

## CSP 2020 提高级 2020

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. 请选出以下最大的数( )。

- A.  $(550)_{10}$                       B.  $(777)_8$                       C.  $2^{10}$                       D.  $(22F)_{16}$

2. 操作系统的功能是( )。

- A. 负责外设与主机之间的信息交换  
B. 控制和管理计算机系统的各种硬件和软件资源的使用  
C. 负责诊断机器的故障  
D. 将源程序编译成目标程序

3. 现有一段 8 分钟的视频文件, 它的播放速度是每秒 24 帧图像, 每帧图像是幅分辨率为  $2048 \times 1024$  像素的 32 位真彩色图像。请问要存储这段原始无压缩视频, 需要多大的存储空间?( )。

- A. 30G                      B. 90G                      C. 150G                      D. 450G

4. 今有一空栈 S, 对下列待进栈的数据元素序列 a, b, c, d, e, f 依次进行: 进栈, 进栈, 出栈, 进栈, 进栈, 出栈的操作, 则此操作完成后, 栈底元素为( )。

- A. b                      B. a                      C. d                      D. c

5. 将 (2, 7, 10, 18) 分别存储到某个地址区间为  $0 \sim 10$  的哈希表中, 如果哈希函数  $h(x) = ( )$ , 将不会产生冲突, 其中  $a \bmod b$  表示 a 除以 b 的余数。

- A.  $x^2 \bmod 11$                       B.  $2x \bmod 11$   
C.  $x \bmod 11$                       D.  $[x/2] \bmod 11$ , 其中  $[x/2]$  表示  $x/2$  下取整

6. 下列哪些问题不能用贪心法精确求解?( )

- A. 霍夫曼编码问题                      B. 0-1 背包问题  
C. 最小生成树问题                      D. 单源最短路径问题

7. 具有 n 个顶点, e 条边的图采用邻接表存储结构, 进行深度优先遍历运算的时间复杂度为( )。

- A.  $O(n+e)$                       B.  $O(n^2)$                       C.  $O(e^2)$                       D.  $O(n)$

8. 二分图是指能将顶点划分成两个部分, 每一部分内的顶点间没有边相连的简单无向图。那么, 24 个顶点的二分图至多有( )条边。

- A. 144                      B. 100                      C. 48                      D. 122

9. 广度优先搜索时, 一定需要用到的数据结构是( )。

- A. 栈                      B. 二叉树                      C. 队列                      D. 哈希表

10. 一个班学生分组做游戏, 如果每组三人就多两人, 每组五人就多三人, 每组七人就多四人, 问这个班的学生人数 n 在以下哪个区间? 已知  $n < 60$ 。( )。

- A.  $30 < n < 40$                       B.  $40 < n < 50$                       C.  $50 < n < 60$                       D.  $20 < n < 30$

11. 小明想通过走楼梯来锻炼身体, 假设从第 1 层走到第 2 层消耗 10 卡热量, 接着从第 2 层走到第 3 层消耗 20 卡热量, 再从第 3 层走到第 4 层消耗 30 卡热量, 依此类推, 从第 k 层走到第 k+1 层消耗  $10k$  卡热量 ( $k > 1$ )。如果小明想从 1 层开始, 通过连续向上爬楼梯消耗 1000 卡热量, 至少要爬到第几层楼?( )。

- A. 14                      B. 16                      C. 15                      D. 13

12. 表达式  $a*(b+c)-d$  的后缀表达形式为( )。

- A.  $abc*+d-$                       B.  $-+*abcd$                       C.  $abcd*+-$                       D.  $abc*+d-$

13. 从一个  $4 \times 4$  的棋盘选取不在同一行也不在同一列上的两个方格, 共有( )种方法。

- A. 60                      B. 72                      C. 86                      D. 64

14. 对一个 n 个顶点、m 条边的带权有向简单图用 Dijkstra 算法计算单源最短路径时, 如果不使用堆或其它优先队列进行优化, 则其时间复杂度为( )。

- A.  $O((m+n^2) \log n)$                       B.  $O(mn+n^3)$                       C.  $O((m+n) \log n)$                       D.  $O(n^2)$

15. 1948 年, ( ) 将热力学中的熵引入信息通信领域, 标志着信息论研究的开端。

- A. 欧拉 (Leonhard Euler)                      B. 冯·诺伊曼 (John von Neumann)  
C. 克劳德·香农 (Claude Shannon)                      D. 图灵 (Alan Turing)

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 1.5 分, 选择题 3 分, 共计 40 分)

1.

```
01 #include <iostream>
02 using namespace std;
03
04 int n;
05 int d[1000];
06
07 int main() {
08     cin >> n;
09     for (int i = 0; i < n; ++i)
10         cin >> d[i];
11     int ans = -1;
12     for (int i = 0; i < n; ++i)
13         for (int j = 0; j < n; ++j)
14             if (d[i] < d[j])
15                 ans = max(ans, d[i] + d[j] - (d[i] & d[j]));
16     cout << ans;
17     return 0;
18 }
```

假设输入的  $n$  和  $d[i]$  都是不超过 10000 的正整数, 完成下面的判断题和单选题:

● 判断题

- 1)  $n$  必须小于 1000, 否则程序可能会发生运行错误。( )
- 2) 输出一定大于等于 0。( )
- 3) 若将第 13 行的 “ $j=0$ ” 改为 “ $j=i+1$ ”, 程序输出可能会改变。( )
- 4) 将第 14 行的 “ $d[i]<d[j]$ ” 改为 “ $d[i]\neq d[j]$ ”, 程序输出不会改变。( )

● 单选题

- 5) 若输入  $n$  为 100, 且输出为 127, 则输入的  $d[i]$  中不可能有( )。  
A. 127                      B. 126                      C. 128                      D. 125
- 6) 若输出的数大于 0, 则下面说法正确的是( )。  
A. 若输出为偶数, 则输入的  $d[i]$  中最多有两个偶数  
B. 若输出为奇数, 则输入的  $d[i]$  中至少有两个奇数  
C. 若输出为偶数, 则输入的  $d[i]$  中至少有两个偶数  
D. 若输出为奇数, 则输入的  $d[i]$  中最多有两个奇数

2.

```
01 #include <iostream>
02 #include <cstdlib>
03 using namespace std;
04
05 int n;
06 int d[10000];
07
08 int find(int L, int R, int k) {
```

```

09     int x = rand() % (R - L + 1) + L;
10     swap(d[L], d[x]);
11     int a = L + 1, b = R;
12     while (a < b) {
13         while (a < b && d[a] < d[L])
14             ++a;
15         while (a < b && d[b] >= d[L])
16             --b;
17         swap(d[a], d[b]);
18     }
19     if (d[a] < d[L])
20         ++a;
21     if (a - L == k)
22         return d[L];
23     if (a - L < k)
24         return find(a, R, k - (a - L));
25     return find(L + 1, a - 1, k);
26 }
27
28 int main() {
29     int k;
30     cin >> n;
31     cin >> k;
32     for (int i = 0; i < n; ++i)
33         cin >> d[i];
34     cout << find(0, n - 1, k);
35     return 0;
36 }

```

假设输入的 $n, k$ 和 $d[i]$ 都是不超过10000的正整数，且 $k$ 不超过 $n$ ，并假设 $\text{rand}()$ 函数产生的是均匀的随机数，完成下面的判断题和单选题：

●判断题

- 1) 第9行的“ $x$ ”的数值范围是 $L+1$ 到 $R$ ，即 $[L+1, R]$ 。( )
- 2) 将第19行的“ $d[a]$ ”改为“ $d[b]$ ”，程序不会发生运行错误。( )

●单选题

- 3) (2.5分) 当输入的 $d[i]$ 是严格单调递增序列时，第17行的“swap”平均执行次数是( )。  
 A.  $O(n \log n)$                       B.  $O(n)$                       C.  $O(\log n)$                       D.  $O(n^2)$
- 4) (2.5分) 当输入的 $d[i]$ 是严格单调递减序列时，第17行的“swap”平均执行次数是( )。  
 A.  $O(n^2)$                       B.  $O(n)$                       C.  $O(n \log n)$                       D.  $O(\log n)$
- 5) (2.5分) 若输入的 $d[i]$ 为 $i$ ，此程序①平均的时间复杂度和②最坏情况下的时间复杂度分别是( )。  
 A.  $O(n), O(n^2)$                       B.  $O(n), O(n \log n)$   
 C.  $O(n \log n), O(n^2)$                       D.  $O(n \log n), O(n \log n)$
- 6) (2.5分) 若输入的 $d[i]$ 都为同一个数，此程序平均的时间复杂度是( )。  
 A.  $O(n)$                       B.  $O(\log n)$                       C.  $O(n \log n)$                       D.  $O(n^2)$

```

3.
01 #include <iostream>
02 #include <queue>
03 using namespace std;
04
05 const int maxl = 20000000;

```

```

06
07 class Map {
08     struct item {
09         string key; int value;
10     } d[max1];
11     int cnt;
12 public:
13     int find(string x) {
14         for (int i = 0; i < cnt; ++i)
15             if (d[i].key == x)
16                 return d[i].value;
17         return -1;
18     }
19     static int end() { return -1; }
20     void insert(string k, int v) {
21         d[cnt].key = k; d[cnt++].value = v;
22     }
23 } s[2];
24
25 class Queue {
26     string q[max1];
27     int head, tail;
28 public:
29     void pop() { ++head; }
30     string front() { return q[head + 1]; }
31     bool empty() { return head == tail; }
32     void push(string x) { q[++tail] = x; }
33 } q[2];
34
35 string st0, st1;
36 int m;
37
38 string LtoR(string s, int L, int R) {
39     string t = s;
40     char tmp = t[L];
41     for (int i = L; i < R; ++i)
42         t[i] = t[i + 1];
43     t[R] = tmp;
44     return t;
45 }
46
47 string RtoL(string s, int L, int R) {
48     string t = s;
49     char tmp = t[R];
50     for (int i = R; i > L; --i)
51         t[i] = t[i - 1];
52     t[L] = tmp;
53     return t;
54 }
55
56 bool check(string st, int p, int step) {

```

```

57     if (s[p].find(st) != s[p].end())
58         return false;
59     ++step;
60     if (s[p ^ 1].find(st) == s[p].end()) {
61         s[p].insert(st, step);
62         q[p].push(st);
63         return false;
64     }
65     cout << s[p ^ 1].find(st) + step << endl;
66     return true;
67 }
68
69 int main() {
70     cin >> st0 >> st1;
71     int len = st0.length();
72     if (len != st1.length()) {
73         cout << -1 << endl;
74         return 0;
75     }
76     if (st0 == st1) {
77         cout << 0 << endl;
78         return 0;
79     }
80     cin >> m;
81     s[0].insert(st0, 0); s[1].insert(st1, 0);
82     q[0].push(st0); q[1].push(st1);
83     for (int p = 0;
84         !(q[0].empty() && q[1].empty());
85         p ^= 1) {
86         string st = q[p].front(); q[p].pop();
87         int step = s[p].find(st);
88         if ((p == 0 &&
89             (check(LtoR(st, m, len - 1), p, step) ||
90              check(RtoL(st, 0, m), p, step)))
91             ||
92             (p == 1 &&
93              (check(LtoR(st, 0, m), p, step) ||
94               check(RtoL(st, m, len - 1), p, step))))
95             return 0;
96     }
97     cout << -1 << endl;
98     return 0;
99 }

```

●判断题

1) 输出可能为 0。( )

- 2) 若输入的两个字符串长度均为 101 时, 则  $m=0$  时的输出与  $m=100$  时的输出是一样的。( )
- 3) 若两个字符串(长度均为  $n$ , 则最坏情况下, 此程序的时间复杂度为  $O(n!)$ 。( )

● 单选题

- 4) (2.5 分) 若输入的第一个字符串长度由 100 个不同的字符构成, 第二个字符串是第一个字符串的倒序, 输入的  $m$  为 0, 则输出为( )。

A. 49                                      B. 50                                      C. 100                                      D. -1

- 5) (4 分) 已知当输入为 “0123\n3210\n1” 时输出为 4, 当输入为 “012345\n543210\n1” 时输出为 14, 当输入为 “01234567\n76543210\n1” 时输出为 28, 则当输入为 “0123456789ab\nba9876543210\n1” 输出为( )。其中 “\n” 为换行符。

A. 56                                      B. 84                                      C. 102                                      D. 68

- 6) (4 分) 若两个字符串的长度均为  $n$ , 且  $0 < m < n-1$ , 且两个字符串的构成相同(即任何一个字符在两个字符串中出现的次数均相同), 则下列说法正确的是( )。提示: 考虑输入与输出有多少对字符前后顺序不一样。

- A. 若  $n$ 、 $m$  均为奇数, 则输出可能小于 0。
- B. 若  $n$ 、 $m$  均为偶数, 则输出可能小于 0。
- C. 若  $n$  为奇数、 $m$  为偶数, 则输出可能小于 0。
- D. 若  $n$  为偶数、 $m$  为奇数, 则输出可能小于 0。

三、完善程序(单选题, 每小题 3 分, 共计 30 分)

1. (分数背包) 小 S 有  $n$  块蛋糕, 编号从 1 到  $n$ 。第  $i$  块蛋糕的价值是  $w_i$ , 体积是  $v_i$ 。他有一个大小为  $B$  的盒子来装这些蛋糕, 也就是说装入盒子的蛋糕的体积总和不能超过  $B$ 。

他打算选择一些蛋糕装入盒子, 他希望盒子里装的蛋糕的价值之和尽量大。

为了使盒子里的蛋糕价值之和更大, 他可以任意切割蛋糕。具体来说, 他可以选择一个  $a$  ( $0 < a < 1$ ), 并将一块价值是  $w$ , 体积为  $v$  的蛋糕切割成两块, 其中一块的价值是  $a \times w$ , 体积是  $a \times v$ , 另一块的价值是  $(1-a) \times w$ , 体积是  $(1-a) \times v$ 。他可以重复无限次切割操作。

现要求编程输出最大可能的价值, 以分数的形式输出。

比如  $n=3, B=8$ , 三块蛋糕的价值分别是 4、4、2, 体积分别是 5、3、2。那么最优的方案就是将体积为 5 的蛋糕切成两份, 一份体积是 3, 价值是 2.4, 另一份体积是 2, 价值是 1.6, 然后把体积是 3 的那部分和后两块蛋糕打包进盒子。最优的价值之和是 8.4, 故程序输出 42/5。

输入的数据范围为:  $1 \leq n \leq 1000, 1 \leq B \leq 105; 1 \leq w_i, v_i \leq 100$ 。

提示: 将所有的蛋糕按照性价比  $w_i/v_i$  从大到小排序后进行贪心选择。试补全程序。

```
01 #include <cstdio>
02 using namespace std;
03
04 const int maxn = 1005;
05
06 int n, B, w[maxn], v[maxn];
07
08 int gcd(int u, int v) {
09     if (v == 0)
10         return u;
11     return gcd(v, u % v);
12 }
13
14 void print(int w, int v) {
15     int d = gcd(w, v);
16     w = w / d;
17     v = v / d;
18     if (v == 1)
19         printf("%d\n", w);
```

```

20     else
21         printf("%d/%d\n" w, v);
22 }
23 void swap(int &x, int &y) {
24     int t = x; x = y; y = t;
25 }
26
27 int main() {
28     scanf("%d %d" &n, &B);
29     for (int i = 1; i <= n; i++) {
30         scanf("%d %d", &w[i], &v[i]);
31     }
32     for (int i = 1; i < n; i++)
33         for (int j = 1; j < n; j++)
34             if (①) {
35                 swap(w[j], w[j + 1]);
36                 swap(v[j], v[j + 1]);
37             }
38     int curV, curW;
39     if (②) {
40         ③
41     } else {
42         print(B * w[1] , v[1]);
43         return 0;
44     }
45     for (int i = 2; i <= n; i++)
46         if (curV + v[i] <= B) {
47             curV += v[i];
48             curW += w[i];
49         } else {
50             print (④);
51             return 0;
52         }
53     print(⑤);
54     return 0;
55 }

```

1)①处应填( )

- A.  $w[j]/v[j] < w[j+1]/v[j+1]$   
 C.  $v[j]*w[j+1] < v[j+1]*w[j]$

- B.  $w[j]/v[j] > w[j+1]/v[j+1]$   
 D.  $w[j]*v[j+1] < w[j+1]*v[j]$

2)②处应填( )

- A.  $w[1] \leq B$                       B.  $v[1] \leq B$

- C.  $w[1] \geq B$

- D.  $v[1] \geq B$

3)③处应填( )

- A. `print(v[1],w[1]);return 0;`  
 C. `print(w[1],v[1]);return 0;`

- B. `curV=0;curW=0;`

- D. `curV=v[1];curW=w[1];`

4)④处应填( )

- A. `curW* v[i] + curV * w[i], v[i]`

- B. `(curW - w[i])* v[i] + (B - curV) * w[i], v[i]`

- C. `curW + v[i], w[i]`

D.  $\text{curW} * v[i] + (B - \text{curV}) * w[i], v[i]$

5) ⑤处应填( )

A.  $\text{curW}, \text{curV}$

B.  $\text{curW}, 1$

C.  $\text{curV}, \text{curW}$

D.  $\text{curV}, 1$

2. (最优子序列) 取  $m=16$ , 给出长度为  $n$  的整数序列  $a_1, a_2, \dots, a_n (0 \leq a_i < 2^m)$ 。对于一个二进制数  $x$ , 定义其分值  $w(x)$  为  $x + \text{popcnt}(x)$ , 其中  $\text{popcnt}(x)$  表示  $x$  二进制表示中 1 的个数。对于一个子序列  $b_1, b_2, \dots, b_k$ , 定义其子序列分值  $S$  为  $w(b_1 \oplus b_2) + w(b_2 \oplus b_3) + w(b_3 \oplus b_4) + \dots + w(b_{k-1} \oplus b_k)$ 。其中  $\oplus$  表示按位异或。对于空子序列, 规定其子序列分值为 0。求一个子序列使得其子序列分值最大, 输出这个最大值。

输入第一行包含一个整数  $n (1 \leq n \leq 40000)$ 。接下来一行包含  $n$  个整数  $a_1, a_2, \dots, a_n$

提示: 考虑优化朴素的动态规划算法, 将前  $m/2$  位和后  $m/2$  位分开计算。

$\text{Max}[x][y]$  表示当前的子序列下一个位置的高 8 位是  $x$ 、最后一个位置的低 8 位是  $y$  时的最大价值。

试补全程序。

```
01 #include <iostream>
02
03 using namespace std;
04
05 typedef long long LL;
06
07 const int MAXN = 40000, M = 16, B = M >> 1, MS = (1 << B) - 1;
08 const LL INF = 1000000000000000LL;
09 LL Max[MS + 4][MS + 4];
10
11 int w(int x)
12 {
13     int s = x;
14     while(x)
15     {
16         ①;
17         s++;
18     }
19     return s;
20 }
21
22 void to_max(LL &x, LL y)
23 {
24     if(x < y)
25         x = y;
26 }
27
28 int main()
29 {
30     int n;
31     LL ans = 0;
32     cin >> n;
33     for(int x = 0; x <= MS; x++)
```

```

34         for(int y = 0; y <= MS; y++)
35             Max[x][y] = -INF;
36     for(int i = 1; i <= n ; i++)
37     {
38         LL a;
39         cin >> a;
40         int x = ② , y = a & MS;
41         LL v = ③;
42         for(int z = 0; z <= MS; z++)
43             to_max(v, ④);
44         for(int z = 0; z <= MS; z++)
45             ⑤;
46         to_max(ans , v);
47     }
48     cout << ans << endl;
49     return 0;
50 }

```

1)①处应填( )

A.  $x \gg 1$                       B.  $x^{\wedge}=x \& (x^{\wedge}(x+1))$                       C.  $x-=x|-x$                       D.  $x^{\wedge}=x \& (x^{\wedge}(x-1))$

2)②处应填( )

A.  $(a \& MS) \ll B$                       B.  $a \gg B$                       C.  $a \& (1 \ll B)$                       D.  $a \& (MS \ll B)$

3)③处应填( )

A.  $-INF$                       B.  $Max[y][x]$                       C.  $0$                       D.  $Max[x][y]$

4)④处应填( )

A.  $Max[x][z] + w(y^{\wedge}z)$                       B.  $Max[x][z] + w(a^{\wedge}z)$   
 C.  $Max[x][z] + w(x^{\wedge}(z \ll B))$                       D.  $Max[x][z] + w(x^{\wedge}z)$

5)⑤处应填( )

A.  $to\_max(Max[y][z], v + w(a^{\wedge}(z \ll B)))$   
 B.  $to\_max(Max[z][y], v + w((x^{\wedge}z) \ll B))$   
 C.  $to\_max(Max[z][y], v + w(a^{\wedge}(z \ll B)))$   
 D.  $to\_max(Max[x][z], v + w(y^{\wedge}z))$