

扫码获取
答案解析

普及组 CSP-J 2025 初赛模拟卷 6

一、单项选择题 (共 15 题, 每题 2 分, 共计 30 分; 每题有且仅有一个正确选项)

1. 深度优先搜索时, 控制与记录搜索过程的数据结构是 ()。
A. 队列 B. 栈 C. 链表 D. 哈希表
2. 计算机的中央处理器的组成部件是 ()。
A. 控制器和存储器 B. 运算器和存储器
C. 控制器、存储器和运算器 D. 运算器和控制器
3. 一个正整数在十六进制下有 200 位, 则它在二进制下最多可能有 () 位。
A. 801 B. 798 C. 799 D. 800
4. 一个由 2025 个元素组成的数组已经从小到大排好序, 采用二分查找, 最多需要 () 次能够判断是否存在所查找的元素。
A. 2025 B. 12 C. 11 D. 10
5. 无向完全图 G 有 10 个顶点, 它有 () 条边。
A. 45 B. 90 C. 72 D. 36
6. 在 8 位二进制补码中, 10110110 表示的是十进制下的 ()。
A. -202 B. -74 C. 202 D. 74
7. 某市有 2025 名学生参加编程竞赛选拔, 试卷中有 20 道选择题, 每题答对得 5 分, 答错或者不答得 0 分, 那么至少有 () 名同学得分相同。
A. 99 B. 98 C. 97 D. 96
8. 以下哪个操作运算符优先级最高? ()
A. $\&\&$ B. $||$ C. $>>$ D. $++$

9. 如果根结点的深度是 1, 则一棵恰好有 2025 个叶子结点的二叉树的深度不可能是 ()。
- A. 11 B. 12 C. 13 D. 2025
10. 现代通用计算机之所以可以表示比较大或者比较小的浮点数, 是因为使用了 ()。
- A. 原码 B. 补码 C. 反码 D. 阶码
11. 在 C++ 语言中, 一个数组定义为 `int a[6] = {1, 2, 3, 4, 5, 6};`, 一个指针定义为 `int *p = &a[3];`, 则执行 `a[2] = *p;` 后, 数组 `a` 中的值会变为 ()。
- A. {1, 2, 4, 4, 5, 6} B. {2, 2, 3, 4, 5, 6}
C. {1, 2, 2, 4, 5, 6} D. {1, 2, 3, 4, 5, 6}
12. 下面的 C++ 代码执行后的输出是 ()。

```
#include <bits/stdc++.h>
using namespace std;
int print(int x)
{
    cout << x << "$";
    if(1==x || 2==x)
        return x;
    else
        return print(x-1) + print(x-2);
}
int main()
{
    cout<< print(4)<< endl;
    return 0;
}
```

- A. 4\$3\$2\$2\$4 B. 4\$3\$2\$2\$1\$5 C. 4\$3\$2\$1\$2\$4 D. 4\$3\$2\$1\$2\$5
13. 小明往一个图书馆送书, 第 1 天送 1 本, 第 2 天送 2 本, 第 3 天送 3 本……第 n 天送 n 本, 他准备累计送到图书馆的书的总数能整除 106 就停止, 那么小明应连续送 () 天。
- A. 50 B. 51 C. 52 D. 53

14. $7 + 77 + 777 + \dots + 77\dots77$ (共 2025 个连续的 7) 的末 2 位数是 ()。
- A. 45 B. 55 C. 65 D. 75
15. 在无重复数字的五位数 $a_1a_2a_3a_4a_5$ 中, 若 $a_1 < a_2$, $a_2 > a_3$, $a_3 < a_4$, $a_4 > a_5$, 则称该五位数为波形数, 如 89674 就是一个波形数, 由 1、2、3、4、5 组成的没有重复数字的五位数是波形数的概率是 ()。
- A. $1/5$ B. $1/6$ C. $2/15$ D. $1/3$

二、阅读程序 (程序输入不超过数组或字符串定义的范围; 判断题正确填√, 错误填×; 除特殊说明外, 判断题每题 1.5 分, 选择题每题 3 分, 共计 40 分)

(1)

```

01 #include <bits/stdc++.h>
02 using namespace std;
03 using i64 = long long;
04
05 i64 check(const string &s, int p)
06 {
07     return (p>=1 && ((s[p-1]-'0')*10 + (s[p]-'0'))%4==0)?1ll*p:0ll;
08 }
09
10 int main()
11 {
12
13     string s;
14     cin >> s;
15     i64 ans = 0;
16     for(int i = 0; i < s.length(); i++)
17         ans += ((s[i] - '0') % 4 == 0);
18     for(int i = s.length() - 1; i >= 0; i--)
19         ans += check(s, i);
20     cout << ans << endl;
21     cout << check("114514", 3) << endl;
22     return 0;
23 }

```

假设 $1 \leq s.length() \leq 3 \times 10^5$, 回答下面的问题。

■ 判断题

16. 若程序输入 124, 则程序输出 4 (换行) 0。 ()
17. 对于这段代码, `check("1234510", 2)` 的返回值为 2。 ()
18. 若将头文件 `<bits/stdc++.h>` 换为 `<cstdio>`, 程序依然可以正常运行。 ()

■ 选择题

19. 若输入 5810438174, 则输出是 ()。
- A. 7 (换行) 0 B. 8 (换行) 0
- C. 9 (换行) 0 D. 10 (换行) 0
20. 下面哪个选项是正确的? ()
- A. 把 `check` 函数中的第一个参数 `const` 去掉也可以正常运行
- B. 把 `check` 函数中的 `p >= 1` 去掉依然可以得到正确的答案
- C. `check` 函数用来判断由 `s[p-1]` 和 `s[p]` 组成的两位数是否为 4 的倍数
- D. 整段程序的时间复杂度为 $O(n\log n)$

邮
电

(2)

```
01 #include <bits/stdc++.h>
02 using namespace std;
03
04 string calc(string s, string t)
05 {
06     const int n = s.size();
07     if(t.size() > s.size())
08         return "";
09     unordered_map<char, int> mp;
10     int cnt = t.size();
11     for(auto v: t)
12         mp[v]++;
13     string ans;
14     int len = 0x3f3f3f3f;
15     for(int i = 0, j = 0; i < n; i++)
16     {
17         if(mp[s[i]] > 0)
18             cnt--;
19         mp[s[i]]--;
20         if(cnt == 0)
```

```

21     {
22         while(mp[s[j]] < 0)
23             mp[s[j++]]++;
24         int len = i - j + 1;
25         if(ans.empty() || ans.size() > len)
26             ans = s.substr(j, len);
27         mp[s[j++]]++;
28         cnt++;
29     }
30 }
31 return ans;
32 }
33
34 int main()
35 {
36     string s, t;
37     cin >> s >> t;
38     cout << calc(s, t) << endl;
39     return 0;
40 }

```

■ 判断题

21. 若输入 ADOBECODEBANC ABC, 则输出为 BANC。 ()
22. calc 函数中的变量 j 只会增大, 不会减小。 ()
23. (2 分) 若删除第 14 行中的 len 变量, 程序将不能正常运行。 ()

■ 选择题

24. 当输入为 a aa 时, 程序的输出为 ()。
- A. "a" B. "aa" C. "" D. "-1"
25. 若删除第 22 行代码, 则当输入为 cabwefgewcwaefgcf cae 时, 程序的输出为 ()。
- A. "cwae" B. "abwe" C. "cabwe" D. "fgewc"
26. (4 分) 设 $n=s.size()$, $m=t.size()$, 则这段程序的时间复杂度为 ()。
- A. $O(n)$ B. $O(m)$ C. $O(m+n)$ D. $O((m+n)\log n)$

(3)

```
01 #include <iostream>
02 #include <cstdio>
03 #include <algorithm>
04
05 using namespace std;
06
07 const int N = 1e5 + 10;
08 int n, s, cnt, ans, res;
09 int a[N], b[N];
10
11 bool check(int mid)
12 {
13     for(int i = 1; i <= n; i++)
14         b[i] = a[i] + i * mid;
15     sort(b + 1, b + 1 + n);
16     res = 0;
17     for(int i = 1; i <= mid && res <= s; i++)
18         res += b[i];
19     return res <= s;
20 }
21
22 int main()
23 {
24     scanf("%d%d", &n, &s);
25     for(int i = 1; i <= n; i++)
26         scanf("%d", &a[i]);
27     int l = 0, r = n;
28     while(l <= r)
29     {
30         int mid = (l + r) >> 1;
31         if(check(mid)) cnt = mid, ans = res, l = mid + 1;
32         else r = mid - 1;
33     }
34     printf("%d %d\n", cnt, ans);
35     return 0;
36 }
```

■ 判断题

27. 若输入 4 100 1 2 5 6, 则程序的输出为 4 54。

()

28. 对于任意的输入, cnt 的一个必定合法的取值为 n 。 ()
29. 这个程序的时间复杂度为 $O(n\log n)$ 。 ()

■ 选择题

30. 当输入为 3 11 2 3 5 时, 程序的输出为 ()。
- A. 1 11 B. 2 11 C. 3 8 D. 0 0
31. 代码中 check 函数的作用是什么? ()
- A. 判断当前数组是否有序
- B. 检查是否能从数组中选出 mid 个数, 使得它们的总和小于或等于 s
- C. 判断数组的所有元素是否大于某个值
- D. 计算数组元素的平均值
32. (4 分) 变量 cnt 和 ans 的作用分别是什么? ()
- A. cnt 记录满足条件的最大 mid 值, ans 记录对应的总和
- B. cnt 记录数组的长度, ans 记录数组中的最大值
- C. cnt 表示排序后的最小值索引, ans 记录当前结果的最小值
- D. cnt 表示满足条件的元素个数, ans 记录最终的目标值

三、完善程序 (单选题, 每小题 3 分, 共计 30 分)

(1) 题目描述:

有 T 组数据, 每组数据输入 n ($1 \leq n \leq 1 \times 10^4$) 和长为 n 的数组 a ($1 \leq a[i] \leq 1 \times 10^6$)。

你可以执行如下操作任意次: 选择 $a[i]$ 和 $a[j]$ ($i \neq j$), 以及 $a[i]$ 的一个因子 x 。

然后执行 $a[i] /= x$ 和 $a[j] *= x$ 。能否使 a 中所有元素都相同?

输出 YES 或 NO。

```
01 #include <bits/stdc++.h>
02 using namespace std;
03 #define maxn 500005
04 int a[maxn];
05 map<int,int>q;
06 void check(int x)
07     for (int i = 2;i <= ①;i ++)
08     {
09         while(x % i == 0)
```

```
10     {
11         q[i] ++;
12         x /= i;
13     }
14 }
15 if(x > 1) ②;
16 }
17 void solve()
18 {
19     int n;
20     cin >> n;
21     ③;
22     for (int i = 1; i <= n; i++)
23     {
24         scanf("%d", &a[i]);
25         ④;
26     }
27     for (auto i:q)
28     {
29         int k = i.second;
30         if(⑤)
31         {
32             cout << "NO"<<endl;
33             return;
34         }
35     }
36     cout << "YES"<<endl;
37     return;
38 }
39 int main()
40 {
41     int T = 1;
42     cin >> T;
43     while (T --)
44     {
45         solve();
46     }
47     return 0;
48 }
```


33. ①处应填 ()。

- A. `sqrt(x)` B. `pow(x, 2)` C. `pow(x, 3)` D. `log(x)`

34. ②处应填 ()。

- A. `q[x]--` B. `q[x] /= 2` C. `q[x]++` D. `q[x] *= 2`

35. ③处应填 ()。

- A. `q.clear()` B. `q.erase(q.begin())`
C. `q.swap(a)` D. `q.erase(q.end())`

36. ④处应填 ()。

- A. `check(q)` B. `check(a)`
C. `check(a[i])` D. `check(a[i-1])`

37. ⑤处应填 ()。

- A. `k / n == 0` B. `k / n != 0` C. `k % n == 0` D. `k % n != 0`

(2) 题目描述:

输入 n ($1 \leq n \leq 1 \times 10^5$), 表示有 n 座激光塔。然后输入 n 行, 每行有两个数 $p[i]$ ($0 \leq p[i] \leq 1 \times 10^6$) 和 $k[i]$ ($1 \leq k[i] \leq 1 \times 10^6$), 分别表示第 i 座激光塔的位置和威力。保证所有激光塔的位置互不相同。

游戏规则: 按照 $p[i]$ 从大到小依次激活激光塔。当一座激光塔被激活时, 它会摧毁它左侧所有满足 $p[i] - p[j] \leq k[i]$ 的激光塔 j 。被摧毁的激光塔无法被激活。

在游戏开始前, 你可以在最右边的激光塔的右侧, 再添加一座激光塔, 位置和威力由你决定。

你希望被摧毁的激光塔的数量尽量少。输出这个最小值。

```
01 #include <bits/stdc++.h>
02 using namespace std;
03 const int N=100000;
04 const int inf=2147483647;
05 struct beacon
06 {
07     int pos;
08     int power;
```

```
09 };
10 int n,ans=inf,dp[N+5];
11 beacon beacons[N+5];
12 bool cmp(beacon a,beacon b)
13 {
14     return ①;
15 }
16 int main()
17 {
18     cin>>n;
19     for(int i=1;i<=n;++i)
20     {
21         cin>>beacons[i].pos>>beacons[i].power;
22     }
23     sort(beacons+1,beacons+n+1,cmp);
24     ②;
25     for(int i=2;i<=n;++i)
26     {
27         beacon find;
28         find.pos=max(0, ③);
29         int destroy=④-(beacons+1);
30         dp[i]=dp[destroy];
31         dp[i]+=(i-destroy-1);
32     }
33     for(int i=1;i<=n;++i)
34     {
35         int destruction=⑤;
36         if(destruction<ans) ans=destruction;
37     }
38     cout<<ans<<endl;
39     return 0;
40 }
```

38. ①处应填 ()。

A. a.power < b.power

B. a.pos > b.pos

C. a.pos < b.pos

D. a.power > b.power

39. ②处应填 ()。

A. dp[1] = 0

B. dp[1] = inf

C. dp[1] = 1

D. dp[1] = -inf

40. ③处应填 ()。

- A. beacons[i].pos
- B. beacons[i].power
- C. beacons[i].pos + beacons[i].power
- D. beacons[i].pos - beacons[i].power

41. ④处应填 ()。

- A. lower_bound(beacons+1, beacons+n, find, cmp)
- B. upper_bound(beacons+1, beacons+n+1, find, cmp)
- C. lower_bound(beacons+1, beacons+n+1, find, cmp)
- D. upper_bound(beacons+1, beacons+n, find, cmp)

42. ⑤处应填 ()。

- A. n-dp[i]
- B. dp[i]-i
- C. dp[i]
- D. dp[i]+n-i