

## 2023 年 12 月认证 Python 七级真题解析

CCF 编程能力等级认证，英文名 Grade Examination of Software Programming (以下简称 GESP)，由中国计算机学会发起并主办，是为青少年计算机和编程学习者提供学业能力验证的平台。GESP 覆盖中小学全学段，符合条件的青少年均可参加认证。GESP 旨在提升青少年计算机和编程教育水平，推广和普及青少年计算机和编程教育。

GESP 考察语言为图形化 (Scratch) 编程、Python 编程及 C++ 编程，主要考察学生掌握相关编程知识和操作能力，熟悉编程各项基础知识和理论框架，通过设定不同等级的考试目标，让学生具备编程从简单的程序到复杂程序设计的编程能力，为后期专业化编程学习打下良好基础。

本次为大家带来的是 2023 年 12 月份 Python 七级认证真题解析。

### 一、单选题（每题 2 分，共 30 分）

题号	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
答案	B	D	B	C	D	D	B	C	B	C	D	C	B	D	B

1、假设变量 x 为 float 类型，如果下面代码输入为 100，输出最接近( )。

```
1 import math
2 def function():
3     x=float(input("Please input a:"))
4     b=math.log10(x)-math.log2(x)
5     print(b)
6 function()
```

- A.0
- B.-5
- C.-8
- D.8

【答案】B

【解析】本题属于考察用 python 解决数学问题，使用了一个数学上对数方式来计算，计算 10 为底的对数 5 减以 2 为底的对数 5, 答案为: -5。

2、对于下面动态规划方法实现的函数，以下选项中最适合表达其状态转移函数的为( )。

```

1 #函数调用前，已定义s为list成员数量为MAX_N，
2 #f同为list，成员数量为MAX_N，每个成员为MAX_N个成员list
3 def stone_merge(n,a):
4     for i in range(1,n+1):
5         s[i] = s[i - 1] + a[i]
6     for i in range(1,n+1):
7         for j in range(1,n+1):
8             if i == j:
9                 f[i][j] = 0
10            else:
11                f[i][j] = MAX_F #预定义的值
12    for l in range(1, n):
13        for i in range(1, n - l + 1):
14            j = i + l
15            for k in range(i,j):
16                f[i][j] = min([f[i][j],f[i][k] + f[k + 1][j] + s[j] - s[i-1]])
17    return f[1][n]

```

- A.  $f(i, j) = \min_{i \leq k < j} (f(i, j), f(i, k) + f(k + 1, j) + s(j) - s(i - 1))$
- B.  $f(i, j) = \min_{i \leq k < j} (f(i, j), f(i, k) + f(k + 1, j) + \sum_{k=i}^j a(k))$
- C.  $f(i, j) = \min_{i \leq k \leq j} (f(i, k) + f(k + 1, j) + \sum_{k=i}^{j+1} a(k))$
- D.  $f(i, j) = \min_{i \leq k < j} (f(i, k) + f(k + 1, j)) + \sum_{k=i}^j a(k)$

【答案】D

【解析】本题属于考察区间动态规划状态转移表达式的题目，根据程序，我们可以抓住程序的终点，可以知道，在本区间中，i 和 j 是两个端点值，k 是分界点，所以 k 的范围是在 [i,j], 所以 C 中累加到 j+1 错误。通过 16 行状态转移方程得知，s 列表和 k 无关，所以将其单独提出来，所以选择 D。

3、下面代码可以用来求最长上升子序列 (LIS) 的长度，如果输入是整数序列是：5 1 7 3 5 9，则输出是( )。

```
arr = list(map(int,input().split(",")))
cnt = 1
ans = 0

for i in range(len(arr)-1):
    if arr[i] < arr[i+1]:
        cnt += 1
    else:
        ans = max(ans,cnt)
        cnt = 1
    ans = max(ans,cnt)
print(ans)
```

- A. 9 7 5 1 1 9
- B. 1 2 2 2 3
- C. 1 3 5 7 9 9
- D. 1 1 1 1 1 1

**【答案】 B**

**【解析】** 本题属于考察最长上升子序列的问题，要寻找 arr 中最长上升子序列，要明白，子序列未必是连续的。其中 cnt 是用来记录连续上升元素的个数，如果中途被打断，则 cnt 从 1 开始重新计算。而 ans 则是用来记录列表的最长子序列的个数。将数字 5,1,7,3,5,9 分别带入，ans 值分别为 1 2 2 2 3，选择 B。

4、Python 中，下列关于类描述不正确的是( )。

- A. 类属性可以通过类的名称访问其值。
- B. 类属性也可以称之为类的静态属性。
- C. 和实例属性相同，类属性前面也必须有 self 关键字，并以句点间隔。
- D. 如果在类的某个实例（对象）中修改类属性的值，则其他该类实例访问该值时，其值也随之改变。

**【答案】 C**

**【解析】** 本题属于考察类的知识点，类属性是指在类中函数之外的属性，不需要前面有 self 关键字，所以选择 C。

5、G 是一个非连通无向图，共有 28 条边，则该图至少有( )个顶点。

- A. 6
- B. 7
- C. 8
- D. 9

**【答案】D**

**【解析】**本题属于考察非连通无向图，问至少有几个顶点，所以最多会有一个点是独立的点，我们可以看成有 28 条边的无向完全图有多少顶点再额外+1 点，根据公式  $n(n-1)/2=28$ ，得出 n 为 8，所以该非连通无向图至少为 9 个顶点，选择 D。

6、哈希表长 31，按照下面的程序依次输入 4 17 28 30 4，则 4 存入哪个位置？（ ）

```
1 N = 31
2 htab, flag = [0 for i in range(N)], [0 for i in range(N)]
3
4 n, x, i, j, k = 0, 0, 0, 0, 0
5
6 n = int(input())
7 for i in range(n):
8     x = int(input())
9     k = x % 13
10
11     while(flag[k]):
12         k = (k+1) % 13
13
14     htab[k] = x
15     flag[k] = 1
16
17 for i in range(N):
18     print(htab[i], end=" ")
```

- A. 3
- B. 4
- C. 5
- D. 6

**【答案】D**

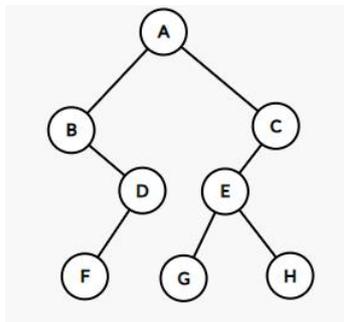
**【解析】**本题属于考察哈希表，解决哈希冲突的问题，根据题意我们发现，是按照%13 进行哈希，并且在发生冲突的时候，放到下一个位置，我们依次将 4 17 28 30 4 带入，得出 17 在 4 号，28 在 2 号，30 本应在 4 号但是被占用顺位到 5 号，4 在 4 号，发生冲突后顺位到 6 号，选择 D。

7、某二叉树 T 的先序遍历序列为：{A B D F C E G H}，中序遍历序列为：{B F D A G E H C}，则下列说法中正确的是( )。

- A. T 的度为 1
- B. T 的高为 4
- C. T 有 4 个叶节点
- D. 以上说法都不对

**【答案】 B**

**【解析】**本题属于考察二叉树的遍历问题，已知先序和中序，可得到如图二叉树：



选择 B。

8、下面代码段可以求两个字符串 s1 和 s2 的最长公共子串（LCS），下列相关描述不正确的是( )。

```
1 def LCS(str1,str2):
2     res = [[0]*len(str2) for _ in range(len(str1))]
3     max_len=0
4     max_str=[]
5     for i in range(len(str1)):
6         for j in range(len(str2)):
7             if str1[i]==str2[j]:
8                 if i>0 and j>0:
9                     res[i][j]=1+res[i-1][j-1]
10                else:
11                    res[i][j]=1
12                if res[i][j]>max_len:
13                    max_len=res[i][j]
14                    max_str=[str1[i-max_len+1:i+1]]
15                elif res[i][j]==max_len:
16                    max_str.append(str1[i-max_len:i])
17        return max_str
18 str1,str2=(input().split())
19 n=LCS(str1,str2)
```

- A. 代码的时间复杂度为  $O(n^2)$
- B. 代码的空间复杂度为  $O(n^2)$
- C. 空间复杂度已经最优
- D. 采用了动态规划求解

**【答案】 C**

**【解析】** 本题属于考察最长公共子串 LCS，根据程序，我们发现，代码中使用了双重 for 循环，所以时间复杂度为  $O(n^2)$ ，A 为正确。代码中使用了二维列表 res，所以空间复杂度为  $O(n^2)$ ，B 为正确。代码使用了动态规划的算法。其中 C 选项，我们可以将二维列表优化为一维列表，每次向下滚动后，原来的一维列表内容就不再使用，所以可以使用更新的方式，不断的更新一维列表，是空间复杂度降为  $O(n)$ ，选择 C。

9、图的广度优先搜索中既要维护一个标志数组标志已访问的图的结点，还需哪种结构存放结点以实现遍历？ ( )

- A. 双向栈
- B. 队列
- C. 哈希表
- D. 堆

**【答案】 B**

**【解析】** 本题属于考察广度优先搜索算法的题，根据之前所学，使用队列的先进先出的原则，来遍历所有的节点，选择 B。

10、对关键字序列{44, 36, 23, 35, 52, 73, 90, 58}建立哈希表，哈希函数为  $h(k)=k\%7$ ，执行下面的 Insert 函数，则等概率情况下的平均成功查找长度（即查找成功时的关键字比较次数的均值）为( )。

```
1 class Node:
2     def __init__(self,Data=None,Next=None):
3         self.Data = Data
4         self.Next = Next
5 hTab = [Node() for i in range(7)]
6 Key = [44, 36, 23,35, 52, 73, 90, 58, 0]
7
8 def Insert():
9     i, j = 0, 0
10    x = Node()
11    while Key[i]:
12        j = Key[i] % 7
13        x = Node();
14        x.Data = Key[i]
15        x.Next = hTab[j]
16        hTab[j] = x
17
18    i += 1
19
20 Insert()
```

- A. 7/8
- B. 1
- C. 1.5
- D. 2

**【答案】 C**

**【解析】** 本题属于考察哈希表知识点，其中哈希函数为%7，代码中使用了类来模拟了单向链表，将所有哈希地址相同的记录都放在了同一个链表中。我们将值44 36 23 35 52 73 90 58依次带入，得到的哈希地址分别为2 1 2 0 3 3 6 2，其中0~6的元素为1 1 3 2 0 0 1，这8个数字查找成功的次数分别是1,1,2,1,1,2,1,3，总数为12次,12/8=1.5,选择C。

11、学生在读期间所上的某些课程中需要先上其他的课程，所有课程和课程间的先修关系构成一个有向图G，有向边<U, V>表示课程U是课程V的先修课，则找到某门课程C的全部先修课下面哪种方法不可行？( )

- A. BFS 搜索
- B. DFS 搜索
- C. DFS+BFS
- D.动态规划

**【答案】 D**

【解析】本题属于考察有向图的搜索，即寻找有向图中有多少个点能到达指定点，所以 BFS,DFS,DFS+BFS 都可以，选择 D。

12、一棵完全二叉树有 2023 个结点，则叶结点有多少个？( )

- A. 1024
- B. 1013
- C. 1012
- D. 1011

【答案】C

【解析】本题属于考察二叉树叶结点的问题，通过公式，在总结点为奇数时，带入 $(n+1)/2$ 可得 1012，选择 C。

13、用下面的邻接表结构保存一个有向图 G，InfoType 和 VertexType 是定义好的类。设 G 有 n 个顶点、e 条弧，则求图 G 中某个顶点 u（其顶点序号为 k）的度的算法复杂度是( )。

```
1 class ArcNode:
2     def __init__(self,adjvex=None,nextarc=None,info=None):
3         self.adjvex = adjvex
4         self.nextarc = nextarc
5         self.info = info
6
7 class VNode:
8     def __init__(self,data=None,firstarc=None):
9         self.data = data
10        self.firstarc = firstarc
11
12 class ALGraph:
13     def __init__(self,vertices,vexnum,arcnum,kind):
14         self.vertices = vertices
15         self.vexnum, self.arcnum = vexnum, arcnum
16         self.kind = kind
17
18 AdjList = [VNode() for i in range(MAX_VERTEX_NUM)]
```

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

【答案】B

**【解析】**本题中使用了邻接表来存储有向图边的信息，在查找某点的度时，有两种方法：

需要考虑出度和入度，在考虑出度时候，直接遍历该点的出边即可，入度则在反图中进行遍历即可。

直接在邻接表中找到所有关于此点的弧的数目，就是此点的度。

两种方法总复杂度均为  $O(e)$ ，所以选择 B。

14、给定一个简单有向图  $G$ ，判断其中是否存在环路的下列说法哪个最准确？（ ）

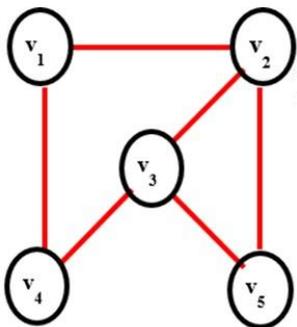
- A. BFS 更快
- B. DFS 更快
- C. BFS 和 DFS 一样快
- D. 不确定

**【答案】** D

**【解析】**本题属于考察搜索算法速率的题目，BFS 和 DFS 搜索的效率在不同的图中是不一样的，所以没法确定，选择 D。

15、从顶点  $v_1$  开始遍历下图  $G$  得到顶点访问序列，在下面所给的 4 个序列中符合广度优先的序列有几个？（ ）

$\{v_1 v_2 v_3 v_4 v_5\}$  ,  $\{v_1 v_2 v_4 v_3 v_5\}$  ,  $\{v_1 v_4 v_2 v_3 v_5\}$  ,  $\{v_1 v_2 v_4 v_5 v_3\}$



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

**【答案】** B

【解析】本题属于考察广度优先搜索题目，广度优先是先将与该结点相连的点遍历一次，上面4个都是从V1开始搜索，和V1相连接的只有V2,V4，所以V1之后应该是V2,V4其顺序没有关系，将第一个排除，和V2相连接的有V3,V5,和V4相连接的只有V3。所以后三个完全符合广度优先搜索，选择B。

## 二、判断题（每题2分，共20分）

题号	1	2	3	4	5	6	7	8	9	10
答案	√	√	×	√	√	×	×	×	√	×

1、小杨这学期准备参加GESP的7级考试，其中有关于三角函数的内容，他能够通过下面的代码找到结束循环的角度值。（ ）

```

1  import math
2  def function():
3      while True:
4          x=float(input())
5          x=x/180*3.14
6          if int(math.sin(x)*math.sin(x)+math.cos(x)*math.cos(x))==1:
7              print(x)
8              break
9
10 function()

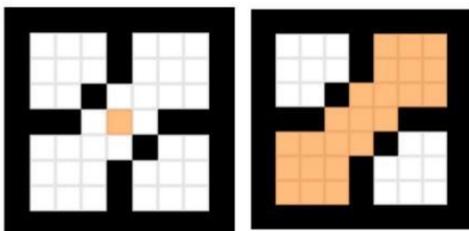
```

【答案】正确

【解析】本题是一道数学题，其中第5行是将角度转换为弧度，通过数学定理  $\sin^2x+\cos^2x=1$ ，来找到合适的角度，所以正确。

2、小杨在开发画笔刷小程序（applet），操作之一是选中黄颜色，然后在下面的左图的中间区域双击后，就变成了右图。这个操作可以用图的泛洪算法来实现。

（ ）



【答案】正确

【解析】泛洪算法的本质是通过一个点向周边相邻区域进行扩散，符合题目要求，所以正确。

3、假设一棵完全二叉树共有  $N$  个节点，则树的深度为  $\log(N)+1$ 。( )

【答案】错误

【解析】计算完全二叉树的公式为  $\text{floor}(\log_2 N)+1$ ，和题意不符，错误。

4、给定一个数字序列  $A_1, A_2, A_3, \dots, A_n$ ，要求  $i$  和  $j$  ( $1 \leq i \leq j \leq n$ )，使  $A_i + \dots + A_j$  最大，可以使用动态规划方法来求解。( )

【答案】正确

【解析】本题是考察求最大字段和，是动态规划的经典题目，正确。

5、若变量  $x$  为 float 类型正数，则  $\log(\text{math.exp}(x)) > \text{math.log10}(x)$ 。( )

【答案】正确

【解析】题目中  $\text{math.exp}(x)$  为  $e^x$ ，所以  $\log e^x$  结果为  $x$ ，根据  $\log_{10} x$  函数图像得知，只有在  $0 < x < 1$  时， $\log_{10} x$  为负数，在  $x=1$  时， $\log_{10} x$  为 0，且函数  $\log_{10} x$  的斜率小于函数  $x$  的斜率，所以  $x > \log_{10} x$ ，符合题意，正确。

6、简单有向图有  $n$  个顶点和  $e$  条弧，可以用邻接矩阵或邻接表来存储，二者求节点  $u$  的度的时间复杂度一样。( )

【答案】错误

【解析】根据题意，求结点  $u$  邻接矩阵的时间复杂度为  $O(n)$ ，邻接表时间复杂度为  $O(e)$ ，不符合题意，错误。

7、某个哈希表键值  $x$  为整数，为其定义哈希函数  $H(x)=x\%p$ ，则  $p$  选择素数时不会产生冲突。( )

【答案】错误

【解析】根据题意，如果两个整数  $x$  分别为  $p$  的倍数，则会产生哈希冲突，不符合题意，错误。

8、动态规划只要推导出状态转移方程，就可以写出递归程序来求出最优解。( )

【答案】错误

【解析】动态规划算法的核心是状态转移方程，但是也有状态，初始条件和边界条件等要求，所以不符合题意，错误。

9、广度优先搜索（BFS）能够判断图是否连通。( )

【答案】正确

【解析】在图论中遍历图的算法包含 BFS，可以通过任意一点找到图中所有点，将不同点的个数相加和总数做对比，则能判断是否为连通图。正确。

10、在 Python 中，如果定义了构造函数，则创建对象时先执行完缺省的构造函数，再执行这个定义的构造函数。( )

【答案】错误

【解析】一个类中可以 1 个或多个构造函数，但是只会默认执行最后一个构造函数。不符合题意，错误。

三、编程题（每题 25 分，共 50 分）

题号	1	2
答案		

1、闯关游戏

**问题描述**

市场上共有  $N$  种商品，编号从  $0$  至  $N-1$ ，其中，第  $i$  种商品价值  $v_i$  元。

现在共有  $M$  个商人，编号从  $0$  至  $M-1$ 。在第  $j$  个商人这，你可以使用第  $x_j$  种商品交换第  $y_j$  种商品。每个商人都会按照商品价值进行交易，具体来说，如果

$v_{x_j} > v_{y_j}$ ，他将会付给你  $v_{y_j} - v_{x_j}$  元钱；否则，那么你需要付给商人  $v_{x_j} - v_{y_j}$  元钱。

除此之外，每次交易商人还会收取 1 元作为手续费，不论交易商品的价值孰高孰低。

你现在拥有商品 a，并希望通过一些交换来获得商品 b。请问你至少要花费多少钱？（当然，这个最小花费也可能是负数，这表示你可以在完成目标的同时赚取一些钱。）

### 输入描述

第一行四个整数  $N, M, a, b$ ，分别表示商品的数量、商人的数量、你持有的商品以及你希望获得的商品。保证  $0 \leq a, b < N$ 、保证  $a \neq b$ 。

第二行  $N$  个用单个空格隔开的正整数  $v_0, v_1, \dots, v_{N-1}$ ，依次表示每种商品的价值。保证  $1 \leq v_i < 10^9$ 。

接下来  $M$  行，每行两个整数  $x_j, y_j$ ，表示第  $j$  个商人愿意使用第  $x_j$  种商品交换第  $y_j$  种商品。保证  $0 \leq x_j, y_j < N$  保证  $x_j \neq y_j$ 。

### 输出描述

输出一行一个整数，表示最少的花费。特别地，如果无法通过交换换取商品，请输出 No solution

### 特别提醒

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

### 样例输入 1

```
1 | 3 5 0 2
2 | 1 2 4
3 | 1 0
4 | 2 0
5 | 0 1
6 | 2 1
7 | 1 2
```

### 样例输出 1

```
1 | 5
```

### 样例解释 1

可以先找 2 号商人，花  $2-1=1$  元的差价以及 1 元手续费换得商品 1，再找 4 号商人，花  $4-2=2$  元的差价以及 1 元手续费换得商品 2。总计花费  $1+1+2+1=5$  元。

## 样例输入 2

```
1 | 3 3 0 2
2 | 100 2 4
3 | 0 1
4 | 1 2
5 | 0 2
```

## 样例输出 2

```
1 | -95
```

## 样例解释 2

可以找 2 号商人，直接换得商品 2 的同时，赚取  $100 - 4 = 96$  元差价，再支付 1 元手续费，净赚 95 元。

也可以先找 0 号商人换取商品 1，再找 1 号商人换取商品 2，不过这样只能赚 94 元。

## 样例输入 3

```
1 | 4 4 3 0
2 | 1 2 3 4
3 | 1 0
4 | 0 1
5 | 3 2
6 | 2 3
```

## 样例输出 3

```
1 | No solution
```

## 数据规模

对于 30% 的测试点，保证  $N \leq 10$ ， $M \leq 20$ 。

对于 70% 的测试点，保证  $N \leq 10^3$ ， $M \leq 10^4$ 。

对于 100% 的测试点，保证  $N \leq 10^5$ ， $M \leq 2 \times 10^5$ 。

**【解题思路】** 本题属于最短路问题且权值为 1，则可以使用 BFS 来求解最短路。该算法从初始商品开始，探索所有可能的交易组合，计算到达目标商品的最低成本。在这个过程中，维护一个队列来追踪待探索的交易路径，并记录已探索商品的最低成本，避免重复计算。该问题的关键在于合理设置数据结构来存储商品、商人和交易的信息，并高效实现 BFS 算法来找到最优解。

其中最低成本的计算：在 BFS 的过程中，记录到达每种商品的当前最低成本。如果找到一条新的交易路径，其总成本比当前记录的成本低，则更新成本。如果能够通过交易到达目标商品，并且没有其他更低成本的路径，算法结束并输出最低成本。如果无法通过交易到达目标商品，则输出“无解”。

此题目的关键是最小化交易次数，因为不同商品间的价值差是固定的。通过将每件商品视为一个点，如果商品  $x$  可以换成商品  $y$ ，那么在  $x$  和  $y$  之间连一条边。

目标是从持有的商品  $a$  尽快到达希望获得的商品  $b$ ，即最少经过的边数。这可以通过广度优先搜索（BFS）来实现。求出的最少边数记为  $\text{min\_dist}[\text{dst}]$ 。最终答案是  $\text{min\_dist}[\text{dst}] + v[\text{b}] - v[\text{a}]$ ，其中  $v[\text{b}]$  和  $v[\text{a}]$  分别是商品  $b$  和  $a$  的价值。

### 【参考程序】

```
from collections import deque
max_n = int(1e5) + 10
n = 0
edge = [[] for _ in range(max_n)]
val = [0] * max_n
min_dist = [float('inf')] * max_n
queue = deque()
def bfs(src):
    global min_dist
    queue.clear()
    min_dist = [float('inf')] * max_n
    1 -95
    1 No solution queue.append(src)
    min_dist[src] = 0
    while queue:
        u = queue.popleft()
        for v in edge[u]:
            if min_dist[u] + 1 < min_dist[v]:
                min_dist[v] = min_dist[u] + 1
                queue.append(v)
    if __name__ == "__main__":
        n, m, src, dst = map(int, input().split())
        n_values = list(map(int, input().split()))
        for i in range(n):
            val[i] = n_values[i]
        for _ in range(m):
            x, y = map(int, input().split())
            edge[x].append(y)
        bfs(src)
        if min_dist[dst] > n:
            print("No solution")
        else:
            print(min_dist[dst] - val[src] + val[dst])
```

## 2、工作沟通

### 问题描述

你和小杨在玩一个纸牌游戏。

你和小杨各有 3 张牌，分别是 0、1、2。你们要进行  $N$  轮游戏，每轮游戏双方都要出一张牌，并按 1 战胜 0，2 战胜 1，0 战胜 2 的规则决出胜负。第  $i$  轮的胜者可以获得  $2a_i$  分，败者不得分，如果双方出牌相同，则算平局，二人都可获得  $a_i$  分 ( $i=1,2,\dots,N$ )。

玩了一会后，你们觉得这样太过于单调，于是双方给自己制定了不同的新规则。小杨会在整局游戏开始前确定自己全部  $n$  轮的出牌，并将他的全部计划告诉你：而你从第 2 轮开始，要么继续出上一轮出的牌，要么记一次“换牌”。游戏结束时，你换了  $t$  次牌，就要额外扣  $b_1 + \dots + b_t$  分。

请计算出你最多能获得多少分。

### 输入描述

第一行一个整数  $N$ ，表示游戏轮数。

第二行  $N$  个用单个空格隔开的非负整数  $a_1, \dots, a_N$ ，意义见题目描述。

第三行  $N-1$  个用单个空格隔开的非负整数  $b_1, \dots, b_{N-1}$ ，表示换牌的罚分，具体含义见题目描述。由于游戏进行  $N$  轮，所以你至多可以换  $N-1$  次牌。

第四行  $N$  个用单个空格隔开的整数  $c_1, \dots, c_N$ ，依次表示小杨从第 1 轮至第  $N$  轮出的牌。保证  $c_i \in \{0,1,2\}$ 。

### 输出描述

一行一个整数，表示你最多获得的分数。

### 特别提醒

在常规程序中，输入、输出时提供提示是好习惯。但在本场考试中，由于系统限定，请不要在输入、输出中附带任何提示信息。

### 样例输入 1

```
1 4
2 1 2 10 100
3 1 100 1
4 1 1 2 0
```

### 样例输出 1

1 | 219

### 样例解释

你可以第 1 轮出 0，并在第 2,3 轮保持不变，如此输掉第 1,2 轮，但在第 3 轮中取胜，获得  $2 \times 10 = 20$  分；随后，你可以在第 4 轮中以扣 1 分为代价改出 1，并在第 4 轮中取得胜利，获得  $2 \times 100 = 200$  分。如此，你可以获得最高的总分  $20 + 200 - 1 = 219$ 。

### 样例输入 2

```
1 | 6
2 | 3 7 2 8 9 4
3 | 1 3 9 27 81
4 | 0 1 2 1 2 0
```

### 样例输出 2

1 | 56

### 数据规模

对于 30% 的测试点，保证  $N \leq 15$ 。

对于 60% 的测试点，保证  $N \leq 100$ 。

对于所有测试点，保证  $N \leq 1,000$ ；保证  $0 \leq a_i, b_i \leq 10^6$ 。

**【解题思路】** 本题基于动态规划（DP）算法。我们定义  $dp[i][j][k]$  表示前  $i$  轮中，第  $i$  轮出牌为  $j$  ( $0 \leq j \leq 2$ )，且已经换过  $k$  次牌的最大得分。状态转移方程如下：

$$dp[i][j][k] = \max(dp[i-1][j][k] + \text{result}(j, c[i]) * a[i], dp[i-1][j'][k-1] + \text{result}(j, c[i]) * a[i] - b[k])$$
 其中  $j' \neq j$ 

这里  $a$  数组表示每轮的奖励得分， $b$  数组是换牌的惩罚， $c$  数组记录小杨每轮的出牌。 $\text{result}(x, y)$  函数判断出牌为  $x$  时与  $y$  的胜负情况（胜利返回 2，平局返回 1，失败返回 0）。

第一部分的状态转移表示当前轮出的牌与上一轮相同，不需要额外的换牌成本。

第二部分表示当前轮出的牌与上一轮不同，需要支付换牌的代价  $b[k]$ 。

最终答案是  $\max(dp[n][j][k])$ ，其中  $j$  的范围是 0 到 2， $k$  的范围是 0 到  $n-1$ 。

通过这种方式，我们可以有效地计算在不同出牌选择和换牌次数下的最大得分，并找出获得最高总得分的策略。

### 【参考程序】

```
max_n = 1005

n = int(input())
a = [0] * max_n
b = [0] * max_n
c = [0] * max_n

dp = [[0] * max_n for _ in range(3)]

def result(x, y):
    if x == y + 1 or x == y - 2:
        return 2
    if x == y:
        return 1
    return 0

a[1:] = list(map(int, input().split()))
b[1:] = list(map(int, input().split()))
c[1:] = list(map(int, input().split()))

for k in range(3):
    dp[k][0] = result(k, c[1]) * a[1]

for i in range(2, n + 1):
    for j in range(i - 1, -1, -1):
        for k in range(3):
            curr_score = result(k, c[i]) * a[i]
            dp[k][j] += curr_score
            if j > 0:
                for l in range(3):
                    dp[k][j] = max(dp[k][j], dp[l][j - 1] + curr_score - b[j])

ans = -2e9
x = 0
for j in range(n):
    for k in range(3):
        ans = max(ans, dp[k][j])

print(ans)
```



**CCF-GESP编程能力等级认证**  
Grade Examination of Software Programming

**GESP 2024年3月认证**  
认证语言: C++/Python/图形化编程  
报名时间: 2024年1月18日至3月5日24点截止  
缴费时间: 2024年1月18日至3月5日24点截止  
认证时间: 1-4级 2024年3月16日 上午 09:30-11:30  
5-8级 2024年3月16日 下午 13:30-16:30  
认证方式: 全国统一线下机考

扫码即刻报名

GESP面向全国征集授权服务中心和考点 (考点仅限公立校)  
欢迎申请, 扫码至官网了解更多

### 【联系我们】

1. GESP 微信: 关注“CCF GESP”公众号, 将问题以文字方式留言即可得到回复。

2. GESP 邮箱: [gesp@ccf.org.cn](mailto:gesp@ccf.org.cn)

注: 请在邮件中详细描述咨询的问题并留下考生的联系方式及姓名、身份证号, 以便及时有效处理。

3. GESP 电话: 0512-67656856

咨询时间: 周一至周五(法定节假日除外): 上午 8:30-12:00; 下午 13:00-17:30

GESP 第五期认证报名已启动, 扫描下方二维码, 关注 GESP 公众号即可报名

