

认证时间: 2022 年 8 月 23 日 09:30~11:30

认证时间: 2022 年 8 月 23 日 09:30~11:30

- 试题纸共有 17 页，答题纸共有 1 页，满分 100 分。请在答题纸上作答，写在试题纸上的一律无效。
- 不得使用任何电子设备（如计算器、手机、电子词典等）或查阅任何书籍资料。

1.  $(1047)_8 = ( \quad )_{10}$ 。

- A.  $(B \vee C \vee D) \vee D \wedge A$       B.  $((\neg A \wedge B) \vee C) \wedge \neg B$   
C.  $(A \wedge B) \vee \neg(C \wedge D \vee \neg A)$       D.  $A \wedge (D \vee \neg C) \wedge B$

- ```
int f(int x) {
    if(x <= 2)
        return 1;
    int ans = f(x - 1) + f(x - 2);
    ans %= 10000;
    return ans;
}
```

A. 运行时间超时  
B. 栈溢出  
C. 访问无效内存  
D. 返回错误的答案

- A.  $-a/*b-c-def$                       B.  $abcd-*e/+f-$   
C.  $+ab*-cd/e-f$                       D.  $f-e/d-d*b+a$



```

int ans=0;
void dfs(int x){
    ++ans;
    if(x<=1)return;
    dfs(x/3);
    if(x/3+1<x) dfs(x/3+1);
    if(x/3+2<x) dfs(x/3+2);
}

```

A. 24                      B. 25                      C. 26                      D. 27

13. 最大子段和问题，即给出一个长度为  $n$  的序列  $a$ ，选出其中连续且非空的一段使得这段和最大。当我们采用分治算法去计算这道题的结果时（不可直接动态规划且采用最优的分治方式），最优平均复杂度为（ ）。

A.  $\Theta(n \log^2 n)$                       B.  $\Theta(n \log n)$   
C.  $\Theta(n)$                       D.  $\Theta(1)$

14. 若某算法的时间计算表示为递推关系式：

$$T(n)=9T(n/3)+n$$

$$T(1)=1$$

则该算法的时间复杂度是（ ）

A.  $\Theta(n)$                       B.  $\Theta(2^n)$                       C.  $\Theta(n^2)$                       D.  $\Theta(n \log n)$

15. 中国计算机协会成立于（ ）年。

A. 1961                      B. 1962                      C. 1971                      D. 1972

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 2 分，选择题 3 分，共计 40 分）

1.

```

1  #include <stdio>
2  #include <string>
3
4  const int maxn = 1003;
5
6  int type, n, m;
7  char s[maxn], t[maxn];
8
9  int main() {
10     scanf("%d %s %s", &type, t, s);
11     n = strlen(s); m = strlen(t);
12     if (type < 2) {
13         for (int i = 0; i < m; ++i) s[i] = t[i];
14     } else if (type == 2) {
15         strcpy(s, t);

```

```

16 // 提示：如果此时调用 printf("%s\n", s)，则本次输出结果为整个 t 字符串和换行，没有其他字符。
17     } else {
18         for (int i = 0; i < m; i += 4) {
19             unsigned int code = 0, pos = i;
20             for (int j = 1; pos < i+4; j*=100, ++pos) {
21                 if (pos == m) break;
22                 code += t[pos] * j;
23             }
24             pos = i;
25             while (code != 0) {
26                 s[pos++] = code % 100;
27                 code /= 100;
28             }
29         }
30     }
31     for (int i = 0; i < n; ++i) printf("%c", s[i]);
32     printf("\n");
33 }

```

输入保证  $t$  的长度不大于  $s$  的长度，且两字符串均只含有大小写字母，不是空串， $type = 1, 2, 3$ ，完成下面的判断题和单选题：

● 判断题

- 1) 将程序中所有的小于号 ( $<$ ) 改为不等于号 ( $\neq$ )，则程序对所有符合要求的输入的输出结果**不变**。 ( )
- 2) 当输入为 1 xyz abcd 时，程序的输出为 xyzd。 ( )
- 3) 程序在输入为 1 xyz abcd 时的输出与输入为 2 xyz abcd 的输出**相同**。 ( )
- 4) 将程序第 25~28 行的 while 循环替换为 do-while 循环（判断条件和循环体不变），则程序对同一合法输入的输出结果**一定不变**。( )

● 单选题

- 5) 若将程序第 13 行改为 `for (int i = 0; i < strlen(t); ++i) s[i] = t[i];`，且已知输入的  $type$  一定为 1 的情况下，用  $n$  表示  $s$  的长度， $m$  表示  $t$  的长度，则程序的时间复杂度为 ( )。
 

|                      |                        |
|----------------------|------------------------|
| A. $\Theta(n + m)$   | B. $\Theta(n + m^2)$   |
| C. $\Theta(n^2 + m)$ | D. $\Theta(n^2 + m^2)$ |
- 6) 给程序分别输入选项 ( ) 的两组输入数据，得到的输出**不同**。
 

|                        |                        |
|------------------------|------------------------|
| A. 1 ab abc 和 3 ab abc | B. 1 AB ABC 和 3 AB ABC |
| C. 1 de fgh 和 3 de fgh | D. 1 DE FGH 和 3 DE FGH |

2.

```
1  #include <iostream>
2  using namespace std;
3
4  const int INF = 1000000000;
5  #define Front 0
6  #define Back 1
7  #define Left 2
8  #define Right 3
9  #define Up 4
10 #define Down 5
11 int w[6], a[1003][1003];
12 const int way1[] = {Up, Right, Down, Left};
13 const int way2[] = {Up, Front, Down, Back};
14 const int way3[] = {Left, Front, Right, Back};
15 int get_max(int &a, int b) {
16     return a = max(a, b);
17 }
18 int right_rotate(int &u) {
19     for (int i = 0; i < 4; ++ i)
20         if (u == way1[i])
21             return u = way1[(i + 1) % 4];
22     return u;
23 }
24 int front_rotate(int &u) {
25     for (int i = 0; i < 4; ++ i)
26         if (u == way2[i])
27             return u = way2[(i + 1) % 4];
28     return u;
29 }
30 const int anchorX = Up;
31 const int anchorY = Front;
32 const int anchorZ = Right;
33 int find_down(int u, int v) {
34     if (u == Down || u == Up) return anchorX^(u == Up);
35     if (v == Down || v == Up) return anchorY^(v == Up);
36     for (int i = 0; i < 4; ++ i)
37         if (u == way3[i])
38             return anchorZ ^ (v == way3[(i + 1) % 4]);
39     return -1;
40 }
41 int n, m, dp[1003][1003][6][6];
42 int main() {
43     cin >> n >> m;
```

```

44     for (int i = 0; i < n; ++ i)
45         for (int j = 0; j < m; ++ j)
46             cin >> a[i][j];
47     for (int i = 0; i < 6; ++ i)
48         cin >> w[i];
49     for (int i = 0; i < n; ++ i)
50         for (int j = 0; j < m; ++ j)
51             for (int a = 0; a < 6; ++ a)
52                 for (int b = 0; b < 6; ++ b)
53                     dp[i][j][a][b] = -INF;
54     dp[0][0][anchorX][anchorY] = a[0][0] * w[Down];
55     for (int i = 0; i < n; ++ i)
56         for (int j = 0; j < m; ++ j)
57             for (int p = 0; p < 6; ++ p)
58                 for (int q = 0; q < 6; ++ q)
59                     if (dp[i][j][p][q] != -INF) {
60                         int x = dp[i][j][p][q];
61                         int u1 = p, v1 = q;
62                         right_rotate(u1);
63                         right_rotate(v1);
64                         get_max(dp[i][j + 1][u1][v1],
65                             x + w[find_down(u1, v1)] * a[i][j + 1]);
66                         int u2 = p, v2 = q;
67                         front_rotate(u2);
68                         front_rotate(v2);
69                         get_max(dp[i + 1][j][u2][v2],
70                             x + w[find_down(u2, v2)] * a[i + 1][j]);
71                     }
72     int ans = -INF;
73     for (int p = 0; p < 6; ++ p)
74         for (int q = 0; q < 6; ++ q)
75             ans = max(ans, dp[n - 1][m - 1][p][q]);
76     printf("%d\n", ans);
77     return 0;
78 }

```

以下程序的输入数据的绝对值均不超过 $10^3$ 。完成下面的判断题和单选题：

● 判断题

- 1) 存在一种合法的输入数据，使得运行程序时，某次 `find_down` 函数的返回值是 `-1`。(    )
- 2) 该程序的时间复杂度为  $\Theta(n^2m^2)$ 。(    )

- 3) 对于任意  $u \in [0,6)$ , 「先执行 `front_rotate(u)`, 再执行 `right_rotate(u)`」, 与「先执行 `right_rotate(u)`, 再执行 `front_rotate(u)`」, 最终  $u$  的值相同。( )

● 单选题

- 4) 将 `anchorX`、`anchorY`、`anchorZ` 依次更换为 ( ) 时, 对于全部合法数据, 与改变之前的输出结果无异。

A. Left、Front、Down  
B. Left、Up、Front  
C. Left、Down、Back  
D. Down、Right、Front

- 5) (2分) 对于以下的输入数据, 输出结果为 ( )。

```
5 5
2 8 15 1 10
5 19 19 3 5
6 6 2 8 2
12 16 3 8 17
12 5 3 14 13
1 1 1 1 1 1
```

A. 95                      B. 97                      C. 94                      D. 103

- 6) (2分) 对于以下的输入数据, 输出结果为 ( )。

```
2 5
2 8 15 3 10
5 19 19 3 5
1 2 3 4 5 6
```

A. 194                      B. 157                      C. 193                      D. 201

3.

```
1  #include <bits/stdc++.h>
2  using namespace std;
3  const int MAXN = 309;
4  const int MAXM = 109;
5  const int MAXP = 100000;
6  typedef long long ll;
7  int n, m, seed, rt, btm, s[MAXN], Len, dfncnt, full_dist, P, t;
8  int lg[MAXN], dfn[MAXN], fa[MAXN], dep[MAXN], lson[MAXN];
9  vector<int> e[MAXN];
10 vector<int> L[MAXM], leaves;
11 int F[MAXM];
12 namespace LCA {
13     int st[24][MAXN];
```

```

14 int minNode(int x, int y){return dep[x]<dep[y] ? x:y;}
15 void init() {
16     for (int i = 1; i <= n; ++i) st[0][dfn[i]]=fa[i];
17     for (int i = 1; i <= lg[n]; ++i)
18         for (int j = 1; j + (1 << i) - 1 <= n; ++j)
19             st[i][j] = minNode(st[i - 1][j], st[i - 1][j + (1
<< (i - 1))]);
20 }
21 int lca(int u, int v) {
22     if (u == v) return u;
23     if ((u = dfn[u]) > (v = dfn[v])) swap(u, v);
24     int d = lg[v - u++];
25     return minNode(st[d][u], st[d][v - (1 << d) + 1]);
26 }
27 }
28 namespace Gen {
29     mt19937 rnd;
30     int pos[MAXN], deg[MAXN];
31     vector <pair<int, int>> edges;
32     void calc() {
33         for (int i = 1; i <= n - 2; ++i)
34             ++deg[pos[i]];
35         set <int> ret; ret.clear();
36         for (int i = 1; i <= n; ++i)
37             if (deg[i] == 1) ret.insert(i);
38         for (int i = 1; i <= n - 2; ++i) {
39             int npos = *ret.begin();
40             edges.push_back(make_pair(npos, pos[i]));
41             ret.erase(ret.begin());
42             --deg[pos[i]];
43             if (deg[pos[i]] == 1)
44                 ret.insert(pos[i]);
45         }
46         edges.push_back(make_pair(*ret.begin(), n));
47     }
48     void build() {
49         for (auto i : edges) {
50             int u = i.first, v = i.second;
51             e[u].push_back(v); e[v].push_back(u);
52         }
53     }
54     void generate(int seed) {
55         rnd.seed(seed);
56         edges.clear();

```



```

57     for (int i = 1; i <= n; ++i) deg[i] = 1;
58     for (int i = 1; i <= n - 2; ++i)
59         pos[i] = rnd() % n + 1;
60     calc();
61     build();
62 }
63 }
64 int dfs(int u, int _fa, int &bottom) {
65     dfn[u] = ++dfncnt;
66     if (e[u].size() == 1) leaves.push_back(u);
67     lson[u] = -1; fa[u] = _fa; dep[u] = dep[_fa] + 1;
68     int maxlen = 0, dson = u, temp;
69     for (int v: e[u])
70         if (v != _fa) {
71             int p = dfs(v, u, temp);
72             if (p > maxlen) maxlen = p, lson[u]=v, dson=temp;
73         }
74     bottom = dson;
75     return maxlen + 1;
76 }
77 #define dist(u, v) (dep[u]+dep[v]-2*dep[LCA::lca(u, v)])
78 int v[MAXP+9], prime[MAXP + 9], prime_cnt, vis[MAXP+9];
79 void prime_init(int n) {
80     prime_cnt = 0;
81     for (int i = 2; i <= n; ++i) {
82         if (!v[i]) prime[++prime_cnt] = v[i] = i;
83         for(int j=1; j <= prime_cnt && i*prime[j]<=n; ++j) {
84             if (v[i] < prime[j]) break;
85             v[i * prime[j]] = prime[j];
86         }
87     }
88 }
89 vector<int> answer, gline;
90 void solve() {
91     cin >> n >> m >> seed >> t;
92     lg[0] = lg[1] = 0;
93     for (int i = 2; i <= n; ++i) lg[i] = lg[i >> 1] + 1;
94     for (int i = 1; i <= n; ++i) e[i].clear();
95     leaves.clear();
96     if (!t) Gen::generate(seed);
97     else {
98         for (int i = 1, u, v; i < n; ++i) {
99             cin >> u >> v;
100             e[u].push_back(v); e[v].push_back(u);

```

```

101     }
102 }
103 dep[0] = 0; dfs(1, 0, rt);
104 dfncnt=0; leaves.clear();
105 Len = dfs(rt, 0, btm); LCA::init();
106 for (int k = rt; ~k; k = lson[k]) gline.push_back(k);
107 for (int i = 1, tmp; i <= m; ++i) { {
108     cin >> s[i];
109     L[i].clear();
110     for(int j = 1; j <= s[i]; ++j) {
111         cin >> tmp; L[i].push_back(tmp);
112     }
113     F[i] = dist(L[i].front(), L[i].back());
114     for (unsigned j = 0; j < L[i].size() - 1; ++j)
115         F[i] += dist(L[i][j], L[i][j + 1]);
116     int tmp = F[i] >> 1;
117     while (tmp > 1) vis[v[tmp]] = 1, tmp /= v[tmp];
118 }
119 full_dist = dist(leaves.front(), leaves.back());
120 for (unsigned i = 0; i < leaves.size() - 1; ++i)
121     full_dist += dist(leaves[i], leaves[i + 1]);
122 P = -100000;
123 for (int i = 2; i <= prime_cnt; ++i)
124     if (full_dist+2*Len<prime[i] * 2 && !vis[prime[i]]){
125         P = prime[i] * 2; break;
126     }
127 for (int i = 1; i <= m; ++i) {
128     F[i] >>= 1;
129     while (F[i]>1) vis[v[F[i]]] = 0, F[i] /= v[F[i]];
130 }
131 int left=P-full_dist;
132 int fcnt = left / (2 * (Len - 1)); answer = leaves;
133 while (fcnt--) answer.push_back(rt),
answer.push_back(btm), left -= 2 * (Len - 1);
134 if (left>>=1) answer.push_back(rt),
answer.push_back(gline[left]);
135 for (int qwq: answer) cout << qwq << " ";
136 cout << endl;
137 }
138 int main() {
139     prime_init(MAXP);
140     int T; cin >> T;
141     while (T--) solve();
142 }

```

数据保证，输入的  $3 \leq n \leq 300$ ,  $1 \leq m \leq 100$ ,  $1 \leq s[i] \leq n$ ,  $0 \leq t \leq 1$ , `seed` 在 `int` 范围内。当 `seed=623532` 时，`Gen::pos` 数组中的数值（从 1 开始）的前 23 项分别是 22, 25, 12, 23, 7, 11, 20, 4, 3, 13, 10, 2, 7, 18, 1, 9, 23, 13, 10, 5, 23, 18, 6。数据保证 `seed` 随机生成。

调用 `rnd()` 返回一个  $[0, 2^{32})$  的随机整数。完成下列判断题和单选题：

● 判断题

- 1) 调用 `LCA::lca()` 函数，单次操作的时间复杂度是  $O(\log n)$ 。 ( )
- 2) 在  $t=0$  的情况下，`len` 的大小在期望下与  $\sqrt{n}$  同阶。 ( )
- 3) `full_dist` 最大不会超过  $2n$ 。 ( )

● 单选题

- 4) (2分) 对于给定的如下数据后，输出的结果为 ( ) ?

```
1
5 2 123456 1
1 2
2 3
3 4
4 5
2 1 4
2 3 5
```

- |                |                |
|----------------|----------------|
| A. 2 5 1 5 1 5 | B. 4 3 4 3 4 1 |
| C. 5 1 5 1 5 2 | D. 4 1 4 1 4 5 |

- 5) (2分) 对于给定的如下数据后，输出的结果为 ( ) ?

```
1
25 4 623532 0
3 11 3 5
5 24 17 15 19 17
6 14 21 16 8 18 21
8 21 18 17 19 24 21 24 17
```

- |                                      |
|--------------------------------------|
| A. 19 24 17 5 17 19 24 4 3 20 11     |
| B. 19 24 17 15 8 16 21 14 19 8 19 13 |
| C. 14 18 21 8 9 21 14 8 15 24 17 19  |
| D. 14 18 21 8 9 21 14 8 22 16 14     |

- 6) 答案序列长度的理论上界 ( ) ?

- |                      |                       |
|----------------------|-----------------------|
| A. $\Theta(n^{0.5})$ | B. $\Theta(n \log n)$ |
| C. $\Theta(\log n)$  | D. $\Theta(n)$        |

### 三、完善程序（单选题，每小题 3 分，共计 30 分）

1. （凑出 17）给定  $n(1 \leq n \leq 20)$  个互不相同的正整数  $a_1, a_2, \dots, a_n (1 \leq a_i \leq 10^9)$ ，将之排成一行。你需要在每个  $a_i$  前加上一个加号(+)或减号(-)，使这  $n$  个数字组成一个算式。请问是否存在一种添加符号的方案，使该算式的值为 17？如果存在，请输出 Yes，否则输出 No。

例如，给定  $n = 5, a_1 = 1, a_2 = 4, a_3 = 5, a_4 = 9, a_5 = 8$ ，则  $-a_1 - a_2 + a_3 + a_4 + a_5 = 17$ 。

提示：使用穷举法解决这个问题。

试补全程序。

```
1  #include <cstdio>
2
3  using namespace std;
4
5  const int maxn = 25;
6  const int aim = 17;
7
8  int n;
9  int a[maxn];
10 bool ans;
11
12 int getBit(const int s, int p) {
13     return ①;
14 }
15
16 int main() {
17     scanf("%d", &n);
18     for (②) scanf("%d", a + i);
19     for (int s=0, upperBound = ③; s <= upperBound; ++s){
20         ④;
21         for (int j = 0; j < n; ++j) if (getBit(s, j) == 1){
22             sum += a[j];
23         } else {
24             ⑤;
25         }
26         if (int(sum) == aim) {
27             ans = true;
28             break;
29         }
30     }
```

```

31     printf("%s\n", ans ? "Yes" : "No");
32 }

```

1) ① 处应填 ( )。

- |                          |                          |
|--------------------------|--------------------------|
| A. $(s \gg p) \& 1$      | B. $(s \ll p) \& 1$      |
| C. $s \& (1 \ll p) \& 1$ | D. $s \& (1 \gg p) \& 1$ |

2) ② 处应填 ( )。

- |                                           |                                           |
|-------------------------------------------|-------------------------------------------|
| A. <code>int i = 0; i &lt;= n; ++i</code> | B. <code>int i = 1; i &lt;= n; ++i</code> |
| C. <code>int i = 0; i &lt; n; ++i</code>  | D. <code>int i = 1; i &lt; n; ++i</code>  |

3) ③ 处应填 ( )。

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

4) ④ 处应填 ( )。

- |                                        |                                            |
|----------------------------------------|--------------------------------------------|
| A. <code>int sum = 0</code>            | B. <code>unsigned long long sum = 0</code> |
| C. <code>unsigned short sum = 0</code> | D. <code>unsigned int sum = 0</code>       |

5) ⑤ 处应填 ( )。

- |                                   |                                   |
|-----------------------------------|-----------------------------------|
| A. <code>sum = a[j] + sum</code>  | B. <code>sum = a[j] - sum</code>  |
| C. <code>sum = -a[j] + sum</code> | D. <code>sum = -a[j] - sum</code> |

2. (树同构问题) 对于两棵有  $n$  个结点的有根树  $T_1$  和  $T_2$ , 如果存在一个长度为  $n$  的排列  $a_i$ , 使得对于  $T_1$  中的任意结点对  $(i, j)$ ,  $T_1$  中存在边  $i \rightarrow j$  当且仅当  $T_2$  中存在边  $a_i \rightarrow a_j$ , 则称  $T_1$  与  $T_2$  同构。现在给出两棵树, 求出它们是否同构。

为了解决该问题, 有一个算法叫 **AHU 算法**, 其时间复杂度为  $O(n \log n)$ , 步骤如下:

- 考虑将有根树转化为括号序列的递归算法, 我们在回溯时会将子树的括号序列依次拼接起来。
- 如果保证拼接子树括号序列得到的结果与遍历子树的顺序无关, 则可保证同构的有根树可以转换为完全一致的括号序列。
- 考虑回溯时将子树括号序列按字典序排序, 将以此方式得到的两棵树的括号序列进行比较即可得知两树是否同构。

于是我们得到了一个  $O(n^2)$  时间复杂度的朴素算法, 考虑下述优化:

- 考虑朴素算法的复杂度瓶颈, 在于括号序列的长度与子树大小相关。
- 递归时深度为  $k$  的结点对应的括号序列, 可以仅由深度为  $k+1$  的子结点的括号序列得出。

- 我们考虑对于每个深度  $k$ ，将两树中深度为  $k$  的结点对应的括号序列作为值域进行离散化。
- 将原先的子结点括号序列的拼接（字符串），用子结点离散化后结果的拼接（整数序列）替代。

这样我们就将与子树大小挂钩的括号序列替换成了与结点度数挂钩的序列，复杂度大大降低。试补全程序。

```

1  #include <iostream>
2  #include <vector>
3  #include <algorithm>
4
5  typedef std::vector<int> Nodes;
6
7  struct RootedTree {
8      int size, root;
9      std::vector<Nodes> edges;
10 };
11
12 RootedTree read() {
13     RootedTree res;
14     std::cin >> res.size >> res.root;
15     res.edges.resize(res.size + 1);
16
17     for (int i = 1; i < res.size; ++ i) {
18         int u, v;
19         std::cin >> u >> v;
20         res.edges[u].push_back(v);
21         res.edges[v].push_back(u);
22     }
23
24     return res;
25 }
26
27 std::vector<Nodes> D; // 某个深度下的所有结点
28 std::vector<int> fa;
29
30 int dfs_depth(const RootedTree &tree, int now, int f, int
depth, int nodesDiff) {
31     if (①) // 1
32         D.push_back(Nodes());
33
34     int nowId = now + nodesDiff;
35     D[depth].push_back(nowId);

```

```

36     if (f != -1)
37         fa[nowId] = f + nodesDiff;
38
39     int maxDepth = 0;
40     for (int i = 0; i < tree.edges[now].size(); ++ i) {
41         int child = tree.edges[now][i];
42         if (②) { // 2
43             maxDepth = std::max(maxDepth,
44                 dfs_depth(tree, child, now, depth+1, nodesDiff));
45         }
46     }
47     return maxDepth + 1;
48 }
49
50 typedef std::vector<int> Hash;
51
52 std::vector<Hash> hash;
53
54 bool cmp(const int x, const int y) {
55     return ③; // 3
56 }
57
58 bool check(const RootedTree &t1, const RootedTree &t2) {
59     if (t1.size != t2.size)
60         return false;
61     int size = t1.size;
62
63     D.clear();
64     fa.clear();
65     fa.resize(size * 2 + 1);
66
67     int dep1 = dfs_depth(t1, t1.root, -1, 0, 0);
68     int dep2 = dfs_depth(t2, t2.root, -1, 0, size);
69     if (dep1 != dep2)
70         return false;
71
72     hash.clear();
73     hash.resize(size * 2 + 1);
74     std::vector<int> rank(size * 2 + 1);
75
76     for (④) { // 4
77         for (int i = 0; i < D[h + 1].size(); ++ i) {
78             int node = D[h + 1][i];
79             hash[fa[node]].push_back(rank[node]);

```

```

80         }
81         std::sort(D[h].begin(), D[h].end(), cmp);
82         rank[D[h][0]] = 0;
83         for (int i = 1, cnt = 0; i < D[h].size(); ++ i) {
84             if (⑤) // 5
85                 ++ cnt;
86             rank[D[h][i]] = cnt;
87         }
88     }
89
90     return hash[t1.root] == hash[t2.root + size];
91 }
92
93 int main() {
94     RootedTree treeA, treeB;
95
96     treeA = read();
97     treeB = read();
98
99     bool isomorphism = check(treeA, treeB);
100
101     std::cout
102         << (isomorphism ? "wow!" : "emm...")
103         << std::endl;
104     return 0;
105 }

```

提示: `std::vector::operator<` 的判定方法为: 对两个 `vector a,b`, 找到最小的满足 `a[i]!=b[i]` 的 `i`, 返回 `a[i]<b[i]` 的判定结果。若这样的 `i` 不存在, 返回 `false`。

- 1) ① 处应填 (     )。
 

|                                     |                                     |
|-------------------------------------|-------------------------------------|
| A. <code>D.size() &lt; depth</code> | B. <code>D.size() &gt; depth</code> |
| C. <code>D.size() != depth</code>   | D. <code>D.size() == depth</code>   |
- 2) ② 处应填 (     )。
 

|                                                    |                                        |
|----------------------------------------------------|----------------------------------------|
| A. <code>child != f + nodesDiff</code>             | B. <code>child != f</code>             |
| C. <code>child + nodesDiff == f + nodesDiff</code> | D. <code>child + nodesDiff != f</code> |
- 3) ③ 处应填 (     )。
 

|                                              |                                                    |
|----------------------------------------------|----------------------------------------------------|
| A. <code>rank[fa[x]] &lt; rank[fa[y]]</code> | B. <code>hash[x].size() &lt; hash[y].size()</code> |
|----------------------------------------------|----------------------------------------------------|



- C. rank[x] < rank[y]
- D. hash[x] < hash[y]

4) ④ 处应填 (      )。

- A. int h = dep1-1; ~h; --h
- B. int h = dep1-1; h; --h
- C. int h = dep1-2; ~h; --h
- D. int h = dep1-2; h; --h

5) ⑤ 处应填 (      )。

- A. hash[D[h][i]] != hash[D[h][i - 1]]
- B. rank[D[h][i]] != rank[D[h][i - 1]]
- C. hash[D[h][i]] < hash[D[h][i - 1]]
- D. rank[D[h][i]] < rank[D[h][i - 1]]