提高组 CSP-S 2024 初赛模拟卷 1

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. 表达式(8 %(-6))与(-8 % 6)的值为（　　）。

A. -2，-2　　B. 2，-2　　C. -2，2　　D. 2，2

2. 设根结点深度为 0，一棵深度为 h 的满三叉树，即除最后一层无任何子结点外，每一层上的所有结点都有 3 个子结点的树，共有（　　）个结点。

A. $(3^{h+1}-1)/2$　　B. 3^{h-1}　　C. 3^h　　D. $(3^{h-1}-1)/2$

3. 以下不属于调试工具 GDB 中的命令的是（　　）。

A. printf　　B. display　　C. next　　D. step

4. 以下哪个不属于算法的最本质特征？（　　）

A. 有穷性　　B. 确定性　　C. 先进性　　D. 确切性

5. 以下哪个不属于哈希冲突的常见处理方法？（　　）

A. 哈希法　　B. 线性探查法　　C. 二次探查法　　D. 固定分配法

6. 单源最短路问题的常用算法不包含（　　）。

A. Bellman-Ford　　B. Dijkstra　　C. SPFA　　D. Kruskal

7. 设某算法的时间复杂度函数的递推方程是 F(n) = F(n - 1) + 2n（n 为正整数）及 F(0) = 0，则该算法的时间复杂度为（　　）。

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

8. 一棵二叉树一共有 19 个结点，其叶子结点不可能有（　　）个。

A. 11　　B. 10　　C. 9　　D. 1

9. 关于拓扑排序，下面的说法哪个是正确的？（　　）

A. 拓扑排序只能用广度优先遍历实现

B. 每个有向图都至少存在一个拓扑排序
C. 拓扑排序一定从入度为 0 的结点开始
D. 拓扑排序的方案一定是唯一的

10. 下列说法中哪个不是树的性质？（ ）
A. 任意两个结点之间有且只有一条简单路径
B. 树中有可能有一个简单环
C. 边的数目恰好是顶点数目减 1
D. 树中无环

11. 对于完全背包问题（给出 n 种物品和一个容积为 m 的背包，每种物品有无限个，单个大小为 $v[i]$，价值为 $w[i]$，要求选择合适的物品放入背包，满足大小不超过容积且价值最大），设 $f[i]$表示用去 i 的空间能获得的最大价值，倒序枚举 i 为使用的空间，正序枚举 j 为物品编号，则可写出动态转移方程（ ）。
A. `f[i] = max(f[i], f[i-w[j]] + v[j])`
B. `f[i] = max(f[i], f[i-v[j]] + w[j])`
C. `f[i] = min(f[i], f[i-w[j]] + v[j])`
D. `f[i] = min(f[i], f[i-v[j]] + w[j])`

12. 前缀表达式`**a+bcd`的后缀表达式是（ ）。
A. `abcd*++` B. `abc+*d*` C. `a*bc+*d` D. `b+c*a*d`

13. 某学校“信奥”小组有男生和女生各 3 名，现从中挑选 2 名学生去参加市科技节编程竞赛，至少有 1 名男生参加的概率是（ ）。
A. 0.5 B. 0.6 C. 0.7 D. 0.8

14. 四面体的顶点及各棱中点加起来共有 10 个点，在其中取 4 个不共面的点，不同的取法共有（ ）种。
A. 150 B. 147 C. 144 D. 141

15. 设全集 E={a，b，c，d，e}，集合 A={a，c}，B={a，b，d}，C={b，d}，则集合(A∩B)∪~C 为（ ）。
A. {a} B. {c，e} C. {a，e} D. {a，c，e}

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
04 int dp[502][502];
05
06 int main() {
07    string s;
08    cin >> s;
09    for (int j=0; j<=s.size(); j++) {
10       for (int i = 0; i<s.size()-j; i++) {
11          dp[i][i+j] = dp[i+1][i+j] + 1;
12          for (int k=i+1; k<=i+j; k++) {
13             if (s[k] == s[i]) {
14       dp[i][i+j]=min(dp[i][i+j],dp[i+1][k-1]+dp[k+1][i+j]);
15             }
16          }
17       }
18    }
19    cout << dp[0][s.size()-1] << endl;
20    return 0;
21 }
```

■ 判断题

16. 将第 11 行改为 dp[i][j]=INT_MAX-1，代码的输出结果会改变。 （ ）
17. 修改第 13 行为 if (s[i] == s[j])，代码的输出结果不变。 （ ）
18. 用贪心算法可以实现与本程序同样的功能。 （ ）
19. 将第 12 行中 for 循环的 k 的初始值从 i + 1 改为 i，代码的输出结果不变。 （ ）

■ 选择题

20. 本程序的时间复杂度是（ ）。

A. $O(n^2)$ B. $O(n^3)$ C. $O(n^3 \log n)$ D. $O(n^4)$

21. 对于输入 aagog，本程序的输出为（　　）。

A. 1　　B. 2　　C. 3　　D. 5

（2）

```
01 #include <bits/stdc++.h>
02
03 using namespace std;
04 pair<int, int> pi;
05
06 int parent[100000], sz[100000];
07
08 int n, m, components;
09
10 void initialize(){
11    for(int i=0; i<n; i++){
12       parent[i] = i;
13       sz[i] = 1;
14    }
15 }
16
17 int find(int a){
18    if(parent[a] == a){
19       return a;
20    }
21    return find(parent[a]);
22 }
23
24 bool sameDSU(int a, int b){
25    return find(a) == find(b);
26 }
27
28 void merge(int a, int b){
29    a = find(a), b = find(b);
30    if (a == b) return;
31    --components;
32    if (sz[a] > sz[b]) swap(a, b);
33    parent[a] = b;
34    sz[b] += sz[a];
35 }
36
```

```
37 int findsz(int a){
38     return sz[find(a)];
39 }
40
41 int main(){
42     cin >> n >> m;
43     initialize();
44     components = n;
45     int maxComponents = 1;
46     for(int i=0; i<m; i++){
47         int a, b; cin >> a >> b;
48         merge(a, b);
49         maxComponents = max(maxComponents, findsz(a));
50         cout << components << " " << maxComponents << endl;
51     }
52 }
```

■ 判断题

22. 如果输入的 a 和 b 从 1 开始编号，程序依然可以正常运行。 ( )
23. 该程序计算了无向图的连通分量数。 ( )
24. 合并操作总是将较小树的根结点合并到较大树的根结点下。 ( )
25. 为了确保该程序不发生错误，需要保证输入的 a 和 b 互不相同。 ( )

■ 选择题

26. 在程序运行结束时，maxComponents 变量可以存储的最大值是（ ）。

A. n/2 B. n C. m/2 D. m

27. 如果合并操作 merge 总是将结点 b 合并到结点 a，不论它们各自的 sz 大小如何，程序的最坏运行时间会如何变化？（ ）

A. 变快 B. 保持不变 C. 变慢 D. 无法确定

（3）

```
01 #include<bits/stdc++.h>
02 using namespace std;
03 typedef long long ll;
04
05 const int MAXN = 2e5 + 7;
```

```
06 const int BITS = 30;
07 int numNodes, nodeValues[MAXN];
08 int trieNodeCount, trie[MAXN*BITS][2], maxIndex[MAXN*BITS];
09
10 void insert(int value, int index) {
11    int currentNode = 0;
12    for(int bit = BITS - 1; bit >= 0; bit--) {
13       int currentBit = (value >> bit) & 1;
14       if(!trie[currentNode][currentBit]) {
15          trie[currentNode][currentBit] = trieNodeCount++;
16       }
17       currentNode = trie[currentNode][currentBit];
18       maxIndex[currentNode]=max(maxIndex[currentNode],index);
19    }
20 }
21
22 int query(int value, int index) {
23    int currentNode = 0, result = 0;
24    for(int bit = BITS - 1; bit >= 0; bit--) {
25       int currentBit = (value >> bit) & 1;
26       if(trie[currentNode][currentBit] && maxIndex[trie[currentNode]
             [currentBit]] >= index) {
27          currentNode = trie[currentNode][currentBit];
28       } else {
29          currentNode = trie[currentNode][1 - currentBit];
30          result += (1 << bit);
31       }
32    }
33    return result;
34 }
35
36 ll weight;
37 void findMinimumXor(int left, int right, int bitPosition) {
38    if(left == right || bitPosition < 0) return;
39    int mid = left;
40    while(mid<right && !((nodeValues[mid]>>bitPosition)&1)) mid++;
41    findMinimumXor(left, mid, bitPosition - 1);
42    findMinimumXor(mid, right, bitPosition - 1);
43    if(mid == right || mid == left) return;
44    int minEdgeWeight = INT_MAX;
45    for(int i = left; i < mid; i++) {
```

```
46          minEdgeWeight=min(minEdgeWeight,query(nodeValues[i],mid));
47      }
48      weight += minEdgeWeight;
49  }
50
51  int main() {
52      trieNodeCount = 1;
53      cin >> numNodes;
54      for(int i=0; i<numNodes; i++) cin >> nodeValues[i];
55      sort(nodeValues, nodeValues + numNodes);
56      for(int i=0; i<numNodes; i++) insert(nodeValues[i], i);
57      findMinimumXor(0, numNodes, BITS - 1);
58      cout << weight << endl;
59      return 0;
60  }
```

注：输入的 $n \leq 1000$，(a，b)代表一条边，输入保证所有的边会组成一棵树。

■ **判断题**

28.（2 分）当输入的任意一个 nodeValues[i]为 0 时，程序会发生越界。 （ ）

29.（2 分）如果将第 55 行改为 sort(nodeValues, nodeValues + numNodes, greater<int>());，那么代码的输出结果不变。 （ ）

30.（3 分）输入的 nodeValues 数组中的每个元素必须互不相同。 （ ）

■ **选择题**

31. 当输入为

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

时，程序的输出为（ ）。

A. 5　　B. 8　　C. 1　　D. 4

32. 如果增大 BITS 常量，程序将受什么影响？（ ）

A. 运行速度将提高　　B. 能处理更大的整数

C. 需要更多的内存空间　　D. 不会有任何变化

33. 如果输入的 nodeValues 全为 0，则 weight 的值将是（ ）。

A. 0　　B. numNodes-1　　C. BITS　　D. 无法确定

三、完善程序（单选题，每小题 3 分，共计 30 分）

（1）输入一个整数 N 和 N 个$[0, 2^{21}-1]$范围内的整数，求出从任意位置开始的最大区间异或和。如果有多个这样的子区间，则选择最短子区间，输出区间的左右端点。

```
#include<bits/stdc++.h>
using namespace std;
const int MAXN = 1e5 + 10;
const int MAXM = MAXN * 21;
int n, x;
int s[MAXN], id[MAXM];
int tree[MAXM][2], idx;

void insert(int x, int k) {
    int p = 0;
    for (int i = 20; i >= 0; i--) {
        int u = x >> i & 1;
        if (!tree[p][u]) ①;
        ②;
    }
    id[p] = k;
}

int query(int x) {
    int p = 0;
    for (int i = 20; i >= 0; i--) {
        int u = x >> i & 1;
        if ( ③ ) p = tree[p][u ^ 1];
       else p = tree[p][u];
    }
    return id[p];
}

int main() {
    cin >> n;
    insert(s[0], 0);
    int res = -1;
    int l, r;
```

```
34    for (int i = 1; i <= MAXN; ++i) {
35        cin >> x;
36        s[i] = s[i - 1] ^ x;
37        int k = query(s[i]);
38        int maxn = ④;
39        if (maxn > res) {
40            res = maxn;
41            l = k + 1, r = i;
42        }
43        ⑤;
44    }
45    cout << res << " " << l << " " << r << endl;
46    return 0;
47 }
```

34. ①处应填（　　）。

A. tree[p][u] = ++idex　　B. tree[p][u] = tree[p][u^1]

C. tree[p][u] = ++p　　D. tree[p^1][u] = ++idex

35. ②处应填（　　）。

A. p = idex　　B. p = u

C. p = tree[p][u]　　D. p = ++p

36. ③处应填（　　）。

A. tree[p][u]　　B. !tree[p][u^1]

C. !tree[p][u]　　D. tree[p][u^1]

37. ④处应填（　　）。

A. s[i]^s[k]　　B. s[r]^s[k]

C. s[i]^x　　D. s[l]^s[r]

38. ⑤处应填（　　）。

A. insert(s[i], i)　　B. insert(s[r], r-1)

C. insert(s[r], r)　　D. insert(s[i], i-1)

（2）给定一个 $n \times m$ 的矩阵，将 $r \times c$ 的矩阵分成 $r' \times c'$ 和 $r \times (c-c')$ 的矩阵需要 r^2 的代价。求分出一个大小和为 k 的矩阵所需的最小代价。

输入样例 1：

2 2 1

输出样例 1：

5

输入样例 2：

2 2 3

输出样例 2：

5

样例 1 解释：一共需要进行 2 次操作。

i. 把 2×2 的矩阵分为两个 2×1 的矩阵，代价为 $2^2=4$。

ii. 把 2×1 的矩阵分为两个 1×1 的矩阵，代价为 $1^2=1$，此时得到 1×1 的矩阵的和为 1，总代价为 1+4=5。

样例 2 解释：一共需要进行 2 次操作。

i. 把 2×2 的矩阵分为两个 2×1 的矩阵，代价为 $2^2=4$。

ii. 把 2×1 的矩阵分为两个 1×1 的矩阵，代价为 $1^2=1$，此时得到 1×1 的矩阵和 2×1 的矩阵的和为 3，总代价为 1+4=5。

```
01 #include<bits/stdc++.h>
02 using namespace std;
03 const int MAXN = 1e5 + 10;
04 const int MAXM = MAXN * 21;
05 int n, x;
06 int s[MAXN], id[MAXM];
07 int tree[MAXM][2], idx;
08
09 void insert(int x, int k) {
10     int p = 0;
11     for (int i = 20; i >= 0; i--) {
12         int u = x >> i & 1;
13         if (!tree[p][u]) ①;
14         ②;
15     }
16     id[p] = k;
17 }
18
```

```
19 int query(int x) {
20    int p = 0;
21    for (int i = 20; i >= 0; i--) {
22       int u = x >> i & 1;
23       if (③) p = tree[p][u ^ 1];
24       else p = tree[p][u];
25    }
26    return id[p];
27 }
28
29 int main() {
30    cin >> n;
31    insert(s[0], 0);
32    int res = -1;
33    int l, r;
34    for (int i = 1; i <= MAXN; ++i) {
35       cin >> x;
36       s[i] = s[i - 1] ^ x;
37       int k = query(s[i]);
38       int maxn = ④;
39       if (maxn > res) {
40          res = maxn;
41          l = k + 1, r = i;
42       }
43       ⑤;
44    }
45    cout << res << " " << l << " " << r << endl;
46    return 0;
47 }
```

39. ①处应填（　　）。

A. f[x-1][y-1][z-1]　　B. f[x][y][z]

C. f[x%2][y%2][z%2]　　D. f[MAXN-x][MAXN-y][MAXN-z]

40. ②处应填（　　）。

A. z == 1　　B. z == x*y　　C. z == x+y　　D. z >= MAXN

41. ③处应填（　　）。

A. INF　　B. -MAXN　　C. 0　　D. MAXN

42. ④处应填（　　）。

A. `y * y + solve(y, i, j) + solve(x-i, y, z-j)`

B. `y * y + solve(i, y, j) + solve(x-i, y, z-j)`

C. `(x-i) * (x-i) + solve(x-i, i, j) + solve(x-i, y, z)`

D. `(x-i) * (x-i) + solve(i, x-i, j) + solve(x, y-i, z-j)`

43. ⑤处应填（　　）。

A. `res = f[x][y][z]`

B. `f[x-1][y-1][z-1] = res`

C. `f[x][y][z] = res`

D. `res += f[x][y][z]`