

2020 CCF 非专业级别软件能力认证第一轮

(CSP-S) 提高级 C++ 语言试题 海亮内部模拟卷

注意事项:

1. 本试卷为海亮内部作为交流的模拟试卷, 本试卷仅代表个人观点。
2. 此前由于工作的疏忽忘记注明试卷的性质, 含有多处错误, 请以此版本为准。

一、选择题: 每题 2 分, 共 15 题, 30 分. 在每小题给出的四个选项中, 只有一项是符合题目要求的.

1. 下列关于 NOIP 的说法, 错误的是: ▲

(A).NOIP 中文名称为全国青少年信息学奥林匹克联赛, 将于今年恢复举行。

(B).参加 NOIP 是参加 NOI 的必要条件, 不参加 NOIP 将不具有参加 NOI 的资格。

(C).NOIP 竞赛全国前五名将获得进入国家集训队的资格。

(D).在 NOIP 复赛中, NOI 各省组织单位必须严格遵循 CCF《关于 NOIP 数据提交格式的说明》的规范在竞赛结束后规定时间内向 CCF 提交本赛区所有参赛选手的程序。

2. 二进制数 001001 与 100101 进行按位异或的结果为: ▲

(A).101000 (B).100100 (C).101101 (D).101100

3. 在 8 位二进制补码中, 10101011 表示的数是十进制下的 ▲ 。

(A).43 (B).-43 (C).-85 (D).-84

4. 平衡树是计算机科学中的一类数据结构, 为改进的二叉查找树。一般的二叉查找树的查询复杂度取决于目标结点到树根的距离 (即深度), 因此当结点的深度普遍较大时, 查询的均摊复杂度会上升。为了实现更高效的查询, 产生了平衡树。下列数据结构中, 不属于平衡树的为: ▲

(A).线段树 (B).Splay 树 (C).替罪羊树 (D).红黑树

5. 组合数 $\binom{n}{k}$ 为从 n 件有标号物品中选出 k 件物品的方案数, 例如 $\binom{3}{2} = 3$ 。已知 $n, k \in \mathbb{N}$, 下列说法错误的为: ▲

(A). $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$ 。

(B). $\binom{2n}{k} (0 \leq k \leq 2n)$ 在 $k = n$ 时取得最大值。

(C).卡特兰数 $C_n = \binom{2n}{n}/n$

(D).包含 n 个 0 和 k 个 1, 且没有两个 1 相邻的字符串的数量为 $\binom{n+1}{k}$ 。

6. 下列有关 CPU 的说法, 正确的有 ▲

(A).CPU 的用途是将计算机系统所需要的显示信息进行转换驱动显示器。

(B).CPU 的性能和速度取决于时钟频率 (一般以赫兹或千兆赫兹计算, 即 hz 与 Ghz) 和每周期可处理的指令 (IPC), 两者合并起来就是每秒可处理的指令 (IPS)。

(C).AMD 是世界上最大的半导体公司, 也是首家推出 x86 架构处理器的公司。

(D).目前的 CPU 一般都带有 3D 画面运算和图形加速功能, 所以也叫做“图形加速器”或“3D 加速器”。

7. 下列算法中, 没有用到贪心思想的算法为 ▲

(A).计算无向图最小生成树的 Kruskal 算法。

(B).计算无向图点双连通分量的 Tarjan 算法。

(C).计算无向图单源最短路路径的 Dijkstra 算法。

(D).以上算法均使用了贪心的思想。

8. 在图 $G = (V, E)$ 上使用 Bellman-Ford 算法计算单源最短路径, 最坏时间复杂度为 ▲

(A). $\Theta(|V||E|)$ (B). $\Theta(|V| \log |V|)$ (C). $\Theta(|E| \log |E|)$ (D). $\Theta(|E|)$

9. 若要使用 g++ 编译器, 开启 -Ofast 优化, 且使用 C++ 11 标准, 将源文件 prog.cpp 编译为可执行程序 exec, 且保留调试信息, 则需要使用的编译命令为 ▲

(A).g++ prog.cpp -Ofast exec -std=c++11 -debug

(B).g++ prog.cpp -Ofast exec -std=c++11 -g

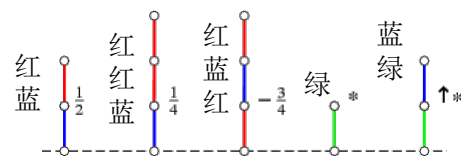
(C).g++ prog.cpp -o exec -Ofast -std=c++11 -debug

(D).g++ prog.cpp -o exec -Ofast -std=c++11 -g

10. 已知袋子 α 中装有 4 张 5 元纸币和 3 张 1 元纸币, 袋子 β 内装有 2 张 10 元纸币与 3 张 1 元纸币, 袋子 γ 中装有 3 张 20 元纸币与 3 张 50 元纸币。现在从每个袋子中随机选出 2 张纸币丢弃, 记第 i 个袋子此时剩下的纸币面值之和为 v_i , 则 $v_\alpha < v_\beta < v_\gamma$ 的概率为 ▲

(A). $\frac{8}{35}$ (B). $\frac{9}{35}$ (C). $\frac{11}{35}$ (D). $\frac{12}{35}$

11. Hackenbush 是一款老少皆宜的双人博弈游戏。该游戏双方分别被称为红方与蓝方。游戏的局面基于一些顶点和边, 其中边分为红色、蓝色与绿色。一些顶点长在大地上 (用虚线表示), 而另一些顶点将通过边直接与间接地与大地相连。每一局将从事先约定好的一方开始, 每一方轮流选择属于自己颜色的边, 然后将该边删除。特别地, 对于绿色的边, 红蓝双方都可以进行删除操作。如果删除后, 某些顶点或边不再与大地联通, 则这些顶点或边将自动被删除。如果轮到某一方时, 属于他的颜色的所有的边都被删除了, 那么他就输了整场游戏。



游戏的局面可以分为四种, 分别为先手必胜局面、后手必胜局面、红方必胜局面、蓝方必胜局面。例如, 图中第一、二个局面为蓝方必胜局面 (无论蓝方先手后手, 它只需要删除唯一的蓝色边, 红方就无法操作了); 第三个局面为红方必胜局面; 第四、五个局面为先手必胜局面。下列说法正确的为 ▲

(A).若 Hackenbush 中不存在绿色边, 则一定不会出现先手必胜局面。

(B).Hackenbush 中不存在后手必胜局面。

(C).即使不存在红色边与蓝色边, Hackenbush 问题仍是 NP-Hard 问题。

(D).若不存在绿色边, 则 Hackenbush 问题是 P 类问题。

12. 记将 n 个有标号球，放到若干个无标号集合，每个集合可空的方案数为 f_n 。 $\{n\}_k$ 为第二类斯特林数，其表示将 n 个有标号球放到 k 个非空的无标号集合内的方案数。则下列说法：

- (a) $f_n = \sum_{k=0}^n \{n\}_k$
 (b) $f_n = \sum_{k=0}^{n-1} \binom{n-1}{k} f_k$
 (c) $f_n = \left\lfloor \frac{x^n}{n!} \right\rfloor e^{e^x - 1}$

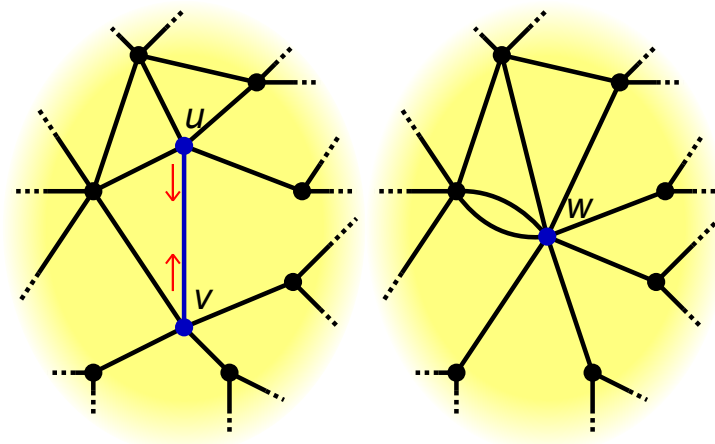
正确的为： ▲

- (A).(a), (b) (B).(a), (c) (C).(c) (D).(a), (b), (c)

13. 已知参数 k ，对于递归式 $T(n) = k\sqrt{n}T(\sqrt{n}) + n$ 的说法，正确的是 ▲

- (A).当 $k = 1$ 时， $T(n) = \Theta(n \log n)$ (B).当 $k = 1$ 时， $T(n) = \Theta(n \log^2 n)$
 (C).当 $k = 4$ 时， $T(n) = \Theta(n \log n)$ (D).当 $k = 4$ 时， $T(n) = \Theta(n \log^2 n)$

14. 对于图 $G = (V, E)$ 中的边 $e \in E$ ，我们记 $G - e$ 表示将边 e 删除后得到的新图 G' ， G/e 表示将 e 删除，并将 e 的两个端点合并为一个点后得到的新图（如图所示， $G/(u, v)$ 即为 u, v 合并为 w 后得到的图）。



需要注意的是，合并操作会保留所有的重边，即在上图操作后， w 向外连边数为 8 而不是 7。特别地，若 u, v 之间有多条重边，则 $G/(u, v)$ 只会删去其中一条，其余的重边将构成新图中 w 指向自己的自环。

图 $G = (V, E)$ 的 **Tutte 多项式** 是一个关于变量 x, y 的多项式，满足：

- 若 $E = \emptyset$ ，则 $T(G; x, y) = 1$ 。
- 若 e 是 G 的一个桥边，则 $T(G; x, y) = xT(G - e; x, y)$ 。
- 若 e 是 G 的一个自环，则 $T(G; x, y) = yT(G - e; x, y)$ 。
- 若 e 既不是桥边，也不是自环，则 $T(G; x, y) = T(G - e; x, y) + T(G/e; x, y)$ 。

容易证明，一张图 G 的 Tutte 多项式是唯一的。设 $K_4(V, E)$ 为四阶完全图， $e \in E$ ，则 $K_4 - e$ 的 Tutte 多项式可能是： ▲

- (A). $x^3 + 2x^2 + x + 2xy + y + y^2$ (B). $x^3 + x^2 + x + 2xy + y + y^2$
 (C). $x^3 + 2x^2 + 2x + 2xy + y + y^2$ (D). $x^3 + x^2 + 2x + 2xy + y + y^2$

15. 满足 $k! = \prod_{i=0}^{n-1} (2^n - 2^i)$ 的正整数对 (k, n) 的数量有 ▲ 个

- (A).1 (B).2 (C).3 (D).4

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 \checkmark ，错误填 \times ；除特殊说明外，判断题每题 1.5 分，选择题每题 4 分，共 40 分）

（一）（12 分）阅读下面一段程序，完成 16 ~ 21 六道小题。

给定长为 n 的序列 a ， a_i 的元素均为 $[0, 10^9]$ 内的整数。计算

$$\sum_{i=1}^n [2 \nmid i] \sum_{\substack{i|j \\ 1 \leq j \leq n}} a_j$$

对 $10^9 + 7$ 取模后的值。其中 $[P]$ 当命题 P 为真时值为 1，否则值为 0。

```
1 #include <cstdio>
2 #include <vector>
3
4 const int mod = 1e9 + 7;
5 std::vector<int> vec;
6
7 inline int inc(int x, int y) { x += y - mod; return x + (x >> 31 & mod); }
8 inline int mul(int x, int y) { return 1ll * x * y % mod; }
9
10 int main()
11 {
12     int n;
13     scanf("%d", &n);
14     vec.push_back(0);
15     for (int i = 1; i <= n; ++i)
16     {
17         int x;
18         scanf("%d", &x);
19         vec.push_back(x);
20     }
21     int ans = 0;
22     for (int i = 1; i <= n; ++i)
23         if (i & 1)
24             for (int j = i; j <= n; j += i)
25                 ans = inc(ans, mul(vec[i], vec[j]));
26     printf("%d\n", ans);
27     return 0;
28 }
```

判断题：

16. (1 分) `vec.push_back(x)` 的含义为向容器 `vec` 的尾部插入元素 x 。 ▲
 17. (1 分) 对于 $0 \leq i \leq 2^{31} - 1$ ，语句 `i&1` 的含义与 $i \bmod 2$ 等价。 ▲
 18. 该程序需要注意运算时可能超出 `int` 类型所能容纳的最大数据的风险。 ▲
 19. 函数 `inc(x, y)` 的含义为返回 $(x + y) \bmod (10^9 + 7)$ 的值 ($0 \leq x, y < 10^9 + 7$)。 ▲

选择题：

20. (3 分) 该算法的时间复杂度为： ▲。
 (A). $\Theta(n)$ (B). $\Theta(n^2)$ (C). $\Theta(n \log^2 n)$ (D). $\Theta(n \log n)$

21. 下列说法错误的为： ▲ 。
- (A).该算法的空间复杂度为 $\Theta(n)$ 。
- (B).在循环语句前执行 `vec.push_back(0)` 是因为 `std::vector` 容器下标从 0 开始计数。
- (C).若将 `inc` 函数接收的参数类型改为 `long long`，该函数仍将返回预期的结果。
- (D).若要对 $x, y \in [0, 10^9 + 7)$ 进行运算 $(x - y) \bmod (10^9 + 7)$ ，可以调用函数 `inc(x, mod - y)`。

(二) (13 分) 阅读下面一段程序，完成 22 ~ 27 六道小题。

```
1  #include <bits/stdc++.h>
2
3  const int N = 1e6 + 50;
4
5  int n, m, min[N], out[N], f[N], siz[N], ans;
6
7  struct edge
8  {
9      int u, v, w;
10 } e[N];
11
12 inline int find(int x)
13 {
14     while (x != f[x])
15         x = f[x] = f[f[x]];
16     return x;
17 }
18
19 inline void merge(int x, int y)
20 {
21     if (siz[x] > siz[y]) std::swap(x, y);
22     f[x] = y;
23 }
24
25 int main()
26 {
27     scanf("%d%d", &n, &m);
28     for (int i = 1; i <= n; ++i)
29     {
30         f[i] = i, siz[i] = 1;
31     }
32     for (int i = 1; i <= m; ++i)
33     {
34         scanf("%d%d%d", &e[i].u, &e[i].v, &e[i].w);
35     }
36     int components = n;
37     while (components > 1)
38     {
39         memset(min, 0x3f, sizeof min);
40         for (int i = 1; i <= m; ++i)
41         {
42             int u = find(e[i].u), v = find(e[i].v), w = e[i].w;
```

```
43         if (u != v)
44         {
45             if (w < min[u]) min[u] = w, out[u] = v;
46             if (w < min[v]) min[v] = w, out[v] = u;
47         }
48     }
49     for (int i = 1; i <= n; ++i)
50     {
51         int x = find(i);
52         if (out[x])
53         {
54             int y = find(out[x]);
55             if (x != y)
56             {
57                 merge(x, y);
58                 --components;
59                 ans += min[x];
60             }
61         }
62     }
63     printf("%d", ans);
64     return 0;
65 }
66 }
```

提示：该程序的输入为一张 n 个点 m 条边的连通无向图。输入格式为：

- 输入的第一行包含两个整数 $n, m (1 \leq n \leq m \leq 100)$ 。
- 接下来一行，包含三个整数 $u, v, w (1 \leq u, v \leq n, 1 \leq w \leq 100)$ 。

判断题：

22. (1 分) 该程序实现了用于求无向图 $G = (V, E)$ 的最小生成树的算法。 ▲
23. (1 分) 程序中使用了邻接矩阵来存储图 $G = (V, E)$ 。 ▲
24. 为了确保该算法的结果符合预期，输入需要保证所有的 w 互不相同。 ▲
25. 对于任意合法的输入，该程序一定会在有限步内返回结果。 ▲

选择题：

26. 变量 `components` 在运行过程中可达到的最小值为 ▲ 。
- (A).1 (B).0 (C). $\lfloor \frac{n}{2} \rfloor$ (D). $m - (n - 1)$
27. 该程序的时间复杂度为 ▲ 。
- (A). $\Theta(nm)$ (B). $\Theta(n^2)$ (C). $\Theta(n \log n)$ (D). $\Theta(m \log n)$

(三) (15 分) 阅读下面一段程序, 完成 28 ~ 33 六道小题。

```
1 #include <bits/stdc++.h>
2
3 const int N = 2e6 + 50;
4 int n, head[N], nxt[N], ver[N], dep[N], prv[N], suc[N], cnt;
5 inline void add(int u, int v)
6 {
7     nxt[++cnt] = head[u], ver[cnt] = v, head[u] = cnt;
8 }
9 void dfs(int u, int fa)
10 {
11     dep[u] = dep[fa] + 1;
12     for (int i = head[u]; i; i = nxt[i])
13         if (ver[i] != fa) dfs(ver[i], u);
14 }
15 void solve(int u, int fa, bool rv = false)
16 {
17     int isLeaf = true, t = u;
18     for (int i = head[u]; i; i = nxt[i])
19     {
20         int y = ver[i];
21         if (y != fa)
22         {
23             isLeaf = false;
24             solve(y, u, rv ^ 1);
25             if (rv)
26             {
27                 int mp = prv[y];
28                 suc[t] = y, prv[y] = t, t = mp;
29             }
30             else
31             {
32                 suc[t] = suc[y];
33                 prv[suc[y]] = t;
34                 suc[t = y] = 0;
35             }
36         }
37     }
38     if (isLeaf) suc[u] = prv[u] = u;
39     else suc[t] = u, prv[u] = t;
40 }
41 int main()
42 {
43     scanf("%d", &n);
44     for (int i = 1, u, v; i < n; ++i)
45     {
46         scanf("%d%d", &u, &v);
47         add(u, v); add(v, u);
48     }
49     dfs(1, 1); solve(1, -1);
50     printf("1 ");
```

```
51     int p = 1, v = suc[p];
52     while (v != p)
53     {
54         printf("%d ", v);
55         v = suc[v];
56     }
57     return 0;
58 }
```

提示: 该程序的输入为一张 n 个点的连通图。

判断题:

28. (1 分) 由程序的建图方式, 可以猜测输入的图 G 为一棵树。 ▲
29. 函数 `dfs(int, int)` 用于计算每个节点的深度, 并存储至数组 `dep[]`。 ▲
30. 函数 `solve()` 通过广度优先搜索, 在树 T 上构造了一条路径。 ▲
31. (2 分) 程序将输出一个大小为 n 的排列。 ▲

选择题:

32. 该程序读入以下数据后, 输出结果为: ▲。

```
1 8
2 1 2
3 2 3
4 3 4
5 4 5
6 4 6
7 4 7
8 7 8
```

- (A).1 3 8 7 5 6 4 2 (B).1 3 7 8 6 5 4 2 (C).1 3 8 7 5 6 2 4 (D).1 3 7 8 6 5 2 4

33. (5 分) 该程序对于给定的图 G , 构造出另一个图 G' , 并在 G' 上构造某条特殊的路径。记 $d_G(u, v)$ 为图 G 中 u, v 两点的距离, 则有关 $G'(V', E')$ 的构造方法与找出的路径的说法, 正确的为: ▲。

- (A). $V' = V, E' = \{(u, v) \mid u, v \in V, d_G(u, v) \leq 2\}$, 找出的路径为哈密顿路径。
- (B). $V' = V, E' = \{(u, v) \mid u, v \in V, d_G(u, v) \leq 2\}$, 找出的路径为欧拉路径。
- (C). $V' = V, E' = \{(u, v) \mid u, v \in V, d_G(u, v) \leq 3\}$, 找出的路径为哈密顿路径。
- (D). $V' = V, E' = \{(u, v) \mid u, v \in V, d_G(u, v) \leq 3\}$, 找出的路径为欧拉路径。

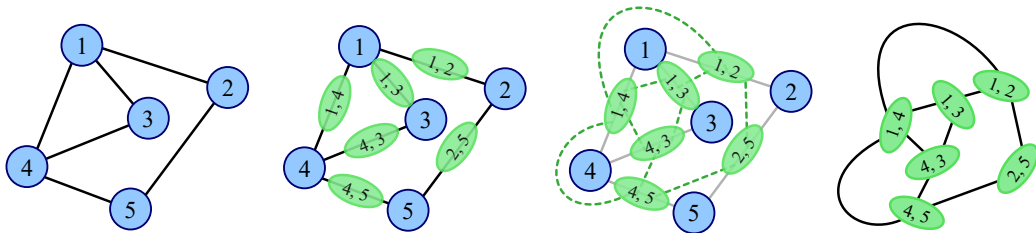
三、完善程序 (单选题, 每小题 3 分, 共 30 分)

(一) (15 分) 阅读下面一段程序, 完成 34 ~ 38 五道小题。

对于图 $G = (V, E)$, 我们定义它的线图 $L(G)$ 为一张 $|E|$ 个点的无向图, 满足:

- 原图 G 中的边 e 对应 $L(G)$ 中的一个点 u 。
- $L(G)$ 中点 u, v 之间有连边, 当且仅当 G 中 u, v 对应的边有公共的顶点。

下图展示了如何通过 G 构造出 $L(G)$ 。



现在，给定一棵树 T ，你需要计算出它的 k 阶线图 $L^k(T)$ 的点数。其中 k 阶线图的定义如下：

$$L^k(G) = \begin{cases} G & k = 0 \\ L(L^{k-1}(G)) & k > 0 \end{cases}$$

输入格式：输入的第一行包含两个整数 n, k 。接下来 $n - 1$ 行，每行两个整数 u, v ，描述树中的一条边。

输出格式：输出一行一个整数，表示答案。

数据范围： $1 \leq k \leq 5, 1 \leq n \leq 50$ 。

```
1 #include <bits/stdc++.h>
2
3 const int N = 5e3 + 7;
4 int n, k, d[N];
5 std::vector<int> p[N];
6 bool e[N][N];
7
8 int main() {
9     scanf("%d%d", &n, &k); --n;
10    for (int i = 1, x, y; i <= n; i++)
11    {
12        scanf("%d%d", &x, &y);
13        for (int j : p[x]) e[i][j] = e[j][i] = 1;
14        for (int j : p[y]) e[i][j] = e[j][i] = 1;
15        ____ (1) ____;
16    }
17    int ans = 0;
18    if (k == 2)
19    {
20        for (int i = 1; i <= n; i++)
21            for (int j = 1; j <= n; j++)
22                ____ (2) ____;
23        ____ (3) ____;
24    }
25    else if (k == 3)
26    {
27        for (int i = 1; i <= n; i++)
28            for (int j = 1; j <= n; j++)
29                d[i] += e[i][j];
30        for (int i = 1; i <= n; i++) ____ (4) ____;
31    }
32    else if (k == 4)
33    {
34        for (int i = 1; i <= n; i++)
35            for (int j = 1; j <= n; j++) d[i] += e[i][j];
36        for (int i = 1; i <= n; i++)
37            for (int j = 1; j <= n; j++)
38                if (e[i][j])
39                    ____ (5) ____;
40        ans /= 4;
41    }
42    else if (k == 5)
43    {
```

```
44        for (int i = 1; i <= n; i++)
45            for (int j = 1; j <= n; j++) d[i] += e[i][j];
46        static int c[N][N];
47        for (int i = 1; i <= n; i++)
48        {
49            int x1 = 0, x2 = 0;
50            for (int j = 1; j <= n; j++)
51                if (e[i][j])
52                    c[i][j] = d[i] + d[j] - 3, x1 += c[i][j], x2 += c[i][j] * c[i][j];
53            for (int j = 1; j <= n; j++)
54                if (e[i][j])
55                    ans += (d[i] - 1) * c[i][j] * c[i][j], ans += 2 * c[i][j] * (x1 -
56                        c[i][j]),
57                    ans += x2 - c[i][j] * c[i][j], ans -= (d[i] - 1) * c[i][j], ans -=
58                        x1 - c[i][j];
59        }
60        ans /= 4;
61    }
62    printf("%d", ans);
63    return 0;
64 }
```

34. ____ (1) ____ 处应填 ▲。
- (A).p[x].push_back(i), p[y].push_back(i) (B).p[x].push_back(y), p[y].push_back(x)
- (C).p[i].push_back(x), p[i].push_back(y) (D).p[x].push_back(y), p[i].push_back(x)
35. ____ (2) ____ 处应填 ▲。
- (A).ans += e[i][j] (B).ans += e[i][j] * 2
- (C).ans += n - e[i][j] (D).ans += n - e[i][j] * 2
36. ____ (3) ____ 处应填 ▲。
- (A).++ans (B).--ans
- (C).ans < <= 1 (D).ans > >= 1
37. ____ (4) ____ 处应填 ▲。
- (A).ans += d[i] * (d[i] + 1) / 2 (B).ans += d[i] * (d[i] - 1) / 2
- (C).ans += d[i] * (d[i] + 1) (D).ans += d[i] * (d[i] - 1)
38. ____ (5) ____ 处应填 ▲。
- (A).ans += (d[i] + d[j]) * (d[i] + d[j] - 1) (B).ans += (d[i] + d[j] - 1) * (d[i] + d[j] - 2)
- (C).ans += (d[i] + d[j] - 2) * (d[i] + d[j] - 3) (D).ans += (d[i] + d[j] - 3) * (d[i] + d[j] - 4)

(二) (15 分) 阅读下面一段程序, 完成 39 ~ 43 五道小题。

给定 n 个区间 $[l_1, r_1], [l_2, r_2], \dots, [l_n, r_n]$ 。你要求出有多少种不同的方案, 给每个区间涂上黑白两种颜色之一, 使得任意同色区间两两交集为空集。

答案可能很大, 因此要对 998,244,353 取模。

输入格式: 输入的第一行包含一个整数 n 。接下来 n 行, 每行两个整数 l_i, r_i 。

输出格式: 输出一行一个整数, 表示答案。

数据范围: $1 \leq n \leq 10^6, 1 \leq l_i \leq r_i \leq 10^9$ 。

```
1 #include <bits/stdc++.h>
2
3 const int N = 5e6 + 50;
4 const int mod = 998244353;
5
6 int n, top, head[N], nxt[N], ver[N], col[N], cnt;
7 std::pair<int, int> stack[N];
8
9 struct seg
10 {
11     int l, r;
12 } p[N];
13
14 inline void add(int u, int v)
15 {
16     nxt[++cnt] = head[u], ver[cnt] = v, head[u] = cnt;
17     nxt[++cnt] = head[v], ver[cnt] = u, head[v] = cnt;
18 }
19
20 // 从 x 出发 dfs, 检测当前图是否为二分图
21 bool dfs(int x, int c)
22 {
23     if (____(3)____) return col[x] == c;
24     col[x] = c;
25     for (int i = head[x]; i; i = nxt[i])
26         if (dfs(ver[i], c ^ 1) == false)
27             return false;
28     return true;
29 }
30
31 int main()
32 {
33     scanf("%d", &n);
34     for (int i = 1; i <= n; ++i) scanf("%d%d", &p[i].l, &p[i].r);
35     std::sort(p + 1, p + n + 1, [](seg u, seg v)
36     {
37         return ____ (1) ____;
38     }); // 将所有区间按照右端点从小到大排序
39     for (int i = 1; i <= n; ++i)
40     {
41         while (top > 0 && ____ (2) ____) // 若区间有交, 则进行建边
42             add(stack[top--].second, i);
43         stack[++top] = std::make_pair(p[i].r, i);
```

```
44     }
45     memset(col, -1, sizeof col);
46     int ans = 1;
47     for (int i = 1; i <= n; ++i)
48         if (col[i] == -1)
49         {
50             if (dfs(i, 0) == false) return printf("0"), 0; // 若区间图不是二分图, 则无解
51             ans = (____(4)____) % mod;
52         }
53     int mx[2]; mx[0] = mx[1] = 0;
54     for (int i = 1; i <= n; ++i)
55     {
56         if (____(5)____) return printf("0"), 0;
57         mx[col[i]] = p[i].r;
58     }
59     printf("%d", ans);
60     return 0;
61 }
```

39. ____ (1) ____ 处应填 ▲。

(A).u.l < v.l (B).u.l > v.l

(C).u.r < v.r (D).u.r > v.r

40. ____ (2) ____ 处应填 ▲。

(A).stack[top].first < p[i].l (B).stack[top].first > p[i].l

(C).stack[top].first < p[i].r (D).stack[top].first > p[i].r

41. ____ (3) ____ 处应填 ▲。

(A).col[x] & 1 (B).col[x] | 1

(C).!col[x] (D).~col[x]

42. ____ (4) ____ 处应填 ▲。

(A).ans + 1 (B).ans * ans

(C).ans / 2 (D).ans * 2

43. ____ (5) ____ 处应填 ▲。

(A).mx[col[i]] > p[i].l (B).mx[col[i]] > p[i].r

(C).col[i] > p[i].l (D).col[i] > p[i].r