选择题训练讲评

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. 计算机直接识别和执行的语言是？（B）

高级语言（例如 C++）在运行时，先通过前端编译器编译成汇编语言，再通过后端编译器（汇编器）编译为机器语言

计算机直接识别和执行的语言是机器语言，机器语言只有 0 和 1

选择题

2. LAN 在计算机科学技术领域的常见含义是？（C）

A. 互联网 Internet
B. 万维网 WWW（World Wide Web）
C. 局域网 LAN（Local Area Network）
D. 无线局域网 WLAN（Wireless Local Area Network）

选择题

3. 以下排序算法中不是稳定排序的是？（A）

一个排序算法是稳定的，当且仅当在排序过程中，相等的两个数不会被交换

稳定排序：归并排序、冒泡排序、插入排序、基数排序

选择题

4. $n(n \geq 2)$ 个点的简单无向图最多有多少条边？（D）

简单图：无自环、无重边

所以最多是无向完全图，共 $\frac{n(n-1)}{2}$ 条边

选择题

5. 对于一个五进制数 14.32 转换成十进制应该是多少？（B）

$$5^1 + 4 \times 5^0 + 3 \times 5^{-1} + 2 \times 5^{-2} = 9.68$$

选择题

6. 对于一个八进制数 4762 转换成十六进制应该是多少？（D）

八进制转十六进制时，用二进制作为中间进制更加方便。

$(4762)_8=(100\ 111\ 110\ 010)_2=(1001\ 1111\ 0010)_2=(9F2)_{16}$

选择题

7. 设 `x=false,y=true,z=false`，以下逻辑运算表达式值为真的是？（B）

逻辑与 $\wedge$，逻辑或 $\vee$，逻辑非 $\neg$，优先级 $\neg > \wedge > \vee$

A. 有 $x \vee z \wedge y = 0 \vee (0 \wedge 1) = 0$

B. 有 $(x \vee y) \wedge (y \vee z) = (0 \vee 1) \wedge (1 \vee 0) = 1 \wedge 1 = 1$

C. 有 $x \wedge y \vee z \wedge y = (0 \wedge 1) \vee (0 \wedge 1) = 0 \vee 0 = 0$

D. 有 $x \wedge (z \vee y) \wedge z = 0 \wedge (0 \vee 1) \wedge 1 = 0 \wedge 1 \wedge 1 = 0$

选择题

8. 将 9 本相同的书放到 3 个不同的书柜里，不可以有书柜空着，有多少种不同的方式？（A）

首先将每个书柜里各放 1 本书，等价于 6 本相同的书放到 3 个不同的书柜里，且可空

由隔板法可得方案数为

$$C_{6+3-1}^{3-1} = C_8^2 = \frac{8 \times 7}{2 \times 1} = 28$$

选择题

9. 先进先出、后进后出的数据结构是？（D）

队列：先进先出、后进后出

栈：后进先出

选择题

10. 中缀表达式 9*3+2*(6+3)/4-5 转为后缀表达式是？（D）

注意第二项应该先 + 再 * 再 /，然后再执行前面的 +

入栈 9 3 后 *，再入栈 2 6 3 后 + *，然后再入栈 4 /

此时第二项也计算完毕，应该和第一项加起来，于是再入栈 + 5 -

得到后缀表达式为 9 3 * 2 6 3 + * 4 / + 5 -

选择题

11. 有六个元素 DABCFE 从左至右依次顺序进栈，在进栈过程中会有元素被弹出栈。问下列哪一个不可能是合法的出栈序列？（C）

A. BACEFD，可能

B. DCBFAE，可能

C. ACDBFE，不可能有 ACDB 的出栈序列

D. FCEBAD，可能

选择题

12. 一个包含 n 个结点的满二叉树，它的叶子结点数目为？（C）

满二叉树：除了最后一层叶子结点外，每一个非叶子结点都有两个儿子

所以每一层的结点数应该是 $1,2,2^2,2^3,\ldots$

设该满二叉树有 k 层，那么 $n = 1 + 2 + 2^2 + \cdots + 2^{k-1} = 2^k - 1$

所以叶子结点数为 $\frac{n+1}{2} = 2^{k-1}$

选择题

13. 有一段长度为 2 分钟的视频，分辨率是 2560×1440，每帧图像都是 32 位真彩色图像，在 10% 的压缩率下大小约为 3.955 GiB，视频的帧率为？（B）

原大小为 $3.955\ \text{GiB} \times 10 = 39.55\ \text{GiB} = (39.55 \times 1024^3)\ \text{B} = 42466489139\ \text{B}$

32 位色为 $(32 \div 8)\ \text{B} = 4\ \text{B}$，时长为 2×60

$42466489139 \div 2560 \div 1440 \div (32 \div 8) \div (2 \times 60) = 24\ \text{Hz}$

选择题

14. C++ 语言中，定义 rd() 为**等概率**随机生成 unsigned int 范围内的一个整数，那么**等概率**随机生成 unsigned long long 范围内的一个整数应该为？（C）

A. 1ull*rd()*rd()，生成不是等概率的，比如对于 7 只有 1*7 和 7*1 两种，但是对于 4 有 1*4、2*2 和 4*1 三种

B. rd()<<32|rd()，unsigned int 左移 32 位会直接变成 0

C. 1ull*rd()<<32|rd()，正确

D. rd()+rd()，无法生成 unsigned long long 范围内的数

选择题

15. 将 3 个红球、4 个绿球、3 个白球排成一行，有几种不同的排法？（A）

总共有 10 个位置，先选择红球的位置再选择绿球的位置

$C_{10}^{3} \times C_{10-3}^{4} = 4200$

阅读程序

原题：P8667

给定 a_n, b_n, c_n 三个数组，求有多少三元组 (i, j, k) 满足 $1 \le i, j, k \le n$ 且 $a_i < b_j < c_k$

二分题，但是没有用 lower_bound 和 upper_bound，用的 2-pointer

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
typedef long long LL;
int a[N], b[N], c[N];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &b[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &c[i]);
    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    sort(c + 1, c + n + 1);
    LL ans = 0;
    int cnta = 1, cntc = 1;
    for (int i = 1; i <= n; i++) {
        while (cnta <= n && a[cnta] < b[i])
            cnta++;
        while (cntc <= n && c[cntc] <= b[i])
            cntc++;
        ans += (LL)(cnta - 1) * (n - cntc + 1);
    }
    cout << ans;
    return 0;
}
```

阅读程序

做法是先对 a_n, b_n, c_n 排序，然后枚举 b_j，计算 i, k 的方案数

可以排序后对 a_n, c_n 二分，这里是维护双指针 cnta,cntc，实际上是维护 $< b_j$ 的 i 和 $> b_j$ 的 k 的个数，相乘即是当前 j 对答案的贡献

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
typedef long long LL;
int a[N], b[N], c[N];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &b[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &c[i]);
    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    sort(c + 1, c + n + 1);
    LL ans = 0;
    int cnta = 1, cntc = 1;
    for (int i = 1; i <= n; i++) {
        while (cnta <= n && a[cnta] < b[i])
            cnta++;
        while (cntc <= n && c[cntc] <= b[i])
            cntc++;
        ans += (LL)(cnta - 1) * (n - cntc + 1);
    }
    cout << ans;
    return 0;
}
```

阅读程序 判断题

1. 该算法的时间复杂度是 $O(n^2)$。（F）

sort 的复杂度是 $O(n \log n)$，双指针是线性的，所以复杂度是 $O(n \log n)$。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
typedef long long LL;
int a[N], b[N], c[N];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &b[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &c[i]);
    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    sort(c + 1, c + n + 1);
    LL ans = 0;
    int cnta = 1, cntc = 1;
    for (int i = 1; i <= n; i++) {
        while (cnta <= n && a[cnta] < b[i])
            cnta++;
        while (cntc <= n && c[cntc] <= b[i])
            cntc++;
        ans += (LL)(cnta - 1) * (n - cntc + 1);
    }
    cout << ans;
    return 0;
}
```

阅读程序 判断题

2. 该程序输出的答案不超过 n^3 。（T）

三元组每一个数都可以取到 1~n，即使任意三元组都满足，答案也是 n^3。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
typedef long long LL;
int a[N], b[N], c[N];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &b[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &c[i]);
    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    sort(c + 1, c + n + 1);
    LL ans = 0;
    int cnta = 1, cntc = 1;
    for (int i = 1; i <= n; i++) {
        while (cnta <= n && a[cnta] < b[i])
            cnta++;
        while (cntc <= n && c[cntc] <= b[i])
            cntc++;
        ans += (LL)(cnta - 1) * (n - cntc + 1);
    }
    cout << ans;
    return 0;
}
```

阅读程序 判断题

3. 代码运用了二分查找来计算 cnta 和 cntc。（F）

用的是 2-pointer 来计算

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
typedef long long LL;
int a[N], b[N], c[N];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &b[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &c[i]);
    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    sort(c + 1, c + n + 1);
    LL ans = 0;
    int cnta = 1, cntc = 1;
    for (int i = 1; i <= n; i++) {
        while (cnta <= n && a[cnta] < b[i])
            cnta++;
        while (cntc <= n && c[cntc] <= b[i])
            cntc++;
        ans += (LL)(cnta - 1) * (n - cntc + 1);
    }
    cout << ans;
    return 0;
}
```

阅读程序 判断题

4. 如果 n 的范围是 $n \leq 2000$，那么 ans 不需要开 long long。（F）

$$2000^3 = 8 \times 10^9 \approx 2^{34}$$

所以需要开 long long。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
typedef long long LL;
int a[N], b[N], c[N];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &b[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &c[i]);
    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    sort(c + 1, c + n + 1);
    LL ans = 0;
    int cnta = 1, cntc = 1;
    for (int i = 1; i <= n; i++) {
        while (cnta <= n && a[cnta] < b[i])
            cnta++;
        while (cntc <= n && c[cntc] <= b[i])
            cntc++;
        ans += (LL)(cnta - 1) * (n - cntc + 1);
    }
    cout << ans;
    return 0;
}
```

阅读程序 单选题

1. 将代码中的 while 循环替换成 lower_bound 和 upper_bound，总时间复杂度为？（B）

复杂度瓶颈在于 sort，所以依然是 $O(n \log n)$。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
typedef long long LL;
int a[N], b[N], c[N];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &b[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &c[i]);
    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    sort(c + 1, c + n + 1);
    LL ans = 0;
    int cnta = 1, cntc = 1;
    for (int i = 1; i <= n; i++) {
        while (cnta <= n && a[cnta] < b[i])
            cnta++;
        while (cntc <= n && c[cntc] <= b[i])
            cntc++;
        ans += (LL)(cnta - 1) * (n - cntc + 1);
    }
    cout << ans;
    return 0;
}
```

阅读程序 单选题

2. 当输入为 3 1 1 1 2 2 2 3 3 3 时，输出结果为？（A）

任何一组三元组都满足条件，所以输出结果为 $3^3 = 27$。

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e5 + 5;
typedef long long LL;
int a[N], b[N], c[N];
int main() {
    int n;
    cin >> n;
    for (int i = 1; i <= n; i++)
        scanf("%d", &a[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &b[i]);
    for (int i = 1; i <= n; i++)
        scanf("%d", &c[i]);
    sort(a + 1, a + n + 1);
    sort(b + 1, b + n + 1);
    sort(c + 1, c + n + 1);
    LL ans = 0;
    int cnta = 1, cntc = 1;
    for (int i = 1; i <= n; i++) {
        while (cnta <= n && a[cnta] < b[i])
            cnta++;
        while (cntc <= n && c[cntc] <= b[i])
            cntc++;
        ans += (LL)(cnta - 1) * (n - cntc + 1);
    }
    cout << ans;
    return 0;
}
```

完善程序

（P7588）定义双重素数为这样的素数：它的各位数字之和也是一个素数。T 组询问，每组询问给定一个闭区间 $[l,r]$，求该区间内双重素数的个数。$T \le 100, 1 \le l < r \le 10^8$。

区间内质数个数 $\pi(n) \sim \frac{n}{\ln n}$，对于每个质数处理一次的复杂度是 $O(\log_{10} n)$

于是线性筛求出 $1 \sim 10^8$ 内的所有质数，然后对于每个质数求是否是双重的

时间复杂度 $O(n)$，常数很小，可以通过

由于原题空间只给了 90 MiB，用 `bitset` 代替大小 10^8 的 `bool` 数组

`bitset` 原理是将每 w 个 `bool` 位压在一起，空间复杂度 $O\left(\frac{n}{w}\right)$

w 的大小取决于计算机系统，64 位下取 $w = 64$

`bitset` 这里可以直接当 `bool` 数组用

完善程序

1. ① 处应当填入（C）

线性筛枚举 i 的最小质因子 j，在 i%p[j]==0 时 break

A. i%p[j]

B. i*p[j]<=n

C. !(i%p[j])

D. i>p[j]

```
#include <bits/stdc++.h>
using namespace std;
const int n = 1e8;
int cnt, p[n / 10];
bitset<n + 1> vis;
void getprime() {
    for (int i = 2; i <= n; i++) {
        if (!vis[i])
            p[++cnt] = i;
        for (int j = 1; i * p[j] <= n; j++) {
            vis[i * p[j]] = 1;
            if (_____①_____)
                break;
        }
    }
}
```

完善程序

2. ② 处应当填入（ D ）

观察得出 calc 函数是在计算各位数字之和，从最低位加到最高位即可

A. ans=x

B. ans=x%10

C. ans+=x

D. ans+=x%10

```
int calc(int x) {
    int ans = 0;
    while (x) {
        ____②____;
        x /= 10;
    }
    return ans;
}
```

完善程序

3. ③ 处应当填入（ A ）

观察得出 getdprime 函数将素数中的双重素数筛出

对于素数 p[i]，各位数字之和 calc(p[i]) 也需要是素数

所以应该是 !vis[calc(p[i])]

A. !vis[calc(p[i])]

B. !vis[p[i]]

C. calc(p[i])

D. vis[calc(p[i])]

```
void getdprime() {
    int cntt = 0;
    for (int i = 1; i <= cnt; i++)
        if (_____③_____)
            p[++cntt] = p[i];
    cnt = cntt;
}
```

完善程序

4. ④ 处应当填入（ D ）

调用 getprime 和 getdprime 后，p[cntt] 内是已经筛好的双重素数，只需要查询 $[l,r]$ 内有多少双重素数即可

于是找到第一个 $> r$ 的双重素数和第一个 $\geq l$ 的双重素数，下标的差就是答案

用 upper_bound 找到第一个 > 某数的下标

用 lower_bound 找到第一个 ≥ 某数的下标

A. lower_bound(p+1,p+cnt+1,r)-p-1

B. lower_bound(p+1,p+cnt+1,r)-p

C. upper_bound(p+1,p+cnt+1,r)-p-1

D. upper_bound(p+1,p+cnt+1,r)-p

```
int main() {
    getprime();
    getdprime();
    int t;
    scanf("%d", &t);
    while (t--) {
        int l, r;
        scanf("%d%d", &l, &r);
        int ans1 = _____④_____;
        int ans2 = _____⑤_____;
        printf("%d\n", ans1 - ans2);
    }
    return 0;
}
```

完善程序

5. ⑤ 处应当填入（ B ）

调用 getprime 和 getdprime 后，p[cntt] 内是已经筛好的双重素数，只需要查询 $[l,r]$ 内有多少双重素数即可

于是找到第一个 $> r$ 的双重素数和第一个 $\geq l$ 的双重素数，下标的差就是答案

用 upper_bound 找到第一个 > 某数的下标

用 lower_bound 找到第一个 ≥ 某数的下标

A. lower_bound(p+1,p+cnt+1,l)-p-1

B. lower_bound(p+1,p+cnt+1,l)-p

C. upper_bound(p+1,p+cnt+1,l)-p-1

D. upper_bound(p+1,p+cnt+1,l)-p

```
int main() {
    getprime();
    getdprime();
    int t;
    scanf("%d", &t);
    while (t--) {
        int l, r;
        scanf("%d%d", &l, &r);
        int ans1 = _____④_____;
        int ans2 = _____⑤_____;
        printf("%d\n", ans1 - ans2);
    }
    return 0;
}
```