

# CSP初赛倒计时10天， 信竞生家长要关注什么？

**CSP-J初赛考点解读及技巧分析**



学而思



00

Lesson

# 序章 信竞简介

▶▶ NOIP

---

**N**ational **O**lympiad in **I**nformatics in **P**rovinces

全国青少年信息学奥林匹克联赛

## ▹▹ CSP-J/S

---

**主办方：**中国计算机学会（CCF）

**级别：**全国统一试卷，各省分开评奖

**支持语种：**C++

**组别：**J 入门级、S 提高级

## ▶▶ CSP-J/S

### 第一轮

时间	9:30 – 11:30	14:30 – 16:30
2022年9月18日 (周日)	CSP-J1 入门级	CSP-S1 提高级

### 第二轮

时间	8:30 – 12:00	14:30 – 18:30
2022年10月22日 (周六)	CSP-J1 入门级	CSP-S1 提高级

# C++零基础竞赛班入学测

## — 计算思维评估





- 1 CSP-J初赛模拟赛考情分析
- 2 CSP-J初赛单项选择题考点解读
- 3 CSP-J初赛阅读程序及完善程序题技巧分析
- 4 CSP-J复赛备考规划



# 01

Lesson

## CSP-J初赛模拟赛考情分析



# CSP-J 2022初赛模拟赛概况

参考人数：643人

单项选择题平均分：19.74分(满分30分)

阅读程序题平均分：13.35分(满分40分)

完善程序题平均分：11.03分(满分30分)

总平均分：44.12分(满分100分)

# CSP-J 2022初赛模拟赛每题平均分

## 单项选择题每题平均分统计表

题号	1	2	3	4	5
分值	2	2	2	2	2
考点	计算机的基本构成	计算机网络	进制运算	组合数学	二叉树
平均分	1.47	1.60	1.42	1.58	1.13
题号	6	7	8	9	10
分值	2	2	2	2	2
考点	队列	简单图	逻辑运算	二叉树	递推
平均分	1.80	1.16	1.32	1.01	1.39
题号	11	12	13	14	15
分值	2	2	2	2	2
考点	递归	进制转换	排序算法	C++语法	计算机常识
平均分	1.22	1.18	1.09	1.34	1.08

# CSP-J 2022初赛模拟赛每题平均分

阅读程序题每题平均分统计表

程序1：树结构(单词查找树)

题号	16	17	18	19	20	21
分值	1.5	1.5	1.5	1.5	3	3
平均分	0.66	0.98	0.75	0.52	1.21	0.91

程序2：二分(最长不下降子序列)

题号	22	23	24	25	26
分值	1.5	1.5	1.5	3	4
平均分	0.57	0.87	0.73	1.06	1.30

程序3：快速排序(分治)

题号	27	28	29	30	31	32
分值	1.5	1.5	1.5	4	4	4
平均分	0.58	1.15	0.60	0.64	0.63	0.35

# CSP-J 2022初赛模拟赛每题平均分

完善程序题每题平均分统计表

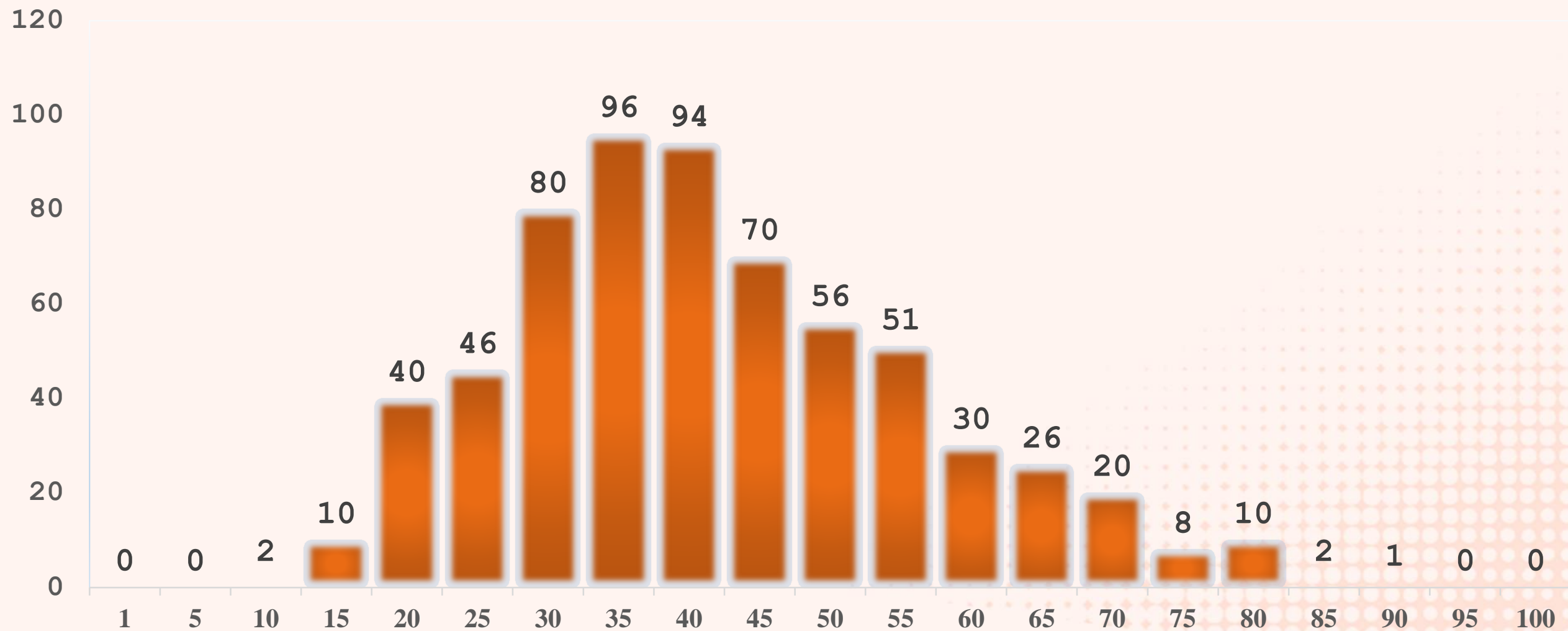
程序1：模拟

题号	33	34	35	36	37
分值	3	3	3	3	3
平均分	1.53	1.17	0.99	1.50	0.99

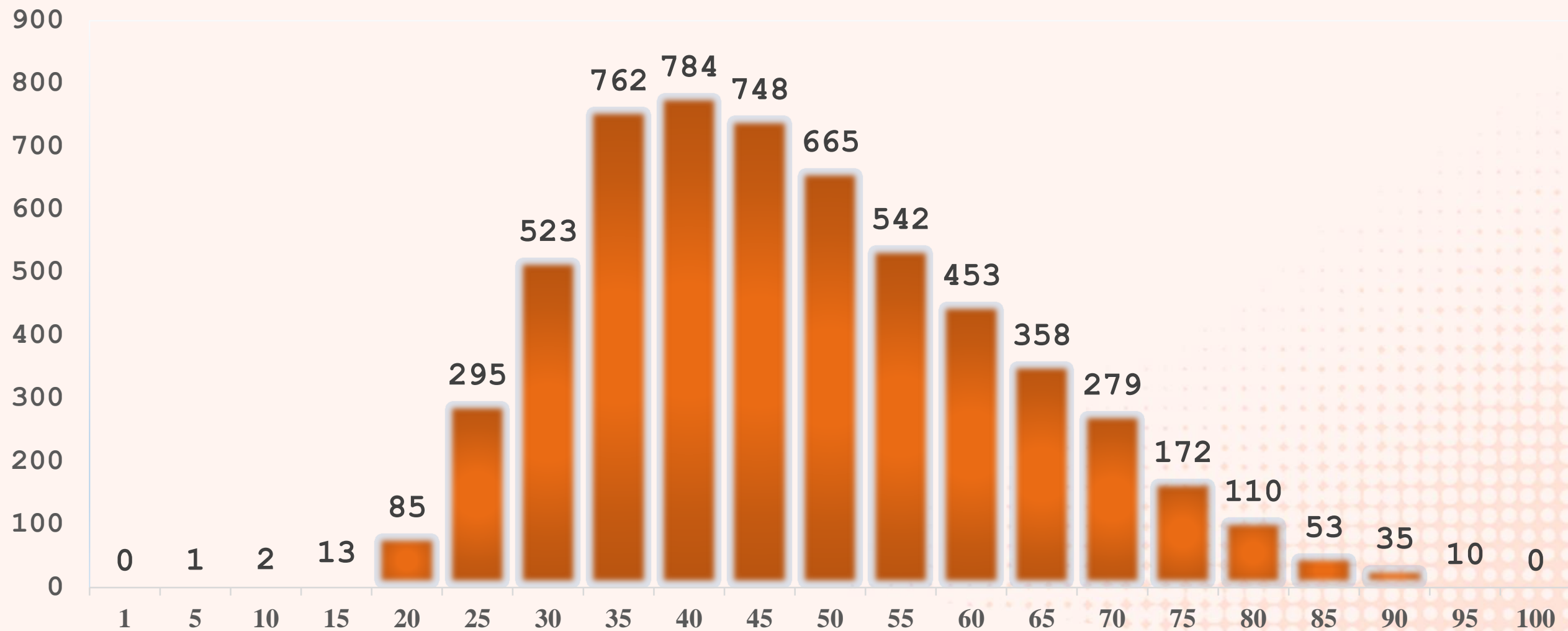
程序2：深搜(回溯)

题号	38	39	40	41	42
分值	3	3	3	3	3
平均分	0.89	1.31	0.99	1.12	0.98

# CSP-J 2022初赛模拟赛分数段人数

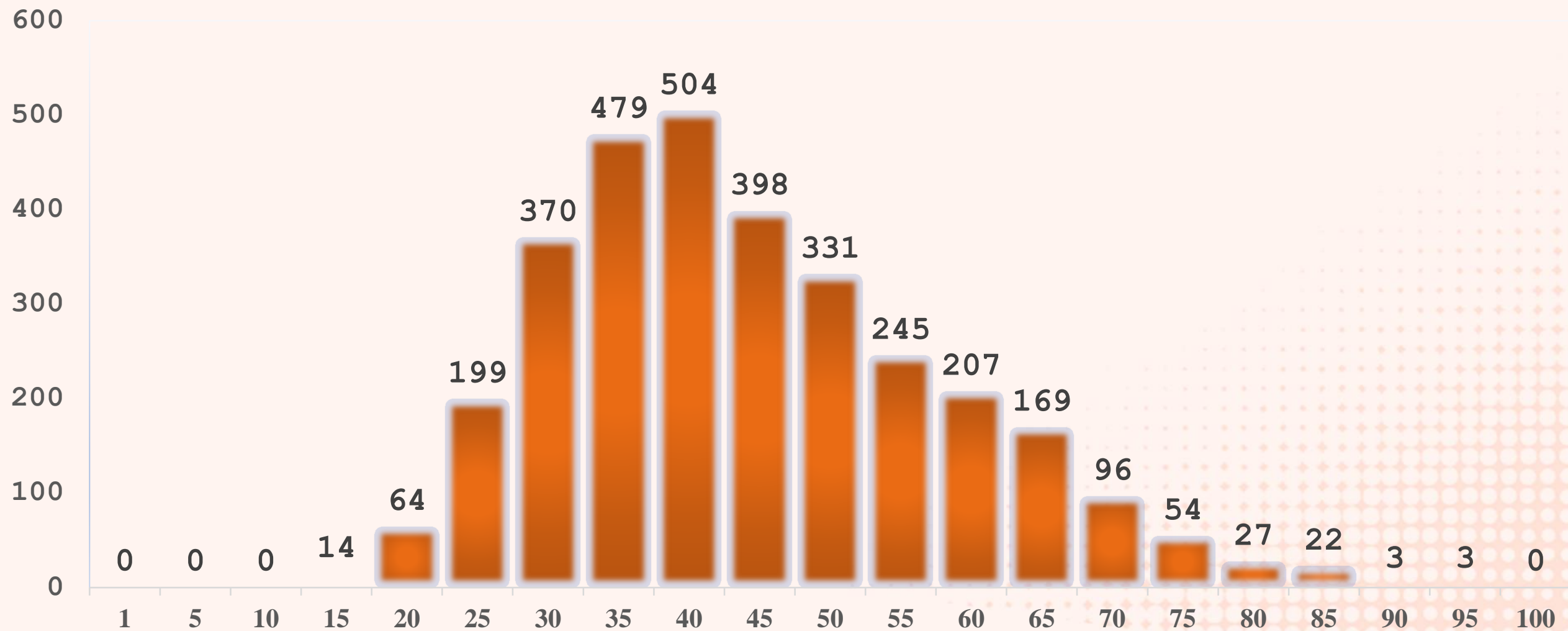


# CSP-J 2021浙江分数段人数





# CSP-J 2021北京分数段人数



# CSP-J 2022初赛模拟赛**分数线**

前5%分数线: **71.5**

前10%分数线: **65.5**

前15%分数线: **60**

前20%分数线: **56**

前25%分数线: **53.5**

前30%分数线: **50.5**

# CSP-J 2019-2021各省市分数

省市	2021	2020	2019	省市	2021	2020	2019	省市	2021	2020	2019
浙江	66	77.5	80	河南	35	12	10	四川	17	70.5	41
广东	59	68	72.5	安徽	34	33	43	云南	17	20	25
北京	53	57.5	54.8	吉林	33	36	35	天津	16	55	18.5
上海	53	61	60.5	澳门	31	45	46.5	山东	15	31.5	72
重庆	52	58.5	56.5	山西	30	40	30	甘肃	15	20	28
香港	46	44	45.5	贵州	30	30	7.5	海南	15	16	29.5
河北	40	49	35	湖南	29	44	45	福建	—	21.5	55.5
辽宁	40	48	47.5	广西	28	30	38	青海	—	—	—
新疆	39	16.5	45.5	陕西	28	25	16	西藏	—	—	—
江苏	38	77	80.5	湖北	26	30	20	台湾	—	—	—
江西	37	51	15	黑龙江	25	45.5	34.5				
内蒙古	35	40	40	宁夏	24	22	25.5				

# CSP-J 2022初赛模拟赛**报名**路径

电脑浏览器打开[www.xiaohoucode.com](http://www.xiaohoucode.com)

1.首页头图处点击进入注册报名

2.左侧边栏—编程活动—CSP-J 2022初赛模拟赛

注意：**“学而思信息学竞赛”** 微信公众号可查看解析

# CSP-J初赛真题**练习平台+视频讲解**

真题  
练习平台

学而思培优  
小猴编程官方网站  
([www.xiaohoucode.com](http://www.xiaohoucode.com))

点击左侧的  
竞赛题库

点击左上角的  
CSP初赛题库

免费视频讲解

• 学而思信息学竞赛微信公众号发布

# 02

Lesson

## CSP-J初赛单项选择题考点解读



## 2019~2021年CSP-J初赛真题单项选择题考点分类汇总

2019		2020		2021	
题号	考点	题号	考点	题号	考点
第01题	计算机基础-计算机网络	第01题	计算机基础-计算机的基本构成	第01题	计算机基础-程序设计语言
第02题	C++语法-逻辑运算	第02题	计算机基础-程序的编译和运行	第02题	计算机基础-计算机常识
第03题	C++语法-基本数据类型	第03题	C++语法-逻辑运算	第03题	计算机基础-计算机的基本构成
第04题	C++语法-循环结构	第04题	计算机基础-计算机的基本构成	第04题	算法-排序
第05题	算法-二分搜索	第05题	算法-排序	第05题	数据结构-栈
第06题	数据结构-链表	第06题	算法-递归	第06题	数据结构-简单树
第07题	数学-组合数学	第07题	数据结构-链表	第07题	数学-进制转换
第08题	数据结构-二叉树	第08题	数据结构-简单图	第08题	数据结构-二叉树
第09题	数学-初等数论	第09题	数学-进制转换	第09题	数据结构-二叉树
第10题	数学-初等数论	第10题	数学-组合数学	第10题	数学-组合数学
第11题	数学-数及其运算	第11题	数据结构-栈	第11题	数据结构-二叉树
第12题	数学-初等数论	第12题	数据结构-二叉树	第12题	数学-组合数学
第13题	数学-组合数学	第13题	数学-数及其运算	第13题	算法-递归
第14题	数据结构-二叉树	第14题	数学-组合数学	第14题	数据结构-简单图
第15题	计算机基础-计算机常识	第15题	数学-组合数学	第15题	算法-贪心

# 全国青少年信息学奥林匹克系列竞赛大纲(CSP-J入门级)

计算机基础与编程环境	C++程序设计	数据结构	算法	数学
计算机的基本构成(CPU、内存、I/O设备等)	程序基本概念	线性表	算法概念与描述	数及其运算
Windows、Linux等操作系统的基本概念及其常见操作	基本数据类型	简单树	入门算法	初中数学
计算机网络和Internet的基本概念	程序基本语句	完全二叉树的定义与基本性质	基础算法	初等数论
计算机附历史及其在现代社会中的常见应用	基本运算	简单图	数值处理算法	组合数学
NOI以及相关活动的历史	数学库常用函数		排序算法	
进制的基本概念与进制转换、字节与字	结构化程序设计		图论算法	
程序设计语言以及程序编译和运行的基本概念	数组		动态规划	
使用图形界面新建、复制、删除、移动文件或目录	字符串的处理			
使用Windows系统下的集成开发环境(例如Dev C++等)	函数与递归			
使用Linux系统下的集成开发环境(例如Code::Blocks等)	结构体类型			
g++、gcc等常见编译器的基本使用	指针类型			
	文件及基本读写			
	STL模板应用			



# 03

Lesson

## CSP-J初赛阅读程序及完善程序题技巧分析

## 阅读程序题

CSP初赛除了基础选择题，还有非常重要的两道大题：阅读程序和完善程序。阅读程序一共有3大题，共40分，是CSP-J/S考试中的一大拉分点。该题首先需要考生认真阅读程序，然后会采用判断题和选择题考查考生对程序的理解深度。

本题所考查的知识点非常多，主要包括计算机语言、多重循环的嵌套、数组的操作等。对于阅读程序题，考生一定要细心，注意循环的判断条件以及程序的输入输出细节，可以按照程序的运行顺序，从主函数还是阅读程序。

## 阅读程序题

阅读程序题通常考察考生以下几个方面的能力：

- 程序设计语言的掌握情况；
- 程序中基础算法的掌握情况；
- 数学计算的知识与运算能力；
- 细心、耐心的计算思维能力。



## 🕒 阅读程序题常见解题方法——模拟法

- 在程序阅读题中，如果难以理解题目所要完成的功能，最简单有效的方法就是**模拟法**。
- 所谓模拟法，就是利用人脑模拟程序的执行运算过程。
- 当程序涉及数组、循环、递归等更复杂的语句的时候，通常会有多个相似的变量发生变化，稍不留神就会导致整个程序发生错误，所以通常会**设计表格**来表示各种数据。





学而

例是

	x	y	dx	dy	cnt	n	m
	1	1	1	1	0	4	3
(1)	2	2	1	1	0		
	3	(3)	1	-1	1		
(2)	(4)	2	-1	-1	1		
	3	(1)	-1	1	1		
	2	2	-1	1	0		
	(1)	(3)	1	-1	2		

```
1 #include<iostream>
2 using namespace std;
3 int main()
4 {
5     int n, m;
6     cin>>n>>m;
7     int x=1, y=1, dx=1, dy=1;
8     int cnt=0;
9     while(cnt!=2) {
10         cnt=0;
11         x=x+dx;
12         y=y+dy;
13         if(x==1 || x==n) {
14             ++cnt;
15             dx=-dx;
16         }
17         if(y==1 || y==m) {
18             ++cnt;
19             dy=-dy;
20         }
21     }
22     cout<<x<<" "<<y<<endl;
23     return 0;
24 }
```

## 🕒 阅读程序题常见解题方法——**猜测验证**

模拟法虽然很有用，但如果考生对整个程序的功能不够理解，则会造成求解的速度很慢。

在详细地阅读程序之前，考生可以借助以前学习程序的经验以及程序中变量和函数一些常用的写法提示，大胆地猜测程序的功能，然后再进行验证。

# 🕒 阅读程序题常见解题方法——猜测验证

## (1) 变量的常用含义

序号	变量名	英文单词全拼/含义	序号	变量名	英文单词全拼/含义
1	ans	answer/计算答案	17	index	索引,多用于数组下标
2	ret	return/返回值	18	first	第一,多用于比较
3	res	result/计算结果	19	second	第二,多用于比较
4	flag	标识,多用于记录状态	20	last	最后,多用于比较
5	done	完成	21	begin	开始,多用于指针位置
6	error	错误,多用于记录错误信息	22	end	结束,多用于指针位置
7	found	找到,多用于 bool 变量	23	start	开始,多用于指针位置
8	success	成功	24	node	节点,多用于链表
9	ok	完成	25	op	操作符,多用于链表
10	num	number/数字	26	min	最小值,多用于比较
11	value	值	27	max	最大值,多用于比较
12	cnt	count/统计	28	avg	平均值,多用于计算
13	target	目标	29	total	总和,多用于统计
14	record	记录	30	preNode	上一个节点,用于链表
15	foo	确实存在东西的普遍替代语	31	curNode	当前节点,用于链表
16	tmp/temp	临时变量	32	nextNode	下一个节点,用于链表

## 🕒 阅读程序题常见解题方法——**猜测验证**

### (2) 算法的结构

很多常见的算法都有一些的特殊结构，了解并掌握这些结构不仅能在阅读程序上快速地判断出程序的功能，而且在以后编写程序时，也能通过算法结构快速地写出程序。

## 🕒 阅读程序题常见解题方法——猜测试验证

### (2) 算法的结构

#### 1、二分法

```
while(l <= r)
{
    mid = (l+r) / 2;
    ...
    if(...) l = mid + 1;
    else r = mid - 1;
}
```



## 🕒 阅读程序题常见解题方法——**猜测验证**

### (2) 算法的结构

#### 2、桶排序

```
for(int i = 1; i <= n; i++) b[a[i]]++;
```

```
for(int i = 1; i <= m; i++)
```

```
    for(int j = 1; j <= b[i]; j++)
```

```
        c[++cnt] = i;
```





## 🕒 阅读程序题常见解题方法——**猜测验证**

### (2) 算法的结构

#### 3、连续字符转换成整数

```
num=0;
```

```
cin>>c;
```

```
while(c>='0'&&c<='9') {
```

```
    num=num*10+c-'0' ;
```

```
    cin>>c;
```

```
}
```



## 🕒 阅读程序题常见解题方法—**猜测验证**

### (2) 算法的结构

#### 4、链表

`data[i]`

`last[next[i]] = last[i];`

`next[last[i]] = next[i];`



## 🕒 阅读程序题常见解题方法—**猜测验证**

---

### (2) 算法的结构

#### 5、分治法

`solve(l, mid, ...);`

`solve(mid+1, r, ...);`

## 🕒 阅读程序题常见解题方法—**猜测验证**

### (2) 算法的结构

#### 6、二叉树

left, right, father, depth

l, r, f

dfs(left, k-1, ...);

dfs(k+1, right, ...);



## 📖 例题2 阅读程序写结果

输入: abacaba

输出: 16

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4  string s;
5  long long magic(int l, int r)
6  {
7      long long ans = 0;
8      for(int i = l; i <= r; ++i)
9          ans = ans * 4 + s[i] - 'a' + 1;
10     return ans;
11 }
12
13 int main()
14 {
15     cin >> s;
16     int len = s.length();
17     int ans = 0;
18     for(int l1 = 0; l1 < len; ++l1)
19         for(int r1 = l1; r1 < len; ++r1)
20         {
21             bool bo = true;
22             for(int l2 = 0; l2 < len; ++l2)
23                 for(int r2 = l2; r2 < len; ++r2)
24                 {
25                     if (magic(l1, r1) == magic(l2, r2) && (l1 != l2 || r1 != r2))
26                         bo = false;
27                 }
28             if (bo) ans += 1;
29         }
30     cout << ans << endl;
31     return 0;
32 }
```

## 完善程序题

在CSP初赛中，完善程序一共有2大题，共30分。完善程序通常会涉及到具体问题的解决和编程，因此具有非常强的整体性。

另一方面，完善程序是分差最大的题型，对于编程基础薄弱的学生，可能面对不完整的程序找不到切入点，导致得分很低；而对于代码能力较强的学生，可能很熟悉相关的问题和代码，得分比较轻松。

对于完善程序题，具体的做题技巧作用是有限的，更需要考生平时多多练习和积累，提高算法的代码实现能力。



## 完善程序题

完善程序题通常考察考生以下几个方面的能力：

- 算法问题的分析能力；
- 经典代码的编写能力；
- 代码功能的阅读能力；
- 代码细节的实现能力。

## 🕒 完善程序题常见解题方法——模拟验证法

由于大部分程序都是对变量或者数组进行数据操作，因此每一条语句的效果都是可以预见的。

完善程序题通常是在四个选项中选出一个，选项之间区别很小的时候，使用具体的数据来模拟每个选项的效果就成为了准确度最高的一种方法。我们可以用相对简单的数据来验证选项的正确性。



例题3. (快速排序) 快速排序是一个常用且经典的排序算法，它基于分治法，基本思想如下：

- 1) 先从数组中取出一个数作为基准数。
- 2) 将比这个数小的数全放在数组的左边，大于等于这个数的数全放到数组的右边，生成左右两个区间。
- 3) 再对左右区间分别重复一、二步，直到各区间只有一个数。

现在给定一个由n个无序整数组成的数组s，使用快速排序将s数组从小到大排序，试补全程序。

```
1  #include<iostream>
2  using namespace std;
3  const int maxn = 100000;
4  int n, s[maxn];
5  void quick_sort(int s[], int l, int r)
6  {
7      if (①) return; //只有一个整数时排序结束
8      int i = l, j = r, x = s[l]; //选第一个数为基准数
9      while (i < j)
10     {
11         while(i < j && s[j] >= x)//从右向左找第一个小于x的数
12             j--;
13         if(i < j) {
14             s[i] = s[j];
15             i++;
16         }
17         while(i < j && ②) //从左向右找第一个大于等于x的数
18             i++;
19         if(i < j) {
20             ③;
21             j--;
22         }
23     }
24     s[i] = x;
25     quick_sort(s, l, i - 1);
26     quick_sort(s, ④);
27 }
28 int main()
29 {
30     cin >> n;
31     for(int i = 0; i < n; i++)
32         cin >> s[i];
33     quick_sort(s, ⑤);
34     for(int i = 0; i < n; i++)
35         cout << s[i] << ' ';
36     return 0;
37 }
```



### 例题3. (快速排序)

1)①处应填 ( )。

- A.  $l > r$                       B.  $l \geq r$   
C.  $l < 0 \parallel r \geq n$               D.  $l < r$

2)②处应填 ( )。

- A.  $s[j] < x$                       B.  $s[j] \geq x$   
C.  $s[i] < x$                       D.  $s[i] \geq x$

3)③处应填 ( )。

- A.  $s[i] = s[j]$                       B.  $s[j] = s[i]$   
C.  $s[i] = x$                       D.  $s[j] = x$

4)④处应填 ( )。

- A.  $0, n - 1$                       B.  $l, r$   
C.  $i, r$                       D.  $i + 1, r$

5)⑤处应填 ( )。

- A.  $0, n - 1$                       B.  $1, n$   
C.  $0, 0$                       D.  $n, n$

```
1  #include<iostream>
2  using namespace std;
3  const int maxn = 100000;
4  int n, s[maxn];
5  void quick_sort(int s[], int l, int r)
6  {
7      if (①) return; //只有一个整数时排序结束
8      int i = l, j = r, x = s[l]; //选第一个数为基准数
9      while (i < j)
10     {
11         while(i < j && s[j] >= x)//从右向左找第一个小于 x 的数
12             j--;
13         if(i < j) {
14             s[i] = s[j];
15             i++;
16         }
17         while(i < j && ②) //从左向右找第一个大于等于 x 的数
18             i++;
19         if(i < j) {
20             ③;
21             j--;
22         }
23     }
24     s[i] = x;
25     quick_sort(s, l, i - 1);
26     quick_sort(s, ④);
27 }
28 int main()
29 {
30     cin >> n;
31     for(int i = 0; i < n; i++)
32         cin >> s[i];
33     quick_sort(s, ⑤);
34     for(int i = 0; i < n; i++)
35         cout << s[i] << ' ';
36     return 0;
37 }
```



## 🕒 完善程序题常见解题方法——模仿相似代码

在我们编写程序的时候，常常会出现相似度很高的代码或者语句，它们通常是对不同的对象做相同或者相似的处理。这种现象常常出现在枚举、分治或者树结构相关的程序上。当我们需要补全语句的时候，参考与它相似的段落往往会给我们带来很多提示和启发。

## 例题4. (二叉查找)

二叉查找树具有如下性质：**每个节点的值都大于其左子树上所有节点的值、小于其右子树上所有节点的值。**

试判断一棵树是否为二叉查找树。

输入的第一行包含一个整数 $n$ ，表示这棵树有 $n$ 个顶点，编号分别为 $1, 2, \dots, n$ ，其中编号为1的为根结点。之后的第 $i$ 行有三个数 $value, lch, rch$ ，分别表示该节点关键字的值、左子节点的编号、右子节点的编号（ $-100,000 \leq value \leq 100,000, 1 \leq lch, rch \leq n$ ）；如果不存在左子节点或右子节点，则用0代替。输出1表示这棵树是二叉查找树，输出0则表示不是，试补全程序。

```
1  #include <iostream>
2  using namespace std;
3  const int SIZE = 100, INFINITE = 1000000;
4  struct node {
5      int lch; // Left child
6      int rch; // right child
7      int value;
8  };
9  node a[SIZE];
10 int is_bst(int root, int lower_bound, int upper_bound)
11 {
12     if (root == 0)
13         return 1;
14     int cur = ①;
15     if ((cur > lower_bound) && (cur < upper_bound) &&
16         (is_bst( ② , lower_bound, cur) == 1) &&
17         (is_bst(a[root].rch, ③ , ④ ) == 1))
18         return 1;
19     return 0;
20 }
21 int main()
22 {
23     int n;
24     cin >> n;
25     for (int i = 1; i <= n; i++)
26         cin >> a[i].value >> a[i].lch >> a[i].rch;
27     cout << is_bst( ⑤ , -INFINITE, INFINITE) << endl;
28     return 0;
29 }
```





#### 例题4. (二叉查找)

1) ①处应填 ( )。

- A. a[root].value    B. a[root].lch  
C. a[root].rch        D. root

2) ②处应填 ( )。

- A. a[root].value    B. a[root].lch  
C. a[root].rch        D. root

3) ③处应填 ( )。

- A. lower\_bound        B. upper\_bound  
C. cur                  D. root

4) ④处应填 ( )。

- A. lower\_bound        B. upper\_bound  
C. cur                  D. root

5) ⑤处应填 ( )。

- A. 0                    B. 1  
C. n                    D. n-1

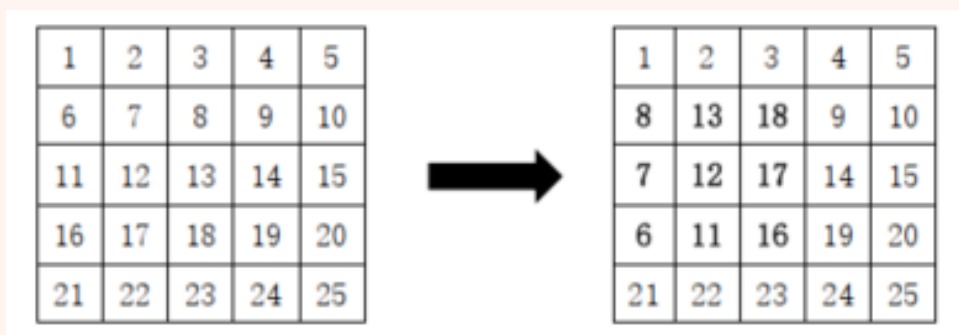
```
1  #include <iostream>
2  using namespace std;
3  const int SIZE = 100, INFINITE = 1000000;
4  struct node {
5      int lch; // Left child
6      int rch; // right child
7      int value;
8  };
9  node a[SIZE];
10 int is_bst(int root, int lower_bound, int upper_bound)
11 {
12     if (root == 0)
13         return 1;
14     int cur = ①;
15     if ((cur > lower_bound) && (cur < upper_bound) &&
16         (is_bst( ② , lower_bound, cur) == 1) &&
17         (is_bst(a[root].rch, ③ , ④ ) == 1))
18         return 1;
19     return 0;
20 }
21 int main()
22 {
23     int n;
24     cin >> n;
25     for (int i = 1; i <= n; i++)
26         cin >> a[i].value >> a[i].lch >> a[i].rch;
27     cout << is_bst( ⑤ , -INFINITE, INFINITE) << endl;
28     return 0;
29 }
```

## 🕒 完善程序题常见解题方法—**相关变量**

对于大部分程序员来说，一个变量的使用场景常常是相同的，在多变量之间进行分辨的时候，如果能明确每个变量的意义，就能更快地排除错误选项，完成题目。

### 例题5. (矩阵变换)

给定一个 $n$ 行 $n$ 列的矩阵，矩阵初始由1到 $n^2$ 的正整数按照从左往右，从上到下的顺序依次填入。接下来对这个矩阵进行 $m$ 次变换，每次变换需要输入四个整数 $x, y, r, z$ ，表示把以第 $x$ 行第 $y$ 列为中心的边长为 $2r+1$ 的方阵按照顺时针或者逆时针方向旋转 $90^\circ$ ，其中 $z=0$ 表示顺时针， $z=1$ 表示逆时针。例如当 $n=5, m=1, x=3, y=2, r=1, z=1$ 时，矩阵变换如下图所示：



编写程序求 $m$ 次变换后得到的矩阵，试补全程序。

```

1  #include<stdio>
2  using namespace std;
3  const int maxn = 505;
4  int n, m, d[maxn][maxn], f[maxn][maxn];
5  int main()
6  {
7      scanf("%d%d", &n, &m);
8      int tot = 0, x, y, r, z;
9      for (int i = 1; i <= n; i++)
10         for (int j = 1; j <= n; j++)
11             d[i][j] = ++tot;
12     for(int k = 1; k <= m; k++)
13     {
14         scanf("%d%d%d%d", &x, &y, &r, &z);
15         if(z == 0) {
16             for(int i = x - r; i <= x + r; i++)
17                 for(int j = y - r; j <= y + r; j++)
18                     f[x - y + j][y + x - i] = d[i][j];
19         }
20         else {
21             for(int i = x - r; i <= x + r; i++)
22                 for(int j = y - r; j <= y + r; j++)
23                     f[0][0] = d[i][j];
24         }
25         for(int i = x - r; i <= x + r; i++)
26             for(int j = 0; j++)
27                 0;
28     }
29     for(int i = 1; i <= n; i++)
30     {
31         for(int j = 1; j <= n; j++)
32             printf("%d ", d[i][j]);
33         printf("\n");
34     }
35     return 0;
37 }
```



### 例题5. (矩阵变换)

1)①处应填 ( )。

A. ++tot

B. i

C. tot++

D. j

2)②处应填 ( )。

A.  $x - y + j$

B.  $x + y - j$

C.  $y - x + i$

D.  $y + x - i$

3)③处应填 ( )。

A.  $x - y + j$

B.  $x + y - j$

C.  $y - x + i$

D.  $y + x - i$

4)④处应填 ( )。

A.  $y - r; j \leq y + r$

B.  $1; j \leq y$

C.  $y; j \leq n$

D.  $1; j \leq n$

5)⑤处应填 ( )。

A.  $d[i][j] = f[j][i]$

B.  $d[i][j] = f[i][j]$

C.  $f[i][j] = d[j][i]$

D.  $f[i][j] = d[i][j]$

```
1  #include<stdio>
2  using namespace std;
3  const int maxn = 505;
4  int n, m, d[maxn][maxn], f[maxn][maxn];
5  int main()
6  {
7      scanf("%d%d", &n, &m);
8      int tot = 0, x, y, r, z;
9      for (int i = 1; i <= n; i++)
10         for (int j = 1; j <= n; j++)
11             d[i][j] = 0;
12     for(int k = 1; k <= m; k++)
13     {
14         scanf("%d%d%d%d", &x, &y, &r, &z);
15         if(z == 0) {
16             for(int i = x - r; i <= x + r; i++)
17                 for(int j = y - r; j <= y + r; j++)
18                     f[x - y + j][y + x - i] = d[i][j];
19         }
20         else {
21             for(int i = x - r; i <= x + r; i++)
22                 for(int j = y - r; j <= y + r; j++)
23                     f[0][0] = d[i][j];
24         }
25         for(int i = x - r; i <= x + r; i++)
26             for(int j = 0; j++)
27                 0;
28     }
29     for(int i = 1; i <= n; i++)
30     {
31         for(int j = 1; j <= n; j++)
32             printf("%d ", d[i][j]);
33         printf("\n");
34     }
35     return 0;
37 }
```





## 🕒 完善程序题常见解题方法——经典算法实现

完善程序题通常会先给一个应用编程题，然后需要编写程序解决。大部分题目都会涉及到一个或多个具体的算法，而很多算法是有明显的代码编写特色的。因此，利用算法本身约定俗成的写法，可以帮助我们完成题目。

## 例题6. (最大公约数之和) (2018年)

下列程序想要求解整数 $n$ 的所有约数两两之间最大公约数的和对10007求余后的值，试补全程序。（第一空2分，其余3分）

举例来说，4的所有约数是1,2,4。1和2的最大公约数为1；2和4的最大公约数为2；1和4的最大公约数为1。于是答案为 $1 + 2 + 1 = 4$ 。要求getDivisor函数的复杂度为 $O(\sqrt{n})$ ，gcd函数的复杂度为 $O(\log \max(a,b))$ 。

```
#include <iostream>
using namespace std;
const int N = 110000, P = 10007;
int n;
int a[N], len;
int ans;
void getDivisor() {
    len = 0;
    for (int i = 1; (1) <= n; ++i)
        if (n % i == 0) {
            a[++len] = i;
            if ((2) != i) a[++len] = n / i;
        }
}

int gcd(int a, int b) {
    if (b == 0) {
        (3);
    }
    return gcd(b, (4));
}

int main() {
    cin >> n;
    getDivisor();
    ans = 0;
    for (int i = 1; i <= len; ++i) {
        for (int j = i + 1; j <= len; ++j) {
            ans = ((5)) % P;
        }
    }
    cout << ans << endl;
    return 0;
}
```





## 例题7. (郊游活动) (NOIP2016)

有 $n$ 名同学参加学校组织的郊游活动，已知学校给这 $n$ 名同学的郊游总经费为 $A$ 元，与此同时，第 $i$ 位同学自己携带了 $M_i$ 元。为了方便郊游，活动地点提供 $B$  ( $\geq n$ ) 辆自行车供人租用，租用第 $j$ 辆自行车的价格为 $C_j$ 元，每位同学可以使用自己携带的钱或者学校的郊游经费，为了方便账务管理，每位同学只能为自己租用自行车，且不会借钱给他人，他们想知道最多有多少位同学能够租用到自行车。(第四、五空2.5分，其余3分。)

本题采用二分法。对于区间 $[1, r]$ ，我们取中间点 $mid$ 并判断租用到自行车的人数能否达到 $mid$ 。判断的过程是利用贪心算法实现的。

```
#include<iostream>
using namespace std;
#define MAXN 1000000
int n, B, A, M[MAXN], C[MAXN], l, r, ans, mid;
bool check(int nn) {
    int count = 0, i, j;
    i = (1);
    j = 1;
    while(i <= n) {
        if((2))
            count += C[j] - M[i];
        i++;
        j++;
    }
    return (3);
}

void sort(int a[], int l, int r) {
    int i = 1, j = r, x = a[(l + r) / 2], y;
    while (i <= j) {
        while(a[i] < x) i++;
        while(a[j] > x) j--;
        if(i <= j){
            y=a[i]; a[i]=a[j]; a[j]=y;
            i++; j--;
        }
    }
    if(i < r) sort(a, i, r);
    if(l < j) sort(a, l, j);
}

int main() {
    int i;
    cin >> n >> B >> A;
    for(i=1; i<=n; i++)
        cin >> M[i];
    for(i=1; i<=B; i++)
        cin >> C[i];

    sort(M, 1, n);
    sort(C, 1, B);
    l = 0;
    r = n;
    while(l <= r) {
        mid = (l+r) / 2;
        if((4)){
            ans = mid;
            l = mid + 1;
        }else
            r = (5);
    }
    cout<<ans<<endl;
    return 0;
}
```

## 🕒 完善程序题常见解题方法——**参考文字提示**

由于完善程序题的整体难度很高，阅读陌生代码对考生综合素质要求很高，而且很多题目写法不止一种。出题人为了保证考生能顺利理解代码，往往会在说明部分解释算法，或者在代码关键位置添加注释。而这些关键的文字信息，可以成为我们做出题目的重要助力。



### 例题8. (序列重排)

全局数组变量a定义如下:

```
const int SIZE=100;
```

```
int a[SIZE], n;
```

它记录着一个长度为n的序列a[1], a[2], ..., a[n]。

现在需要一个函数, 以整数p ( $1 \leq p \leq n$ ) 为参数, 实现如下功能: 将序列a的前p个数与后n-p个数对调, 且不改变这p个数 (或n-p个数) 之间的相对位置。例如, 长度为5的序列1, 2, 3, 4, 5, 当p=2时重排结果为3, 4, 5, 1, 2。

有一种朴素的算法可以实现这一需求, 其时间复杂度为 $O(n)$ 、空间复杂度为 $O(n)$ :

有一种朴素的算法可以实现这一需求, 其时间复杂度为 $O(n)$ 、空间复杂度为 $O(n)$ :

```
void swap1(int p) {  
    int i, j, b[SIZE];  
    for(i=1; i<=p; i++)  
        b[①] = a[i];    // (3分)  
    for(i=p+1; i<=n; i++)  
        b[i-p] = ②;    // (3分)  
    for(i=1; i<=③; i++)    // (2分)  
        a[i] = b[i];  
}
```

## 例题8. (序列重排)

全局数组变量a定义如下:

```
const int SIZE=100;
```

```
int a[SIZE], n;
```

它记录着一个长度为n的序列a[1], a[2], ... , a[n]。

现在需要一个函数, 以整数p ( $1 \leq p \leq n$ ) 为参数, 实现如下功能: 将序列a的前p个数与后n-p个数对调, 且不改变这p个数 (或n-p个数) 之间的相对位置。例如, 长度为5的序列1, 2, 3, 4, 5, 当p=2时重排结果为3, 4, 5, 1, 2。

有一种朴素的算法可以实现这一需求, 其时间复杂度为 $O(n)$ 、空间复杂度为 $O(n)$ :

我们也可以用时间换空间, 使用时间复杂度为 $O(n^2)$ 、空间复杂度为 $O(1)$ 的算法:

```
void swap2(int p) {  
    int i, j, temp;  
    for(i=p+1; i<=n; i++) {  
        temp = a[i];  
        for(j=i; j >= ④; j--) // (3分)  
            a[j] = a[j-1];  
        ⑤ = temp; // (3分)  
    }  
}
```





## 🕒 完善程序题常见解题方法——特殊数据代入

有时候我们观察选项，会发现一些选项相似度过高，如果选择特殊数据，很多选项都会得到一样的结果，那么就能同时排除多个错误选项。对于两个高度相似的选项，也可以设计一个特殊数据，使两者结果不同，然后通过模拟找出正确的结果，就可以锁定正确答案了。



## 例题9. 【CSP-J2019】

1) ①处应填 ( )

- A.  $n \% 2$       B. 0      C. t      D. 1

2) ②处应填 ( )

- A.  $x - \text{step}, y - \text{step}$       B.  $x, y - \text{step}$   
C.  $x - \text{step}, y$       D.  $x, y$

3) ③处应填 ( )

- A.  $x - \text{step}, y - \text{step}$       B.  $x + \text{step}, y + \text{step}$   
C.  $x - \text{step}, y$       D.  $x, y - \text{step}$

4) ④处应填 ( )

- A.  $n - 1, n \% 2$       B. n, 0  
C. n,  $n \% 2$       D.  $n - 1, 0$

5) ⑤处应填 ( )

- A.  $1 \ll (n + 1)$       B.  $1 \ll n$   
C.  $n + 1$       D.  $1 \ll (n - 1)$

```
1  #include <stdio>
2  using namespace std;
3  int n;
4  const int max_size = 1 << 10;
5
6  int res[max_size][max_size];
7
8  void recursive(int x, int y, int n, int t) {
9      if (n == 0) {
10         res[x][y] = ①;
11         return;
12     }
13     int step = 1 << (n - 1);
14     recursive(②, n - 1, t);
15     recursive(x, y + step, n - 1, t);
16     recursive(x + step, y, n - 1, t);
17     recursive(③, n - 1, !t);
18 }
19
20 int main() {
21     scanf("%d", &n);
22     recursive(0, 0, ④);
23     int size = ⑤;
24     for (int i = 0; i < size; ++i) {
25         for (int j = 0; j < size; ++j)
26             printf("%d", res[i][j]);
27         puts("");
28     }
29     return 0;
30 }
```





## 例题9. 【CSP-J2019】

(矩阵变幻) 有一个奇幻的矩阵, 在不停的变幻, 其变幻方式为: 数字0变成矩阵 $\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ , 数字1变成矩阵 $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$ 。最初该矩阵只有一个元素0, 变幻  $n$  次后, 矩阵会变成什么样?

例如, 矩阵最初为: [0]; 矩阵变幻 1 次后:  $\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ ; 矩阵变幻 2 次后:

$$\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}。$$

输入一行一个不超过 10 的正整数  $n$ 。输出变幻  $n$  次后的矩阵。

试补全程序。

提示:

“<<”表示二进制左移运算符, 例如 $(11)_2 << 2 = (1100)_2$ ;

而“^”表示二进制异或运算符, 它将两个参与运算的数中的每个对应的二进制位一一进行比较, 若两个二进制位相同, 则运算结果的对应二进制位为 0, 反之为 1。

```
1  #include <stdio>
2  using namespace std;
3  int n;
4  const int max_size = 1 << 10;
5
6  int res[max_size][max_size];
7
8  void recursive(int x, int y, int n, int t) {
9      if (n == 0) {
10         res[x][y] = ①;
11         return;
12     }
13     int step = 1 << (n - 1);
14     recursive(②, n - 1, t);
15     recursive(x, y + step, n - 1, t);
16     recursive(x + step, y, n - 1, t);
17     recursive(③, n - 1, !t);
18 }
19
20 int main() {
21     scanf("%d", &n);
22     recursive(0, 0, ④);
23     int size = ⑤;
24     for (int i = 0; i < size; ++i) {
25         for (int j = 0; j < size; ++j)
26             printf("%d", res[i][j]);
27         puts("");
28     }
29     return 0;
30 }
```



# 04

Lesson

## CSP-J复赛备考规划

## ▶▶ CSP-J2 各省市一等奖分数

省市	2021	2020	2019	2018	2017	2016	省市	2021	2020	2019	2018	2017	2016
浙江	287	255	265	230	280	300	湖南	210	175	210	190	210	200
广东	277	230	250	215	255	270	山东	190	165	185	175	205	195
江苏	245	205	240	210	245	265	香港	221	175	175	210	225	300
重庆	248	205	230	215	225	270	江西	166	155	175	170	205	200
上海	241	195	225	210	215	235	海南	152	145	160	140	205	160
四川	225	210	215	195	210	215	湖北	160	150	155	180	205	160
北京	236	195	210	195	210	245	天津	145	190	150	185	205	210
福建	/	185	210	195	235	255	辽宁	150	145	150	170	210	200
安徽	215	190	210	190	210	240	山西	156	130	150	170	205	280

## ▶▶ CSP-J2 各省市一等奖分数

省市	2021	2020	2019	2018	2017	2016	省市	2021	2020	2019	2018	2017	2016
河北	188	160	150	165	200	190	甘肃	/	150	120	130	125	0
吉林	152	150	145	180	205	180	内蒙古	/	115	110	/	/	/
新疆	152	/	145	170	210	200	陕西	152	110	110	/	/	/
广西	142	120	140	170	205	175	云南	100	100	110	100	/	210
黑龙江	165	135	140	165	205	200	青海	/	/	/	/	/	/
河南	154	145	135	160	200	190	台湾	/	/	/	/	/	/
澳门	/	100	130	140	130	200	西藏	/	/	/	/	/	/
贵州	/	110	130	130	150	150							
宁夏	/	120	130	100	200	100							

## ▹▹ CSP-J2(NOIP普及组复赛)各省市二、三等奖基准分数线

省市	二等奖	三等奖
2021年CSP-J2	100	70
2020年CSP-J2	100	60
2019年CSP-J2	100	70
2018年NOIP普及组复赛	100	70
2017年NOIP普及组复赛	120	80
2016年NOIP普及组复赛	100	80



## ▹▹ CSP-J2 复赛 考什么?

---

四道题, 100分/题, 共400分

考试时间: 3.5小时

考试策略:

T1	——	争取满分
T2	——	争取满分
T3	——	尽可能多的分数
T4	——	战略性放弃



## » CSP-J2 复赛 考什么?

### T1 考点: Z1 阶段内容, 简单语法

第一题	2021	2020	2019	2018	2017
试题名	分糖果	优秀的拆分	数字游戏	标题统计	成绩
主要考点	数学	枚举	字符串	字符串	算术运算
第一题	2016	2015	2014	2013	2012
试题名	买铅笔	金币	珠心算测试	计数问题	质因数分解
主要考点	算术运算	循环	枚举法	枚举法	枚举法

## » CSP-J2 复赛 考什么?

### T2 考点: Z2 阶段内容, 基础算法

第二题	2021	2020	2019	2018	2017
试题名	插入排序	直播获奖	公交换乘	龙虎斗	图书管理员
主要考点	排序	排序	模拟	枚举法	枚举法
第二题	2016	2015	2014	2013	2012
试题名	回文日期	扫雷游戏	比例简化	表达式求值	寻宝
主要考点	枚举法	二维数组	枚举法、互质	字符串、栈	模拟

## ▶▶ CSP-J复赛备考-**冲刺班推荐**-中秋集训班-(CSP-J代码能力集训)

### 01 适合对象

准备参加2022年CSP入门级复赛，目标在做出复赛第二题的学员

### 02 培养目标

针对CSP-J复赛第2题的常见题型进行刷题

### 03 课程时间

9月10日-9月11日上午08:30-11:00

### 04 课程价格

在线大班：360元

### 05 报名方式

微信扫下面的小程序码报名即可





## ▶▶ CSP-J复赛备考-**冲刺班推荐**-CSP-入门级复赛T2专题突破

### 01 适合对象

学完了C++语言基础知识，准备参加2022年CSP-J复赛并冲刺前两道题的同学。

### 02 培养目标

讲解CSP-J复赛**第2题的常见题型和分析方法**，目标是孩子能够通过本课程的学习熟悉CSP-J复赛第2题常见题型及应对方案。

### 03 课程时间

10月2日上午08:30-12:00,10月2日下午13:30-17:00

### 04 课程价格

在线大班：480元

### 05 报名方式

微信扫下面的小程序码报名即可





## ▹▹ CSP-J2 复赛 考什么?

### T3、T4 考点: Z3 及以上内容

第三题	2021	2020	2019	2018	2017
试题名	网络连接	表达式	纪念品	摆渡车	棋盘
主要考点	字符串、map	字符串、递归	动态规划	动态规划、优化	记忆化搜索
第三题	2016	2015	2014	2013	2012
试题名	海港	求和	螺旋矩阵	小朋友的数字	摆花
主要考点	队列	枚举法、递推	模拟法、优化	动态规划	递推



## » CSP-J2 复赛 考什么?

### T3、T4 考点: Z3 及以上内容

第四题	2021	2020	2019	2018	2017
试题名	小熊的果篮	方格取数	加工零件	对称二叉树	跳房子
主要考点	链表、模拟	动态规划	最短路	树、递归、哈希	二分法、动态规划、优化
第四题	2016	2015	2014	2013	2012
试题名	魔法阵	推销员	子矩阵	车站分级	文化之旅
主要考点	枚举法、优化	贪心法、优先队列	搜索、动态规划、优化	贪心法、拓扑排序	最短路

## ▶▶ CSP-J复赛备考-**冲刺班推荐**-CSP-入门级复赛T3、T4 部分分专项练习

### 01 适合对象

学完C++语言基础和搜索、动规等基本算法，准备参加2022年CSP-J复赛并冲刺后两道题拿部分分的同学。

### 02 培养目标

讲解CSP-J复赛第3、4两题拿到部分分数的方法和技巧，目标是孩子能够通过本课程的学习熟悉CSP-J复赛第3、4题常见题型及部分分拿分技巧。

### 03 课程时间

10月3日上午08:30-12:00,10月3日下午13:30-17:00

### 04 课程价格

在线大班：480元

### 05 报名方式

微信扫下面的小程序码报名即可





## ▶▶ CSP-J复赛备考-**冲刺班推荐**-CSP-入门级复赛T3、T4 AC专项练习

### 01 适合对象

学完C++语言基础和搜索、动规等基本算法，准备参加2022年CSP-J复赛并冲刺后两道题拿高分的同学。

### 02 培养目标

讲解CSP入门组复赛**第3、4两题的满分解法**，目标是孩子能够通过本课程的学习熟悉CSP-J复赛第3、4题常见题型及满分做法。

### 03 课程时间

10月4日上午08:30-12:00,10月4日下午13:30-17:00

### 04 课程价格

在线大班：480元

### 05 报名方式

微信扫下面的小程序码报名即可



# ▶▶ CSP-J复赛备考-**冲刺班推荐**-2022年CSP-J复赛全真模拟考与讲评

## 01 适合对象

通过准备2022年CSP-J初赛，准备参加复赛的学员

## 02 培养目标

CSP-J复赛全真模拟考与讲评。

## 03 课程时间

第一次模考与讲评：10月5日上午08:30-12:00,10月5日下午13:30-17:00

第二次模考与讲评：10月6日上午08:30-12:00,10月6日下午13:30-17:00

第三次模考与讲评：10月7日上午08:30-12:00,10月7日下午13:30-17:00

**注意：**三次模考试题完全不同，同学们可根据自己的时间和备考需求选择一个或多个模考与讲评班进行报名。

## 04 课程价格

第一次模考与讲评：在线大班：480元

第二次模考与讲评：在线大班：480元

第三次模考与讲评：在线大班：480元

## 05 报名方式

微信扫下面的小程序码报名即可





详情咨询请扫码加微信入群





THANKS



学而思