S$，x 的前驱要么不存在，要么在 T 中。求出大小和其中元素之和。

所有元素之和最大且是真子集，大小为 $n-1$

考虑贪心，需要在满足条件的情况下，选尽可能小的一个元素不在 T 中

发现重复的元素删去其中一个，并不会影响前驱的存在性，所以删去最小的重复元素即可

如果不存在重复元素，只能删去最大的元素

排序后枚举求解

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

int n, s[100005];
long long sum = 0;

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> s[i], sum += s[i];
    sort(s + 1, s + n + 1);
    for (int i = 2; i <= n; i++)
        if (s[i] == s[i - 1] || i == n) {
            sum -= s[i];
            break;
        }
    cout << n - 1 << " " << sum << endl;
    return 0;
}
```

阅读程序 1 判断题

1. 该算法的时间复杂度是 $O(n \log n)$。（T）

复杂度瓶颈在于 sort，循环是 $O(n)$ 的。

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

int n, s[100005];
long long sum = 0;

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> s[i], sum += s[i];
    sort(s + 1, s + n + 1);
    for (int i = 2; i <= n; i++)
        if (s[i] == s[i - 1] || i == n) {
            sum -= s[i];
            break;
        }
    cout << n - 1 << " " << sum << endl;
    return 0;
}
```

阅读程序 1 判断题

2. 该程序输出的 sum 值不超过 n^2。（F）

sum 值等于 $n-1$ 个元素的和，所以应该是 $10^9(n-1)$，而 $n \leq 10^5$

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

int n, s[100005];
long long sum = 0;

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> s[i], sum += s[i];
    sort(s + 1, s + n + 1);
    for (int i = 2; i <= n; i++)
        if (s[i] == s[i - 1] || i == n) {
            sum -= s[i];
            break;
        }
    cout << n - 1 << " " << sum << endl;
    return 0;
}
```

阅读程序 1 判断题

3. 对于满足 $s_1 \leq s_2 \leq \cdots \leq s_n$ 的输入数据，将 s_1 的值改为 s_2 后输出结果不变。（F）

如果原本不存在相等的元素，结果就会改变

答案会变为 $s_2 + s_3 + \cdots + s_n$

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

int n, s[100005];
long long sum = 0;

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> s[i], sum += s[i];
    sort(s + 1, s + n + 1);
    for (int i = 2; i <= n; i++)
        if (s[i] == s[i - 1] || i == n) {
            sum -= s[i];
            break;
        }
    cout << n - 1 << " " << sum << endl;
    return 0;
}
```

阅读程序 1 判断题

4. 若输入的 s_i 互不相等，sum 的值会等于 $s_1 + s_2 + \cdots + s_{n-1}$。（F）

输入的 s_i 无序，排完序后 sum 的值才会等于 $s_1 + s_2 + \cdots + s_{n-1}$

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

int n, s[100005];
long long sum = 0;

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> s[i], sum += s[i];
    sort(s + 1, s + n + 1);
    for (int i = 2; i <= n; i++)
        if (s[i] == s[i - 1] || i == n) {
            sum -= s[i];
            break;
        }
    cout << n - 1 << " " << sum << endl;
    return 0;
}
```

阅读程序 1 单选题

1. 当输入为 4 4 5 1 4 时，输出结果为？（B）

去掉重复元素 4，元素的和为 $1+4+5=10$

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

int n, s[100005];
long long sum = 0;

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> s[i], sum += s[i];
    sort(s + 1, s + n + 1);
    for (int i = 2; i <= n; i++)
        if (s[i] == s[i - 1] || i == n) {
            sum -= s[i];
            break;
        }
    cout << n - 1 << " " << sum << endl;
    return 0;
}
```

阅读程序 1 单选题

2. 当输入为 6 1 4 2 8 5 7 时，输出结果为？（D）

没有重复元素，于是去掉最大的元素 8

元素的和为 $1+2+4+5+7=19$

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

int n, s[100005];
long long sum = 0;

int main() {
    cin >> n;
    for (int i = 1; i <= n; i++)
        cin >> s[i], sum += s[i];
    sort(s + 1, s + n + 1);
    for (int i = 2; i <= n; i++)
        if (s[i] == s[i - 1] || i == n) {
            sum -= s[i];
            break;
        }
    cout << n - 1 << " " << sum << endl;
    return 0;
}
```

阅读程序 2

原题：P8646

给定 n 种货币面值 a_n，问有多少种钱数无法被凑出来，$n, a_i \le 100$。

一个显然的结论是，$\gcd(a_1, a_2, \cdots, a_n) > 1$ 时必然有无限多种

然后用背包 dp 一下即可，循环节是 a_i^2，所以数组大小开到 n^2 级别即可

直接看代码就能知道题意的题

```
#include <bits/stdc++.h>
using namespace std;
int read() {
    int x;
    cin >> x;
    return x;
}
int a[105], dp[1000005];
int main() {
    int n = read();
    for (int i = 1; i <= n; i++)
        a[i] = read();
    int t = a[1];
    for (int i = 2; i <= n; i++)
        t = __gcd(t, a[i]);
    if (t > 1) {
        cout << "INF";
        return 0;
    }
    dp[0] = 1;
    for (int i = 0; i <= 1000000; i++)
        if (dp[i]) for (int j = 1; j <= n; j++) dp[i + a[j]] = 1;
    int ans = 0;
    for (int i = 1; i <= 1000000; i++)
        if (!dp[i])
            ans++;
    cout << ans << endl;
    return 0;
}
```

阅读程序 2 判断题

1. 代码中实现的 read() 的读入速度快于 scanf。（F）

这里的 read() 就是 cin，在不关闭同步流的情况下，速度慢于 scanf。

```
#include <bits/stdc++.h>
using namespace std;
int read() {
    int x;
    cin >> x;
    return x;
}
int a[105], dp[1000005];
int main() {
    int n = read();
    for (int i = 1; i <= n; i++)
        a[i] = read();
    int t = a[1];
    for (int i = 2; i <= n; i++)
        t = __gcd(t, a[i]);
    if (t > 1) {
        cout << "INF";
        return 0;
    }
    dp[0] = 1;
    for (int i = 0; i <= 1000000; i++)
        if (dp[i]) for (int j = 1; j <= n; j++) dp[i + a[j]] = 1;
    int ans = 0;
    for (int i = 1; i <= 1000000; i++)
        if (!dp[i])
            ans++;
    cout << ans << endl;
    return 0;
}
```

阅读程序 2 判断题

2. 如果读入的 a_i 全是质数，程序一定不会输出 INF。（F）

全是同一个质数，就会有 t>1

```
#include <bits/stdc++.h>
using namespace std;
int read() {
    int x;
    cin >> x;
    return x;
}
int a[105], dp[1000005];
int main() {
    int n = read();
    for (int i = 1; i <= n; i++)
        a[i] = read();
    int t = a[1];
    for (int i = 2; i <= n; i++)
        t = __gcd(t, a[i]);
    if (t > 1) {
        cout << "INF";
        return 0;
    }
    dp[0] = 1;
    for (int i = 0; i <= 1000000; i++)
        if (dp[i]) for (int j = 1; j <= n; j++) dp[i + a[j]] = 1;
    int ans = 0;
    for (int i = 1; i <= 1000000; i++)
        if (!dp[i])
            ans++;
    cout << ans << endl;
    return 0;
}
```

阅读程序 2 判断题

3. 存在一种合法输入数据，使得程序输出 0。（T）

只要有 $a_i = 1$，就可以凑出任何面值

```
#include <bits/stdc++.h>
using namespace std;
int read() {
    int x;
    cin >> x;
    return x;
}
int a[105], dp[1000005];
int main() {
    int n = read();
    for (int i = 1; i <= n; i++)
        a[i] = read();
    int t = a[1];
    for (int i = 2; i <= n; i++)
        t = __gcd(t, a[i]);
    if (t > 1) {
        cout << "INF";
        return 0;
    }
    dp[0] = 1;
    for (int i = 0; i <= 1000000; i++)
        if (dp[i]) for (int j = 1; j <= n; j++) dp[i + a[j]] = 1;
    int ans = 0;
    for (int i = 1; i <= 1000000; i++)
        if (!dp[i])
            ans++;
    cout << ans << endl;
    return 0;
}
```

阅读程序 2 单选题

1. 当输入为 2 4 5 时，输出结果为？（C）

求得 gcd(4,5)=1，于是不是 INF

手动模拟一下背包，发现只有 1,2,3,6,7,11 凑不出来

```
#include <bits/stdc++.h>
using namespace std;
int read() {
    int x;
    cin >> x;
    return x;
}
int a[105], dp[1000005];
int main() {
    int n = read();
    for (int i = 1; i <= n; i++)
        a[i] = read();
    int t = a[1];
    for (int i = 2; i <= n; i++)
        t = __gcd(t, a[i]);
    if (t > 1) {
        cout << "INF";
        return 0;
    }
    dp[0] = 1;
    for (int i = 0; i <= 1000000; i++)
        if (dp[i]) for (int j = 1; j <= n; j++) dp[i + a[j]] = 1;
    int ans = 0;
    for (int i = 1; i <= 1000000; i++)
        if (!dp[i])
            ans++;
    cout << ans << endl;
    return 0;
}
```

阅读程序 2 单选题

2. 如果读入的 a_i 全是偶数，输出结果为？（D）

那么 t>1，所有奇数面值都无法被凑出来，所以输出 INF

```
#include <bits/stdc++.h>
using namespace std;
int read() {
    int x;
    cin >> x;
    return x;
}
int a[105], dp[1000005];
int main() {
    int n = read();
    for (int i = 1; i <= n; i++)
        a[i] = read();
    int t = a[1];
    for (int i = 2; i <= n; i++)
        t = __gcd(t, a[i]);
    if (t > 1) {
        cout << "INF";
        return 0;
    }
    dp[0] = 1;
    for (int i = 0; i <= 1000000; i++)
        if (dp[i]) for (int j = 1; j <= n; j++) dp[i + a[j]] = 1;
    int ans = 0;
    for (int i = 1; i <= 1000000; i++)
        if (!dp[i])
            ans++;
    cout << ans << endl;
    return 0;
}
```

阅读程序 2 单选题

3. 令 dp 数组的大小为 m，代码的时间复杂度是？（C）

复杂度瓶颈在于背包，时间复杂度为 $O(nm)$。

```
#include <bits/stdc++.h>
using namespace std;
int read() {
    int x;
    cin >> x;
    return x;
}
int a[105], dp[1000005];
int main() {
    int n = read();
    for (int i = 1; i <= n; i++)
        a[i] = read();
    int t = a[1];
    for (int i = 2; i <= n; i++)
        t = __gcd(t, a[i]);
    if (t > 1) {
        cout << "INF";
        return 0;
    }
    dp[0] = 1;
    for (int i = 0; i <= 1000000; i++)
        if (dp[i]) for (int j = 1; j <= n; j++) dp[i + a[j]] = 1;
    int ans = 0;
    for (int i = 1; i <= 1000000; i++)
        if (!dp[i])
            ans++;
    cout << ans << endl;
    return 0;
}
```

阅读程序 3

原题：CF1705C

给定长度为 n 的字符串 s，进行 c 次操作，每次操作将 $s_l \dots s_r$ 复制到字符串尾。全部操作结束后有 q 组询问，每次询问字符串 s 的第 k 位。$\sum n \le 2 \times 10^5, \sum q \le 10^4, c \le 40$，$k, l_i, r_i \le 10^{18}$，$l_i, r_i \le n + \sum_{1 \le j < i} (r_j - l_j + 1)$。

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
struct node {
    int l, r;
} ch[101];
int t, n, sum[101], k, c, q;
char a[200010];
int query(int m) {
    if (m <= n)
        return m;
    int z = lower_bound(sum + 1, sum + c + 1, m) - sum;
    if (sum[z] > m) {
        z--;
        return query(m - sum[z] + ch[z + 1].l - 1);
    }
    return query(ch[z].r);
}
```

```
signed main() {
    cin >> t;
    while (t--) {
        cin >> n >> c >> q;
        cin >> a + 1;
        sum[0] = n;
        for (int i = 1; i <= c; i++)
            cin >> ch[i].l >> ch[i].r,
                sum[i] = sum[i - 1] + ch[i].r - ch[i].l + 1;
        while (q--) {
            cin >> k;
            cout << a[query(k)] << "\n";
        }
    }
    return 0;
}
```

阅读程序 3

操作完后 s 的长度是 $10^{18} \times c$ 的，肯定无法直接存储

考虑到 c 很小，所以用 sum_i 来记录第 i 次操作后的长度

从 sum_c 开始倒着模拟字符串的添加过程，用二分找到第一个 $\geq k$ 的 sum_i，推算出这次操作之前的长度，然后递归继续询问，直到长度 $\leq k$ 为止

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
struct node {
    int l, r;
} ch[101];
int t, n, sum[101], k, c, q;
char a[200010];
int query(int m) {
    if (m <= n)
        return m;
    int z = lower_bound(sum + 1, sum + c + 1, m) - sum;
    if (sum[z] > m) {
        z--;
        return query(m - sum[z] + ch[z + 1].l - 1);
    }
    return query(ch[z].r);
}
```

```
signed main() {
    cin >> t;
    while (t--) {
        cin >> n >> c >> q;
        cin >> a + 1;
        sum[0] = n;
        for (int i = 1; i <= c; i++)
            cin >> ch[i].l >> ch[i].r,
                sum[i] = sum[i - 1] + ch[i].r - ch[i].l + 1;
        while (q--) {
            cin >> k;
            cout << a[query(k)] << "\n";
        }
    }
    return 0;
}
```

阅读程序 3 判断题

1. sum[i] 的值可能会超过 long long 范围。（F）

$$\log_2 10^{18} - \log_2 200000 \approx 42.2$$

需要至少 43 次操作，字符串长度才会超过 10^{18}，所以 $sum_c \leq 2 \times 10^{18}$

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
struct node {
    int l, r;
} ch[101];
int t, n, sum[101], k, c, q;
char a[200010];
int query(int m) {
    if (m <= n)
        return m;
    int z = lower_bound(sum + 1, sum + c + 1, m) - sum;
    if (sum[z] > m) {
        z--;
        return query(m - sum[z] + ch[z + 1].l - 1);
    }
    return query(ch[z].r);
}
```

```
signed main() {
    cin >> t;
    while (t--) {
        cin >> n >> c >> q;
        cin >> a + 1;
        sum[0] = n;
        for (int i = 1; i <= c; i++)
            cin >> ch[i].l >> ch[i].r,
                sum[i] = sum[i - 1] + ch[i].r - ch[i].l + 1;
        while (q--) {
            cin >> k;
            cout << a[query(k)] << "\n";
        }
    }
    return 0;
}
```

阅读程序 3 判断题

2. query 函数最多只会递归调用 c 次。（T）

对于最坏的情况，每一次递归只会让 lower_bound 的结果下标减 1

那么最多只会递归调用 c 次

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
struct node {
    int l, r;
} ch[101];
int t, n, sum[101], k, c, q;
char a[200010];
int query(int m) {
    if (m <= n)
        return m;
    int z = lower_bound(sum + 1, sum + c + 1, m) - sum;
    if (sum[z] > m) {
        z--;
        return query(m - sum[z] + ch[z + 1].l - 1);
    }
    return query(ch[z].r);
}
```

```
signed main() {
    cin >> t;
    while (t--) {
        cin >> n >> c >> q;
        cin >> a + 1;
        sum[0] = n;
        for (int i = 1; i <= c; i++)
            cin >> ch[i].l >> ch[i].r,
                sum[i] = sum[i - 1] + ch[i].r - ch[i].l + 1;
        while (q--) {
            cin >> k;
            cout << a[query(k)] << "\n";
        }
    }
    return 0;
}
```

阅读程序 3 判断题

3. 将程序中的 lower_bound 改为 upper_bound 并删去 z--，程序输出不变。（F）

upper_bound 无法处理某一次操作后的最后一个字符，会导致答案改变

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
struct node {
    int l, r;
} ch[101];
int t, n, sum[101], k, c, q;
char a[200010];
int query(int m) {
    if (m <= n)
        return m;
    int z = lower_bound(sum + 1, sum + c + 1, m) - sum;
    if (sum[z] > m) {
        z--;
        return query(m - sum[z] + ch[z + 1].l - 1);
    }
    return query(ch[z].r);
}
```

```
signed main() {
    cin >> t;
    while (t--) {
        cin >> n >> c >> q;
        cin >> a + 1;
        sum[0] = n;
        for (int i = 1; i <= c; i++)
            cin >> ch[i].l >> ch[i].r,
                sum[i] = sum[i - 1] + ch[i].r - ch[i].l + 1;
        while (q--) {
            cin >> k;
            cout << a[query(k)] << "\n";
        }
    }
    return 0;
}
```

阅读程序 3 单选题

1. 以下代码不能替换 cin>>a+1 的是？（A）

(a+i) 是 a[i]，而 a+i 是以 i 为下标的 char 指针

&(*(a+1)) 就是 a+1，而 for(int i=1;i<=n;i++) cin>>a+i; 是错的

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
struct node {
    int l, r;
} ch[101];
int t, n, sum[101], k, c, q;
char a[200010];
int query(int m) {
    if (m <= n)
        return m;
    int z = lower_bound(sum + 1, sum + c + 1, m) - sum;
    if (sum[z] > m) {
        z--;
        return query(m - sum[z] + ch[z + 1].l - 1);
    }
    return query(ch[z].r);
}
```

```
signed main() {
    cin >> t;
    while (t--) {
        cin >> n >> c >> q;
        cin >> a + 1;
        sum[0] = n;
        for (int i = 1; i <= c; i++)
            cin >> ch[i].l >> ch[i].r,
                sum[i] = sum[i - 1] + ch[i].r - ch[i].l + 1;
        while (q--) {
            cin >> k;
            cout << a[query(k)] << "\n";
        }
    }
    return 0;
}
```

阅读程序 3 单选题

2. 程序运行的时间复杂度是？（A）

由判断题得：query 至多只会递归 c 次，每一次询问是 $O(c \log c)$ 的总询问次数是 $\sum q$，跟 t 无关，所以时间复杂度为 $O(c \log c \sum q)$

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
struct node {
    int l, r;
} ch[101];
int t, n, sum[101], k, c, q;
char a[200010];
int query(int m) {
    if (m <= n)
        return m;
    int z = lower_bound(sum + 1, sum + c + 1, m) - sum;
    if (sum[z] > m) {
        z--;
        return query(m - sum[z] + ch[z + 1].l - 1);
    }
    return query(ch[z].r);
}
```

```
signed main() {
    cin >> t;
    while (t--) {
        cin >> n >> c >> q;
        cin >> a + 1;
        sum[0] = n;
        for (int i = 1; i <= c; i++)
            cin >> ch[i].l >> ch[i].r,
                sum[i] = sum[i - 1] + ch[i].r - ch[i].l + 1;
        while (q--) {
            cin >> k;
            cout << a[query(k)] << "\n";
        }
    }
    return 0;
}
```

阅读程序 3 单选题

3. 修改选项给定的部分数据范围，其他保持不变，在时限 $2s$ 的条件下，下列选项中原代码不能通过的数据范围是？（B）

修改 $t, \sum q$ 都可以在时限内通过，而 $c \leq 100$ 无法通过，虽然不会越界/超时，但是 sum[c] 会爆 long long

```
#include <bits/stdc++.h>
using namespace std;
#define int long long
struct node {
    int l, r;
} ch[101];
int t, n, sum[101], k, c, q;
char a[200010];
int query(int m) {
    if (m <= n)
        return m;
    int z = lower_bound(sum + 1, sum + c + 1, m) - sum;
    if (sum[z] > m) {
        z--;
        return query(m - sum[z] + ch[z + 1].l - 1);
    }
    return query(ch[z].r);
}
```

```
signed main() {
    cin >> t;
    while (t--) {
        cin >> n >> c >> q;
        cin >> a + 1;
        sum[0] = n;
        for (int i = 1; i <= c; i++)
            cin >> ch[i].l >> ch[i].r,
                sum[i] = sum[i - 1] + ch[i].r - ch[i].l + 1;
        while (q--) {
            cin >> k;
            cout << a[query(k)] << "\n";
        }
    }
    return 0;
}
```

完善程序 1

（P8637）有 n 个瓶子，编号为 $1 \sim n$，现在将它们随机打乱为 $a_1, a_2, \ldots, a_n$。每次操作可以任意交换两个瓶子的位置，求让 n 个瓶子编号依次为 $1,2, \ldots, n$ 最少需要多少次交换。$n \leq 10000$。

对于一次交换，只有当两个瓶子恰好互相错位，才可以让两个瓶子回到原序

于是直接贪心模拟，从 1 开始将每一个编号为 i 的瓶子交换到位置 i 上

具体的证明是，错位关系建成的图等价于若干个环，模拟相当于取一个点开始遍历这个环

大小为 2 的环一次交换即可回到原序，其余的环在最后一次交换时一定是相互错位的，所以交换次数最少

完善程序 1

1. ① 处应当填入（ C ）

swap 函数需要引用来修改两个变量

A. int x, int y

B. int *x, int *y

C. int &x, int &y

D. int x[], int y[]

```
#include <bits/stdc++.h>
int n, a[10010], ans = 0;
void sWAp(____①____) {____②____;}
int main() {
    std::cin >> n;
    for (int i = 1; i <= n; i++) {
        std::cin >> a[i];
    }
    for (int i = 1; i <= n; i++) {
        if (____③____) {
            for (____④____) {
                if (____⑤____) {
                    sWAp(a[i], a[j]);
                    ans++;
                    break;
                }
            }
        }
    }
    std::cout << ans << std::endl;
    return 0;
}
```

完善程序 1

2. ② 处应当填入（A）

用异或也可以实现无中间变量交换两个数

B 是错的，B,C 都没有定义中间变量 t

没有调用 std 命名空间，应该 std::swap

A. x ^= y, y ^= x, x ^= y

B. x = t, t = x, y = t

C. t = x, x = y, y = t

D. swap(x, y)

```
#include <bits/stdc++.h>
int n, a[10010], ans = 0;
void sWAp(____①____) {____②____;}
int main() {
    std::cin >> n;
    for (int i = 1; i <= n; i++) {
        std::cin >> a[i];
    }
    for (int i = 1; i <= n; i++) {
        if (____③____) {
            for (____④____) {
                if (____⑤____) {
                    sWAp(a[i], a[j]);
                    ans++;
                    break;
                }
            }
        }
    }
    std::cout << ans << std::endl;
    return 0;
}
```

完善程序 1

3. ③ 处应当填入（ D ）

只有没有排好序的位置需要交换

A. a[i] == i

B. a[i] < a[i+1]

C. a[i] > i && a[i] < i

D. a[i] ^ i

```
#include <bits/stdc++.h>
int n, a[10010], ans = 0;
void sWAp(____①____) {____②____;}
int main() {
    std::cin >> n;
    for (int i = 1; i <= n; i++) {
        std::cin >> a[i];
    }
    for (int i = 1; i <= n; i++) {
        if (____③____) {
            for (____④____) {
                if (____⑤____) {
                    sWAp(a[i], a[j]);
                    ans++;
                    break;
                }
            }
        }
    }
    std::cout << ans << std::endl;
    return 0;
}
```

完善程序 1

4. ④ 处应当填入（ D ）

枚举所有数，找到编号为 i 的数与当前位置交换

前 i-1 位已经排好了，所以说从 i+1 开始枚举

A. `int j = 1; j < n; j++`

B. `int j = 1; j <= i; j++`

C. `int j = i; j < n; j++`

D. `int j = i + 1; j <= n; j++`

```
#include <bits/stdc++.h>
int n, a[10010], ans = 0;
void sWAp(____①____) {____②____;}
int main() {
    std::cin >> n;
    for (int i = 1; i <= n; i++) {
        std::cin >> a[i];
    }
    for (int i = 1; i <= n; i++) {
        if (____③____) {
            for (____④____) {
                if (____⑤____) {
                    sWAp(a[i], a[j]);
                    ans++;
                    break;
                }
            }
        }
    }
    std::cout << ans << std::endl;
    return 0;
}
```

完善程序 1

5. ⑤ 处应当填入（ B ）

让第 i 位有序，应该是枚举 a[j] == i

A. a[i] == j

B. a[j] == i

C. a[i] ^ j

D. a[j] ^ i

```
#include <bits/stdc++.h>
int n, a[10010], ans = 0;
void sWAp(____①____) {____②____;}
int main() {
    std::cin >> n;
    for (int i = 1; i <= n; i++) {
        std::cin >> a[i];
    }
    for (int i = 1; i <= n; i++) {
        if (____③____) {
            for (____④____) {
                if (____⑤____) {
                    sWAp(a[i], a[j]);
                    ans++;
                    break;
                }
            }
        }
    }
    std::cout << ans << std::endl;
    return 0;
}
```

完善程序 2

（CF611B）给定两个正整数 a, b，求出 $[a, b]$ 中的所有整数中，有多少数化为二进制后，二进制位只有一个 0。$1 \le a, b \le 10^{18}$。

直接搜索貌似是 $O(n)$ 的，但是合法的数并不会有那么多

于是直接构造二进制位只有一个 0 的数即可

使用搜索从高位搜索到低位，如果出现一个 0，之后只能填 1

如果含一个 0 的当前数在 $[a, b]$ 中，答案累加上 1，$> b$ 时结束递归

时间复杂度实际上是 $O(\log_{10} n)$ 的

完善程序 2

1. ① 处应当填入（A）

如果含一个 0 的当前数在 $[a,b]$ 中，答案累加上 1

if0 为 1 只有一种方案，代表已经有一个 0

A. x >= a && x <= b && if0

B. x >= a && x <= b && !if0

C. x >= a && x <= b

D. !(x <= a && x >= b)

```
#include <bits/stdc++.h>
using namespace std;
long long a, b, cnt = 0;
void dfs(long long x, bool if0) {
    if (____①____)
        ++cnt;
    if (____②____)
        return;
    if (if0)
        ____③____;
    else {
        dfs(x * 2, 1);
        ____④____;
    }
}
int main() {
    cin >> a >> b;
    ____⑤____;
    cout << cnt;
    return 0;
}
```

完善程序 2

2. ② 处应当填入（ D ）

当前数 $>b$ 时，对答案不会再有贡献，结束递归

A. `x < a || x > b`

B. `x < a && x > b`

C. `x < a`

D. `x > b`

```
#include <bits/stdc++.h>
using namespace std;
long long a, b, cnt = 0;
void dfs(long long x, bool if0) {
    if (_____①_____)
        ++cnt;
    if (_____②_____)
        return;
    if (if0)
        _____③_____;
    else {
        dfs(x * 2, 1);
        _____④_____;
    }
}
int main() {
    cin >> a >> b;
    _____⑤_____;
    cout << cnt;
    return 0;
}
```

完善程序 2

3. ③ 处应当填入（ D ）

已经含有一个 0 后，只能再新增二进制位 1

A. return

B. dfs(x * 2, 1)

C. dfs(x * 2 + 1, 0)

D. dfs(x * 2 + 1, 1)

```
#include <bits/stdc++.h>
using namespace std;
long long a, b, cnt = 0;
void dfs(long long x, bool if0) {
    if (____①____)
        ++cnt;
    if (____②____)
        return;
    if (if0)
        ____③____;
    else {
        dfs(x * 2, 1);
        ____④____;
    }
}
int main() {
    cin >> a >> b;
    ____⑤____;
    cout << cnt;
    return 0;
}
```

完善程序 2

4. ④ 处应当填入（C）

此时还不含 0，当前位有 0,1 两种构造方法

A. return

B. dfs(x * 2, 0)

C. dfs(x * 2 + 1, 0)

D. dfs(x * 2 + 1, 1)

```
#include <bits/stdc++.h>
using namespace std;
long long a, b, cnt = 0;
void dfs(long long x, bool if0) {
    if (____①____)
        ++cnt;
    if (____②____)
        return;
    if (if0)
        ____③____;
    else {
        dfs(x * 2, 1);
        ____④____;
    }
}
int main() {
    cin >> a >> b;
    ____⑤____;
    cout << cnt;
    return 0;
}
```

完善程序 2

5. ⑤ 处应当填入（C）

最高位一定是 1，所以初始 x 为 1

而 if0 初始应该为 0，代表还没含有二进制位 0

A. dfs(0, 0)

B. dfs(0, 1)

C. dfs(1, 0)

D. dfs(1, 1)

```
#include <bits/stdc++.h>
using namespace std;
long long a, b, cnt = 0;
void dfs(long long x, bool if0) {
    if (____①____)
        ++cnt;
    if (____②____)
        return;
    if (if0)
        ____③____;
    else {
        dfs(x * 2, 1);
        ____④____;
    }
}
int main() {
    cin >> a >> b;
    ____⑤____;
    cout << cnt;
    return 0;
}
```