

CSP-S 2020 第一轮试题详解

黄晓燕

(福州第三中学信息技术组 福州 350007)

摘 要 题目来源于中国计算机学会 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。本文针对题型设置、知识分布及考察要点,对试题做详细分析。

关键词 解析;时间复杂度;算法

中图法分类号 TP399 DOI:10.16707/j.cnki.fjpc.2021.02.063

The Task Analysis of CSP-S 2020 Round 1

HUANG Xiaoyan

(Department of Information Technology, Fuzhou No.3 Middle School, Fuzhou, China, 350003)

1 试卷分析

CSP2020-S 组第一试试卷综合考察了多方面知识点,并对常见算法时间复杂度理解与分析有进一步要求。

第一部分单选题主要考察了计算机基础知识、数学计算以及对常见算法的理解与分析。该部分细目如表 1 所示。

表 1 CSP2020-S1 选择题细目表

单选序号	考察知识点	单选序号	考察知识点
1	进制转换	2	操作系统
3	视频文件大小	4	栈的操作
5	哈希冲突	6	常见问题解法
7	算法复杂度	8	简单图论
9	数据结构	10	简单数学
11	简单数学	12	表达式转换
13	组合计数	14	算法复杂度
15	信息技术通识		

第二部分阅读程序包含位运算、复杂度分析及双向搜索,对选手的程序的阅读与分析能力提出了一定要求。

第三部分完善程序给出了清晰的问题描述与提示信息,需要选手充分理解题目所给出的信息并处理好实现细节,对选手的程序编写能力提出了一定要求。

2 单项选择题分析

(1) 请选出以下最大的数(C)

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

解析:正确选项 C。此题为不同进制间数字比大小,可化为同进制进行比较。 $C.2^{10}=1024$,容易排除 A、B。D 经过换算, $(22F)_{16}=(559)_{10}<1024$ 。

(2) 操作系统的功能是(B)。

- 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

解析:操作系统是管理计算机硬件与软件资源的计算机程序,同时也是计算机系统的内核与基石。操作系统的主要作用是提供用户接口,处理如管理与配置内存、决定系统资源供需的优先次序、控制输入与输出设备、操作网络与管理文件系统等

基本事务。

(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, 每帧图像是一幅分辨率为 2048×1024 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?(B)。

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

解析: 存储一幅画面需要的字节数: $2048 \times 1024 \times 32 / 8$, 共 $24 \times 8 \times 60$ 幅画面, $1G = 2^{30}$ 字节, 所以总的存储容量 $2048 \times 1024 \times 32 / 8 \times 24 \times 8 \times 60 / 2^{30} = 90G$ 。

(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(B)。

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

解析: 栈是一种先进后出的数据结构, 类似用桶堆积物品, 先推进(入栈)来的压在底下, 随后一件一件往上堆。取走(出栈)时, 只能从上面一件一件取。堆和取都在顶部进行, 底部一般是不动的。如图 1 所示: 根据待进栈的数据元素序列是 a、b、c、d、e、f, 依次进行进栈(1 进)、进栈(2 进)、出栈(此时栈顶是 2, 2 出栈, 栈顶变为 1)、进栈(3 进)、进栈(4 进)、出栈(栈顶是 4, 4 出, 栈顶变为 3)。

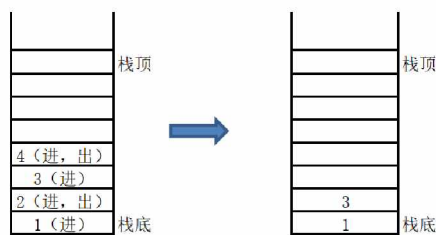


图 1 出入栈

(5) 将(2, 7, 10, 18)分别存储到某个地址区间为 0~10 的哈希表中, 如果哈希函数 $h(x) = (D)$, 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

解析: 哈希表是根据关键码值(Key value)而直接进行访问的数据结构。也就是说, 它通过把关键码值映射到表中一个位置来访问记录, 以加快查找的速度。本题提供 A、B、C、D 四个映射函数, 即

将(2, 7, 10, 18)分别作为 x 带入 A、B、C、D 四个函数中计算, 得到的值即为该数据在哈希表中存储的位置^[2]。分别带入后的结果如下:

$$\begin{aligned} \text{A: } & 2^2 \bmod 11 = 4 \\ & 7^2 \bmod 11 = 5 \\ & 10^2 \bmod 11 = 1 \\ & 18^2 \bmod 11 = 5 \end{aligned}$$

7 和 18 的在哈希表中的存储地址都是 5, 冲突。

$$\begin{aligned} \text{B: } & 2 \times 2 \bmod 11 = 4 \\ & 2 \times 7 \bmod 11 = 3 \\ & 2 \times 10 \bmod 11 = 9 \\ & 2 \times 18 \bmod 11 = 3 \end{aligned}$$

7 和 18 的在哈希表中的存储地址都是 3, 冲突。

$$\begin{aligned} \text{C: } & 2 \bmod 11 = 2 \\ & 7 \bmod 11 = 7 \\ & 10 \bmod 11 = 10 \\ & 18 \bmod 11 = 7 \end{aligned}$$

7 和 18 的在哈希表中的存储地址都是 7, 冲突。

$$\begin{aligned} \text{D: } & [2/2] \bmod 11 = 1 \\ & [7/2] \bmod 11 = 3 \\ & [10/2] \bmod 11 = 5 \\ & [18/2] \bmod 11 = 9 \end{aligned}$$

2、7、10、8 这 4 个数据的带入 D 选项函数, 计算出 4 个不同的存放地址, 无冲突。

(6) 下列哪些问题不能用贪心法精确求解?(B)

- 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

解析: A、C、D 均有贪心的思想, B 的问题用动态规划算法解决。

(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)。

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

解析: 每个点只会被访问一次, 每条边最多在这条边的起点被访问的时候访问一次。所以时间复杂度为 $\theta(n+e)$ 。

(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。24 个顶点的二分图至多有(A)条边。

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

解析: 本题为在二分图的概念上, 进行简单的排列组合。把点分为两部分, 每部分 12 个点, 本部之间的点不连边, 不同的部分的点都要有边, 就可以得到最多边数的二分图, 即 $12 \times 12 = 144$ 。

(9) 广度优先搜索时, 一定需要用到的数据结构是(C)。

A. 栈 B. 二叉树 C. 队列 D. 哈希表

解析: 广度优先搜索用到的数据结构是队列。队列是一种特殊的线性表, 特殊之处在于它只允许在表的前端(front)进行删除操作, 而在表的后端(rear)进行插入操作。和栈一样, 队列是一种操作受限制的线性表。深度优先搜索用到的数据结构是栈。

(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 $n < 60$ 。

(C)。

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

解析: 题目可以看成, 除3余2, 除5余3, 除7余4。没有同余的情况, 可以采用“逐步约束法”, 就是从“除7余4的数”中找出符合“除5余3的数”, 就在7上一加4, 直到所得的数除5余3, 得出数为18, 下面只要在18上一加7和5的最小公倍数35, 直到满足“除3余2”的数53。

$$4+7=11$$

$$11+7=18$$

$$18+35=53$$

(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?(C)。

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

解析: 本题为等差数列求和问题。首项为10, 公差为10, 问和超过1000的最少项数。等差数列求和公式 $S_n = n \cdot a_1 + n(n-1)d/2$, 依题意, 即 $n \cdot 10 + n(n-1) \cdot 10/2 \geq 1000$, 整理后即 $n^2 + n \geq 200$, $n=13$ 显然不对, $n=14$ 则满足该不等式, 即从第14层走向第15层后, 消耗的热量超过了1000。

(12) 表达式 $a*(b+c)-d$ 的后缀表达形式为(D)。

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

解析: 中缀转为后缀时, 可通过画出二叉树来实现, 画树的原则: 数字只能做叶子节点, 左操作数作为左子树, 右操作数作为右子树, 从最先被计

算的那个算式开始画。根据前根遍历, 得出前缀表达式。根据后根遍历, 得到后缀表达式^[3]。

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

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

解析: 任选一个, 去掉同列同行 $4+4-1=7$ 个, 剩 $16-7=9$ 个, $16 \cdot 9=144$ 种, 考虑有类似 (a,b) (b,a) 重复的情况, 则 $144/2=72$ 。

(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(D)。

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

解析: Dijkstra 算法最简单的实现方法是用一个链表或者数组来存储所有顶点的集合 Q , 所以搜索 Q 中最小元素的运算 (Extract-Min(Q)) 只需要线性搜索 Q 中的所有元素。这样的话, 算法的运行时间是 $O(n^2)^{[1]}$ 。

(15) 1948年, (C)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)

解析: 克劳德·艾尔伍德·香农 (Claude Elwood Shannon, 1916年4月30日—2001年2月24日) 是美国数学家, 信息论的创始人。1936年获得密歇根大学学士学位。1940年在麻省理工学院获得硕士和博士学位, 1941年进入贝尔实验室工作。香农提出了信息熵的概念, 为信息论和数字通信奠定了基础。主要论文有: 1938年的硕士论文《继电器与开关电路的符号分析》, 1948年的《通讯的数学原理》和1949年的《噪声下的通信》。

3 阅读程序分析

3.1 程序一

```
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“ $d[i] != d[j]$ ”, 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(C)。

- 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(C)。

- 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

解析: 本题等价于求出给定序列中两个不同元素二进制按位或的最大值求解。

(1) 当 $n=1000$ 时访问 d 数组的下标范围在 0~999 之间, 并不会发生运行错误, 故答案为×。

(2) 若 d 数组中所有元素相同则输出等于-1, 故答案为×。

(3) 可以发现, 当输入为长度大于 2 且严格

降序的序列时输出均改变, 故答案为√。

(4) 若 $d[x] != d[y]$, 则必然在 $i=x, j=y$ 或 $i=y, j=x$ 中的其中一种情况时满足 $d[i] < d[j]$, 故答案为√。

(5) $d[i] + d[j] - (d[i] \& d[j])$ 事实上等价于 $d[i] | d[j]$, 其中“ $|$ ”表示二进制按位或, 所以当 $d[i]$ 中含有 128 时输出必然大于等于 128, 故 C 选项正确。

(6) 若输入的 $d[i]$ 中含有的偶数的数量小于 2, 则(偶数|奇数)=奇数, (奇数|奇数)=奇数, 最终结果也只能是奇数, 故 C 选项正确。

3.2 程序二

```

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 `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 (无正确选项)。

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

(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 (B)。

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

(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)。

- A. $\theta(n)$, $\theta(n^2)$
B. $\theta(n)$, $\theta(n \log n)$
C. $\theta(n \log n)$, $\theta(n^2)$
D. $\theta(n \log n)$, $\theta(n \log n)$

(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 (D)。

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

解析: 本题实现了用快排的思想求解全局第 k 大数的过程。

(1) 当 $\text{rand}() \% (R-L+1) = 0$ 时 $x=L$, 故答案为×。

(2) 当 $L < R$ 时最终必然有 $a=b$ 在范围内, 当 $L=R$ 时事实上原来的程序运行可能出现错误, 但手动模拟可知按照选项改动后无误, 故答案为√。

(3) 平均时间复杂度为 $1+2+\dots+\log n = \theta((\log n)^2)$, 本题给定的选项中无正确选项。

(4) 在第一层递归时会执行第 17 行的 “swap” $\theta(n)$ 次, 第二层的情况与上一小题中的情况类似。故平均执行次数为 $\theta(n)$ 次, B 选项正确^[3]。

(5) 平均时间复杂度为 $n+n/2+n/4+\dots+1 = \theta(n)$; 最坏情况下每层递归长度减 1, 时间复杂度为

$n+(n-1)+(n-2)+\dots+1 = \theta(n^2)$ 。A 选项正确。

(6) 每层递归 b 从 R 扫到 $L+1$, 平均递归 $\theta(n)$ 层, 时间复杂度为 $\theta(n^2)$, D 选项正确^[1]。

3.3 程序三

```

01 #include <iostream>
02 #include <queue>
03 using namespace std;
04
05 const int maxl = 2000000000;
06
07 class Map {
08     struct item {
09         string key; int value;
10     } d[maxl];
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     }
18     static int end() { return -1; }
19     void insert(string k, int v) {
20         d[cnt].key = k; d[cnt++].value = v;
21     }
22 } s[2];
23
24
25 class Queue {
26     string q[maxl];
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             (p == 1 &&
92              (check(LtoR(st, 0, m), p, step) ||
93               check(RtoL(st, m, len - 1), p, step))))
94             return 0;
95     }
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) 若两个字符串的长度均为 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(D)。

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

(5)(4 分)已知当输入为“0123432101”时输出为 4, 当输入为“0123456432101”时输出为 14, 当输入为“012345678765432101”时输出为 28, 则当输入为“0123456789abba98765432101”输出为(D)。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(C)。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。

解析: 本题通过双向搜索求解字符串 s_0 。通过对区间 $[0, m]$ 的字符循环右移一位以及对 $[m, \text{len}-1]$ 的字符循环左移一位这两种操作生成字符串 s_1 的最小步数。需要注意的是, Map 类为题目中定义的, 复杂度与 STL 中的 map 略有不同^[2]。

(1) 可以发现当输入的两个字符串相同时输出为 0, 故答案为 $\sqrt{}$ 。

(2) 不妨举一个例子: $s_0 = \text{"aaa...aab"}, s_1 = \text{"baa...aaa"}$, 当 $m=0$ 时输出为 100, 当 $m=1$ 时输出为 1, 可以发现二者输出不同, 故答案为 \times 。

(3) Map 类的 find 函数的复杂度为 $O(\text{cnt})$, 其中 cnt 表示元素个数。故时间复杂度应为 $O((n!)^2)$, 答案为 \times 。

(4) 无论如何操作都无法做到翻转字符串, 故无解, 输出 -1, D 选项正确。

(5) 猜想结果为关于字符串长度的二次多项式。找规律可得, 对于此类输入, 输出为 $(n^2-8)/2$, 故 D 选项正确。

(6) 题目中所说的输出小于 0 仅可能是输出 -1, 即无解的情况。可以发现当 n 为奇数、 m 为偶数且 $0 < m < n-1$ 时, 若 s_0 中每个字符互不相同, 则 $LtoR(s_0, m, \text{len}-1)$ 有 $n-m-1$ 对字符的相对顺序变化, 为偶数; 而 $RtoL(s_0, 0, m)$ 有 m 对字符的相对顺序变化, 也为偶数。若 s_0 与 s_1 的相对顺序不同的字符的对数为奇数, 则 s_0 无论如何操作都无法得到 s_1 , 故 C 选项正确。

4 完善程序分析

4.1 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 $\alpha (0 < \alpha < 1)$, 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 $\alpha \cdot w$, 体积是 $\alpha \cdot v$, 另一块的价值是 $(1-\alpha) \cdot w$, 体积是 $(1-\alpha) \cdot v$ 。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。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: $1 \leq n \leq 1000, 1 \leq B \leq 10^5, 1 \leq w_i, v_i \leq 100$ 。

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
24 void swap(int &x, int &y) {
25     int t = x; x=y; y=t;
26 }
27
28 int main() {
29     scanf("%d %d", &n, &B);
30     for(int i=1; i<=n; i++){
31         scanf("%d%d", &w[i], &v[i]);
32     }
33     for(int i=1; i<n; i++)
```

```

34   for(int j=1;j<n;j++)
35       if(①) {
36           swap(w[j], w[j + 1]);
37           swap(v[j], v[j + 1]);
38       }
39   int curV, curW;
40   if(②) {
41       ③
42   } else {
43       print(B * w[1], v[1]);
44       return 0;
45   }
46
47   for(int i=2;i<=n;i++)
48       if(curV + v[i] <= B) {
49           curV += v[i];
50           curW += W[i];
51       } else {
52           print(④);
53           return 0;
54       }
55   print(⑤);
56   return 0;
57 }
58
59

```

(1) ①处应填(D)

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

(2) ②处应填(B)

- A. $w[1] <= B$
- B. $v[1] <= B$
- C. $w[1] >= B$
- D. $v[1] >= B$

(3) ③处应填(D)

- A. `print(v[1], w[1]); return 0;`
- B. `curV=0; curW=0;`
- C. `print(w[1], v[1]); return 0;`
- D. `curV = v[1]; curW = w[1];`

(4) ④处应填(D)

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

(5) ⑤处应填(B)

- A. `curW, curV`
- B. `curW, 1`
- C. `curV, curW`
- D. `curV, 1`

解析: 此处进行的是按照性价比从大到小冒泡排序的过程。

(1) 由于此处如果直接使用除法将变为整除, 故不等式两边分别乘 $v[j]*v[j+1]$, 得到 $w[j]*v[j+1] < w[j+1]*v[j]$, D 选项正确。

(2) 根据后面的内容可得出此处为判断是否能完全装下性价比最大的蛋糕, 故 B 选项正确。

(3) `curV` 表示当前选择蛋糕的总体积, `curW` 表示当前选择蛋糕的总价值。故 D 选项正确。

(4) `print(w,v)`实现的是输出分数 w/v 的既约形式。可以发现此时为剩余空间装下第 i 块蛋糕的情况, 此时剩余空间为 $B-curV$, 故最终总价值和为 $(curW*v[i]+(B-curV)*w[i])/v[i]$, D 选项正确。

(5) 此时所有蛋糕都能装下, 直接输出 `curW` 即可, B 选项正确。

4.2 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 $\frac{m}{2}$

位和后 $\frac{m}{2}$ 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 、最后一个位置的低位 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 = 10000000000000LL;
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(D)

- A. $x \geq 1$
- B. $x \neq x \& (x \wedge (x+1))$
- C. $x \neq x \mid -x$
- D. $x \neq x \& (x \wedge (x-1))$

(2) ②处应填(B)

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

(3) ③处应填(C)

- A. -INF
- B. $\text{Max}[y][x]$
- C. 0
- D. $\text{Max}[x][y]$

(4) ④处应填(A)

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

(5) ⑤处应填(B)

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

解析:

(1) $x \neq x \& (x \wedge (x-1))$ 表示将 x 在二进制表示下最低位的 1 变为 0, D 选项正确。

(2) 根据 y 的值以及后面的内容可判断 x 表示取出 a 的高 8 位, B 选项正确。

(3) 当子序列只含一个数时分值为 0, 故 v 的初值赋 0 最为妥当, C 选项正确。

(4) 根据 Max 的定义, 应在 $\text{Max}[x][z]$ 的基础上补上后 8 位的贡献, A 选项正确。

(5) 根据 Max 的定义, 应在 v 的基础上补上前 8 位的贡献, B 选项正确。

参 考 文 献

- [1] 刘汝佳. 算法竞赛入门经典. 第 2 版. 北京: 清华大学出版社, 2014
- [2] 李煜东. 算法进阶指南. 郑州: 河南电子音像出版社, 2018
- [3] 王晓东. 数据结构: C 语言描述. 第 3 版. 北京: 电子工业出版社, 2019