模拟练不停

2023

信息技术

1. 算法练不停
2. **算法练不停(精简版)**
3. 基础练不停
4. 小题练不停

浙江金榜苑国际教育中心

模拟练不停 2023

信息技术

算法练不停(精简版)

小梅哥网络教育中心

配套资料下载地址：
https://github.com/yemaster/EndlessPractice/tree/master/Algorithm

目录 Contents

模拟小卷 1 双指针

试题卷

一、非选择题(共 3 小题，每题 10 分，共 30 分)

1. 给定 n 个数 $a_1, a_2, \cdots, a_n$ 和正整数 C，求有多少个数对 i, j $1 \le i, j \le n$ 满足 $a_i - a_j = C$.

算法思路：先将全部的数从小到大排序，合并相同的数；然后对于每一个 i，统计符合要求的 j。由于 i 单调递增，所以 j 不必从 1 开始，而是从上一次最小的不符合条件的 j 开始即可。

```
cnt = [1] * 100000  # 统计每个数字出现的次数
n, c = list(map(int, input().split()))
nums = list(map(int, input().split()))
for i in range(n - 1):
    for j in ___1___:
        if nums[j] < nums[j - 1]:
            nums[j], nums[j - 1] = nums[j - 1], nums[j]
top = 0
for i in range(1, n):
    if nums[i] == nums[i - 1]:
        cnt[top] += 1
    else:
        top += 1
        ___2___
i, j = top - 1, top
ans = 0
while j > 0:
    while ___3___:
        i -= 1
    if nums[j] - nums[i] == c:
        ans += ___4___
    j -= 1
```

```
print(ans)
```

(1) n 个数分别为 1,1,2,3，$C=1$，则答案为_____。

(2) 请补全空缺处的代码。

(3) 程序的时间复杂度为_____。

2. 博览馆正在展出由世上最佳的 m 位画家所画的图画。游客在购买门票时必须说明两个数字，a 和 b，代表他要看展览中的第 a 幅至第 b 幅画（包含 a,b）之间的所有图画，而门票的价钱就是一张图画一元。小 R 希望入场后可以看到**所有画家**的图画。当然，他想**最小化**购买门票的价格。例如，有一共有 3 位画家(编号 1~3)，7 张画，7 张画对应画家编号为 1,1,2,1,1,1,3，则答案为 $a=3,b=7$。

作为小 R 的好朋友，你编写了以下 Python 程序用于求出价格最少的购票方案 a,b。

```
n, m = list(map(int, input().split()))#n 表示作品数，m 表示画家数
artist = list(map(int, input().split()))
cnt = [0] * 2005
num = 0
l, r = 0, -1
ansa, ansb = -1, 123456789
while    1    :
    r += 1 #选取区间右端点向右伸展
    if (cnt[artist[r]] == 0):
        num += 1
    cnt[artist[r]] += 1
    while cnt[artist[l]] > 1:
        cnt[artist[l]] -= 1
        l += 1
    if num == m:
        if r - l < ansb - ansa:
            ansa, ansb = l, r
        cnt[artist[l]] -= 1
        if (    3    ):
            num -= 1
        l += 1
```

```
print(ansa + 1, ansb + 1)
```

(1) 一共有 5 位画家，12 张画，12 张画的作者分别为 2,5,3,1,3,2,4,1,1,5,4,3，则答案为_____。

(2) 请补全空缺处的代码。

(3) 程序的时间复杂度为_____。

3. 小 R 买了 n 本书，编号 $1 \sim n$，把它们排成了一排。不幸的是，淘气的小 B 把它们打乱了，小 R 不得不花很大的力气把它们重新排号。已知第 i 本书的编号为 a_i，每次小 R 可以选择任意一段区间 $[l,r]$，然后花 $r-l+1$ 的力气把 $a_l, a_{l+1}, \cdots, a_r$ 从小到大排序。

不想花很大的力气在这上面，小 R 想知道把所有 n 本书放回原来的位置，也就是 $a_i = i$ 恒成立，至少需要花费多少力气。

例如：a 为 [1,3,2]，则可以选择区间 [2,3]，花费力气 2。

(1) 若 a 为 [2,1,7,6,3,4,5]，则至少需要花费力气_____。

(2) 作为小 R 的好朋友，你编写了如下 Python 程序帮助小 R 计算。程序思路如下：从左往右枚举 l，找到最小的 r，使得 $[l,r]$ 中的值恰好都在 $[l,r]$ 中，将 $[l,r]$ 排好序，这样就可以把 $[1,r]$ 排好序了。以此处理整个区间。请补全空缺处的代码。

```
n = int(input())
a = list(map(int, input().split()))
l = 0
ans = 0
while l < n:
    if    1    :
        l += 1
    else:
        mx = a[l]
        r = l
        while mx > r + 1:
            r += 1
           2
        ans += r - l + 1
    3
print(ans)
```

模拟小卷 1 双指针

答题卷

姓名：_____

一、非选择题(共 3 小题，每题 10 分，共 30 分)

1. (1) _____ (1 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
   第四空：_____ (2 分)
   (3) _____ (1 分)
2. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
   (3) _____ (2 分)
3. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (3 分)
   第三空：_____ (3 分)

模拟小卷 2 二分

试题卷

一、**非选择题**(共 4 小题，第 1 题 8 分，第 2 题 7 分，第 3 题 9 分，第 4 题 6 分，共 30 分)

1. D 老师是学校的教室管理员，同学们经常借教室举办活动或者自己自习。接下来的 n 天中，第 i 天学校有 r_i 个教室可以借(我们认为这些教室没有差别)，共有 m 份订单，编号 $1 \sim m$，每份订单用三个正整数描述，分别为 d_j, s_j, t_j，表示某租借者需要从第 s_j 天到第 t_j 天租借教室（包括第 s_j 天和第 t_j 天），每天需要租借 d_j 个教室。

   借教室的原则是先到先得。现在 D 老师想知道能不能满足这些订单，如果不能，第一个不能满足的订单编号是多少。

   由于数据量庞大，聪明的小 R 打算用 Python 帮助 D 老师处理信息。

   算法思路：二分第一个不能满足的订单编号，然后对其进行判断。设 $need_i$ 为第 i 天需要的教室个数，对于第 i 个订单，把 $need_{s_i} \sim need_{t_i}$ 全部加 1 即可。此时时间复杂度较高，可以考虑用设 $diff_i = need_i - need_{i-1}$，那么修改时只要修改 $diff_{s_i}$ 和 $diff_{t_i+1}$ 两个值即可，最后再用 $diff$ 还原 $need$ 即可。

```
d, s, t = [0], [0], [0]
n, m = map(int, input().split())
rest = list(map(int, input().split()))
for i in range(m):
    tmp = list(map(int, input().split()))
    d.append(tmp[0]); s.append(tmp[1]); t.append(tmp[2])
def isok(x): # 判断前 x 个订单能否满足需求
    diff = [0] * (n + 10)
    need = [0] * (n + 10)
    for i in range(1, x + 1):
        diff[s[i]] += d[i]
        ___1___
    for i in range(1, n + 1):
```

```
            ___2___
            if need[i] > rest[i - 1]:
                return False
        return True
    l, r = 1, m
    if ___3___:
        print("能完成")
    else:
        while l < r:
            mid = (l + r) // 2
            if isok(mid):
                ___4___
            else:
                r = mid
        print("不能完成")
        print(___5___) # 输出第一个不能完成的订单编号
```

请补全空缺处代码。

2. 小 R 最近爱上了养獾。她有 n 只獾，每天需要很多食物来饲养他们，每只獾需要的食物为 $h_1, h_2, \cdots, h_n$。而且獾比较贪婪，如果一只獾存在，他就多需要额外的食物。每只獾额外需要的 1 份食物为 $g_1, g_2, \cdots, g_n$。

例如，小 R 养了 3 只獾，那么每只獾除了自己需要的食物 h_i，还要额外的两份食物 $2 \times g_i$，这样子每只獾需要食物 $h_i + 2 \times g_i$。

因此，养的越多，食物需要也越多，但小 R 现在每天能提供的食物有限，只有 f，所以不得不将一部分獾送人，只留下一些来养。作为小 R 的好朋友，小 B 编写了如下 Python 代码：

```
n = int(input())
h = [0] + list(map(int, input().split())) # 读取 h[i]
g = [0] + list(map(int, input().split()))  # 读取 g[i]
f = int(input())
l, r = 1, n
ans = -222222
def judge(m):
    z = [0] + [___1___ for i in range(1, n + 1)]
```

```
    for i in range(1, n):
        for j in range(i, n + 1):
            if z[i] > z[j]:
                z[i], z[j] = z[j], z[i]
    s = ___2___
    if s <= f:
        return True
    else:
        return False
while l <= r:
    m = (l + r) // 2
    if judge(m):
        ___3___
        if m > ans:
            ans = m
    else:
        r = m - 1
print(ans)
```

(1) 下列说法正确的是_____(单选)。

A. 程序使用了冒泡排序算法。

B. 程序的时间复杂度为 $\mathcal{O}\ m^2 \log n$ 。

C. 若把 ans 的初值设为 0，答案不变。

(2) 请补全空缺处的代码。

3. 疫情期间，同学们一起排队做核酸。一共有 n 排队伍，由于操作速度不同，每排队伍的通行速度也不一样，平均测量，第 k 排队伍的速度为 T_k 秒/人。现在，有 m 个同学需要做核酸，最少需要多少时间?

```
n, m = map(int, input().split())
T = list(map(int, input().split()))
for i in range(n - 1):
    for j in ___1___:
        if T[j] < T[j - 1]:
            T[j], T[j - 1] = T[j - 1], T[j]
def check(t):
```

```
    num = 0
    for i in T:
        num += ____2____
        if num >= m:
            return True
    return False
L, R = 1, T[-1] * m
ans = R
while ____3____:
    mid = (L + R) // 2
    if check(mid):
        ans = ____4____
        R = mid - 1
    else:
        L = mid + 1
print(ans)
```

(1) 若一共有 7 个队伍，处理速度分别为 3,8,3,6,9,2,4，则做 10 个人至少需要_____分钟。

(2) 请补全空缺处的代码。

4. 有 n 个数 $a_0, a_1, \ldots, a_{n-1}$，满足 $a_0 + a_1 - a_2, a_1 + a_2 - a_3, \ldots, a_{n-3} + a_{n-2} - a_{n-1}$ 先增后减，小 R 利用二分算法编写了以下程序，用于找出最大的 $a_i + a_{i+1} - a_{i+2}$ 并输出它的值，试补全划线处的程序：

```
a = list(map(int, input().split()))
n = len(a)
____1____
r = n - 3
while l - r - 1 <= r:
    mid = ____2____
    if 2 * a[mid + 1] >= a[mid - 1] + a[mid + 2]:
        ____3____
    else:
        r = mid - 1
print(a[r] + a[r + 1] - a[r + 2])
```

模拟小卷 2 二分

答题卷

姓名：_____

一、非选择题(共 4 小题，第 1 题 8 分，第 2 题 7 分，第 3 题 9 分，第 4 题 6 分，共 30 分)

1. 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
   第四空：_____ (1 分)
   第五空：_____ (1 分)
2. (1) _____ (1 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
3. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
   第四空：_____ (1 分)
4. 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)

模拟小卷 3 bfs 和队列

试题卷

一、非选择题(共 3 小题，每题 10 分，共 30 分)

1. 小 R 购买了一款翻译笔。该翻译笔翻译时只会从左到右依次将每个单词翻译为中文。翻译某个单词时，翻译笔首先会在内存中找这个单词的释义，如果没有，再从外部词典中查询，然后把查询结果放到内存中。然而，翻译笔内存容量仅够存 m 个单词，当内存中已经存了 m 个单词后，存入新单词时将会删除最早存入内存的词。初始内存为空。

   给出一篇长为 n 的文章(单词用一个非负整数表示，文章中两个单词是同一个单词，当且仅当它们对应的非负整数相同)。小 R 想要知道翻译这篇文章翻译笔需要查询几次外部词典。

```
m, n = map(int, input().split())
words = list(map(int, input().split()))
que = [0] * 1002; head = tail = 0
isInQue = [False] * 1002
cnt = 0
for i in words:
    if ___1___:
        que[tail] = i
        tail += 1
        isInQue[i] = True
        ___2___
        if ___3___:
            isInQue[que[head]] = False
            head += 1
print(cnt)
```

   (1) 若内存容量为 3，翻译一篇长度为 7 的文章：1,2,1,5,4,4,1，则需要查询外部词典_____次。

   (2) 请补全空缺处的代码。

2. 有一天我做了一个梦，梦见了一种很奇怪的电梯。大楼的每一层楼都可以

停电梯，而且第 i 层楼$(1 \le i \le N)$上有一个数字 $K_i(0 \le K_i \le N)$。电梯只有四个按钮：开，关，上，下。上下的层数等于当前楼层上的那个数字。当然，如果不能满足要求，相应的按钮就会失灵。

例如： $3,3,1,2,5$ 代表了 $K_i(K_1 = 3, K_2 = 3, \cdots)$。在 1 楼，按“上”可以到 4 楼，按“下”是不起作用的，因为没有 -2 楼。

那么，从 A 楼到 B 楼至少要按几次按钮呢？

(1) 沿用题目中的例子，从 1 楼到 5 楼至少按_____次按钮。

(2) 构建 n 条链表，第 i 条链表存储楼层 i 能到达的其他楼层，插入时直接插到链表的头部。然后用 bfs 计算答案。Python 代码如下：

```
n, a, b = map(int, input().split())
k = [0] + list(map(int, input().split()))
link = []
head = [-1] * 420
def add(x, y):
    link.append(    1    )
    head[x] = len(link) - 1
for i in range(1, n + 1):
    if i + k[i] <= n:
        add(i, i + k[i])
    if    2    :
        add(i, i - k[i])
dis = [333333] * 420
dis[a] = 0
que = [a] * 420
isInQue = [False] * 420
isInQue[a] = True
h, t = 0, 1
while h < t:
    x = que[h]
    h += 1
    3
    i = head[x]
    while i != -1:
```

```
            y = link[i][0]
            if dis[x] + 1 < dis[y]:
                dis[y] = dis[x] + 1
                if not isInQue[y]:
                    isInQue[y] = True
                    que[t] = y
                    t += 1
```

4

```
print(dis[b])
```

请补全空缺处的代码。

3. 有一个 $m \times m$ 的棋盘，棋盘上每个格子可能是红色、黄色或没有颜色的。任何一个时刻，你所站在的位置必须是有颜色的，只能向上、下、左、右四个方向前进。当你从一个格子走向另一个格子时，如果两个格子的颜色相同，那你不需要花费金币；如果不同，则你需要花费 1 个金币。

另外，你可以花费 2 个金币施展魔法让一个无色格子暂时变为你指定的颜色。但如果你使用了这个魔法，走到了这个暂时有颜色的格子上，你就不能继续使用魔法；只有当你离开这个位置，走到一个本来就有颜色的格子上的时候，你才能继续使用这个魔法，而当你离开了这个位置(施展魔法使得变为有颜色的格子)时，这个格子恢复为无色。

现在你要从棋盘的最左上角，走到棋盘的最右下角，并且想知道花费的最少金币数是多少。

(1) 如图 3-1 的棋盘，从左上角走到右下角，至少需要花费_____个金币。

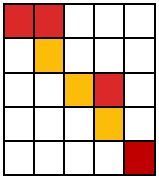


图 3-1

(2) 可以利用 Python 中的 PriorityQueue(优先队列)库。具体方法如下：

q = PriorityQueue()	声明一个优先队列
q.put(x)	将 x 加到 q 中
q.get()	返回 q 中**元素最小**的值并出队

处于简单考虑，认为列表的大小相当于其第一个元素的大小。

算法思路如下：利用类似 bfs 的方法，只不过把队列改成了优先队列，每次选取到达所需金币最少的点向外扩展，扩展过的点直接跳过。如果是有颜色的格子到空白格子，直接用魔法改成当前格子颜色即可。请补全空缺处的代码。

```
# 引入 PriorityQueue 类，读入 m 和棋盘颜色信息，Map[i][j]存储(I,j)位置的颜色，-1 表示空白，0 和 1 分别表示红和黄。代码略。
vis = [[False] * 105 for i in range(105)]
q = PriorityQueue()
q.put([0, 1, 1, Map[1][1]])
dx, dy = [1, -1, 0, 0], [0, 0, 1, -1]
res = -1
while q.qsize() > 0:
    t = q.get()
    if ____1____:
        res = t[0]
        break
    ____2____
    for i in range(4):
        tx = t[1] + dx[i]
        ty = ____3____
        col = Map[tx][ty]
        if tx < 1 or tx > m or ty < 1 or ty > m or vis[tx][ty] or (col == -1 and t[3] == -1):
            continue
        if col == -1:
            q.put([____4____])
        else:
            if col == t[3]:
                q.put([t[0], tx, ty, col])
            else:
                q.put([t[0] + 1, tx, ty, col])
print(res)
```

模拟小卷 3 bfs 和队列

答题卷

姓名：_____

一、非选择题(共 3 小题，每题 10 分，共 30 分)

1. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (3 分)
   第三空：_____ (3 分)
2. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
   第四空：_____ (2 分)
3. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
   第四空：_____ (2 分)

模拟小卷 4 dfs 和栈

试题卷

一、非选择题(共 3 小题，第 1 题 9 分，第 2 题 10 分，第 3 题 11 分，共 30 分)

1. 如图 4-1 所示，可以用四分树来表示一个 32×32 黑白图像，方法是用根节点表示整幅图像，然后把行列个分成两等份，按图 4-2 中的方式编号，从左到右对应 4 个子节点。如果某子节点对应的区域全黑或全白，则直接用一个黑节点或白节点表示；如既有黑又有白，则用一个灰节点表示，并且为这个区域重复上述流程。

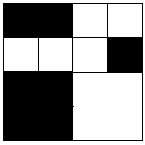
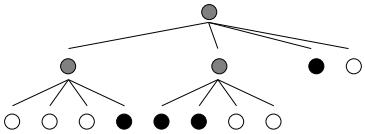


图 a



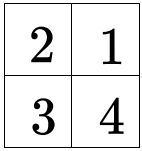


图 b

用 p 表示灰节点，f 表示黑节点，e 表示白节点，那么图 a 中的四分树的先序遍历可表示为 ppeeefpffeefe，代表的图像中共有 448 个黑像素。

可以用 Python 代码根据四分树的先序遍历求出其对应图像黑像素个数：

```
pre_order = input()  # 输入前序遍历
pointer = 0
L = 32 # 图片的长宽
pixel = [[0] * L for i in range(L)]  # pixel 记录每个像素的颜色，0 表示白，1 表示黑
stack = [[0, 0, L]]
cnt = 0
while len(stack):
    r, c, w = stack[-1] # 绘制以(r,c)为左上角，长宽为 w 的图像
    stack.pop()
    ch = ____1____
    pointer += 1
    if ch == 'p':
```

```
①stack.append([r, c, w // 2])
②stack.append([r, c + w // 2, w // 2])
③stack.append([r + w // 2, c, w // 2])
④stack.append([r + w // 2, c + w // 2, w // 2])
```

```
    elif ch == 'f':
        for i in range(r, r + w):
            for j in range(c, c + w):
                if pixel[i][j] == 0:
                    pixel[i][j] = 1
                    ____2____
print(cnt)
```

(1) 如果输入数据为 ppeeefffpeefe，则输出结果为_____。

(2) 请补全空缺处的代码。

(3) 根据题目意思，加框处代码顺序有误，正确顺序为_____。

2. 到中午了，小 R 想吃点东西。她购买了 n 份食品，第 i 份含有的能量为 w_i。但是小 R 不想吃得太胖又不想饿着，所以她决定补充的能量不少于 l，不超过 r。现在，请你帮她求出有多少种不同的选择食物的方案。

```
n, l, r = map(int, input().split())
w = list(map(int, input().split()))
ans = 0

def dfs(u, s):
    global ans
    if s > r:
        return
    if u >= n:
        if l <= s <= r:
            ____1____
        return
    dfs(____2____)
    dfs(u + 1, s)
```

```
dfs(___3___)
print(ans)
```

(1) 有 4 种食物，能量值分别为 $10,10,20,50$，$l=70,r=85$，则有_____种选择食物的方案。

(2) 下列说法正确的是_____(单选)。

A. 程序的时间复杂度为 $\mathcal{O}\ 2^n$ 。

B. 去掉加阴影的代码后，结果不变。

C. 该程序使用的是解析算法。

(3) 请补全空缺处的代码。

3. 乔治有一些**同样长**的小木棍，他把这些木棍随意砍成几段，直到每段的长都不超过 50。

现在，他想把小木棍拼接成原来的样子，但是却忘记了自己开始时有多少根木棍和它们的长度。

给出每段小木棍的长度，编程帮他找出原始木棍的最小可能长度。

思路如下：

设 $\mathrm{dfs}(k,last,rest)$，$k$ 表示正在拼第几根原来的长棍，$last$ 表示使用的上一根木棍的编号，$rest$ 表示当前在拼的长棍还有多少长度未拼。于是循环枚举下一根将要使用的木棍。并且考虑加上如下的优化：

1. 短木棍肯定比长木棍更灵活，因此将所有木棍从大到小排序。
2. 当拼接失败，要更换当前木棍时，不考虑与当前木棍相同的木棍。
3. 拼接时寻找长度不大于 $rest$ 的木棍，可以用二分查找。
4. 从小到大枚举原始长度，这个长度肯定能整除木棍总长度。

```
import sys
n = int(input())
a = [0] + list(map(int, input().split()))
s = sum(a)
for i in range(1, n): # 排序
    for j in range(i + 1, n + 1):
        if ___1___:
            a[i], a[j] = a[j], a[i]
nxt = [n] * (n + 1)  # nxt[i]记录每根木棍后面的最后一根与这根木棍长度相等的木棍
for i in range(n - 1, 0, -1):
```

```
    if a[i] == a[i + 1]:
        nxt[i] = nxt[i + 1]
    else:
        nxt[i] = ___2___
used = [False] * (n + 1)
m = 0
flag = False
def dfs(k, last, rest):
    global flag, m, origin
    if rest == 0:
        if k == m:
            flag = True
            return
        for i in range(1, n + 1):
            if not used[i]:
                break
        used[i] = True
        dfs(k + 1, i, origin - a[i])
        used[i] = False
        if flag:
            return
    l = last + 1
    r = n
    while l < r:
        mid = (l + r) // 2
        if a[mid] <= rest:
            r = mid
        else:
            l = mid + 1
    ___3___ # 从第一根长度不大于 rest 的木棍开始枚举
    while i <= n:
        if not used[i]:
            used[i] = 1
```

```
            dfs(k, i, rest - a[i])
            used[i] = 0
            if flag:
                return
            if rest == a[i] or rest == origin:
                return
            ____4____ # 优化 2
            if i == n:
                return
        i += 1
for origin in range(a[1], s // 2 + 1):
    if s % origin != 0:
        continue
    m = s // origin
    flag = False
    used[1] = True
    dfs(____5____)
    used[1] = False
    if flag:
        print(origin)
        sys.exit(0)
print(s)
```

(1) 若有 9 根小木棍，长度分别为 5,2,1,5,2,1,5,2,1，则原始木棍长度最小可能为_____。

(2) 请补全空缺处的代码。

模拟小卷 4 dfs 和栈

答题卷

姓名：_____

一、非选择题(共 3 小题，第 1 题 9 分，第 2 题 10 分，第 3 题 11 分，共 30 分)

1. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   (3) _____ (3 分)
2. (1) _____ (2 分)
   (2) _____ (2 分)
   (3) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
3. (1) _____ (1 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
   第四空：_____ (2 分)
   第五空：_____ (2 分)

模拟小卷 5 单调队列与单调栈

试题卷

一、非选择题(共 3 小题，第 1 题 8 分，第 2 题 8 分，第 3 题 14 分，共 30 分)

1. 一个含有 n 项的数列，求出每一项前的 m 个数(不包括这一项)到它这个区间内的最小值。若前面的数不足 m 项则从第 1 个数开始，若前面没有数则输出 0。

如果直接用两重循环求解，时间复杂度为 $O\ nm$ ，事实上，我们可以用单调队列的思想优化时间复杂度。用一个列表存储当前区间中可能成为最小值的元素。区间往后移一格，就往列表中加入新的数。如果新的数比老的数还要小，那么老的数在以后的区间中不可能成为最小值了，直接退出列表即可。最后列表中的元素是单调递增的，其中第一个元素就是我们的答案了。

Python 代码如下：

```
n, m = map(int, input().split())
a = list(map(int, input().split()))
q = [-1] * n
head = tail = 0
print(0)
for i in range(n):
    while head < tail and a[q[tail - 1]] > a[i]:
        tail -= 1
    while head < tail and ___1___:
        head += 1
    ___2___
    tail += 1
    if i == n - 1:
        break
    print(___3___)
```

(1) 请补全空缺处的代码。

(2) 已知每个元素最多入队 1 次出队 1 次，则程序的时间复杂度

为_____。

2. 约翰的 N $1 \le N \le 10^5$ 头奶牛站成一排(编号 $1 \sim N$)，奶牛 i 的身高是 H_i $1 \le H_i \le 10^6$ 。现在，每只奶牛都在向右看齐。对于奶牛 i，如果奶牛 j 满足 $i < j$ 且 $H_i < H_j$，我们可以说奶牛 i 可以仰望奶牛 j。求出每只奶牛离她最近的仰望对象。

我们可以利用栈，以 O n 的时间复杂度解决这个问题。

```
n = int(input())
h = [0]
for i in range(n):
    h.append(int(input()))
iCow = [0] * (n + 1)
st = []
for i in range(n, 0, -1):
    while len(st) > 0 and ___1___:
        st.pop()
    if len(st) == 0:
        iCow[i] = 0
    else:
        iCow[i] = ___2___
    ___3___
for i in range(1, n + 1):
    print(iCow[i])
```

(1) 这些奶牛的身高为 3,2,6,1,1,2，则每只奶牛的仰望对象为_____。

(2) 请补全空缺处的代码。

3. 在地面上确定一个起点，然后在起点右侧画 n 个格子，这些格子都在同一条直线上，第 i 个格子距离起点 dis_i。每个格子内有一个数字（整数），表示到达这个格子能得到的分数 val_i。玩家第一次从起点开始向右跳，跳到起点右侧的一个格子内。第二次再从当前位置继续向右跳，依此类推。

规则规定：

玩家每次都必须跳到当前位置右侧的一个格子内。玩家可以在任意时刻结束游戏，获得的分数为曾经到达过的格子中的数字之和。

现在小 R 研发了一款弹跳机器人来参加这个游戏。但是这个机器人有一个非常严重的缺陷，它每次向右弹跳的距离只能为固定的 d。小 R 希望

改进他的机器人，如果他花 g 个金币改进他的机器人，那么他的机器人灵活性就能增加 g，即弹跳距离变为 $[d-g, d+g]$。但是需要注意的是，每次弹跳的距离至少为 1。

现在小 R 希望获得至少 k 分，至少要花多少金币来改造他的机器人？

这道题可以用二分和单调队列解决。dp_i 表示经过第 i 个格子所能达到的最大得分，那么 dp_i 可以从前面能跳到当前格子的格子中得分最大的一个中计算出来，即 $dp_i = \max\limits_{dis_i-d-g \le \mathrm{dis}_k \le \mathrm{dis}_i-d+g} dp_k + s_i$。

```
n, d, k = map(int, input().split())
dis, val = [0], [0]
for i in range(n):
    tmp = list(map(int, input().split()))
    dis.append(tmp[0]) # dis 存储每个格子到起点的距离
    val.append(tmp[1]) # val 存储每个格子的数字
def check(g):
    jumpL = ___1___ # jumpL,jumpR 表示拿 g 个金币改造后能弹跳的距离范围。
    jumpR = d + g
    dp = [-123412341234] * (n + 1)
    dp[0] = 0
    que = [0] * (n + 1)
    canL = 0 # canL,canR 表示能跳到当前格子的格子范围
    canR = 0
    head = tail = 0
    for i in range(1, n + 1):
        while dis[i] - dis[canR] >= jumpL:
            while head < tail and ___2___:
                tail -= 1
            tail += 1
            que[tail] = canR
            ___3___
        while head < tail and dis[i] - dis[que[head + 1]] > jumpR:
```

```
            head += 1
        if head >= tail:
            continue
        dp[i] = ____4____
        if dp[i] >= k:
            return True
    return False
l, r = 0, dis[n]
res = -1
while l <= r:
    mid = (l + r) // 2
    if check(mid):
        res = mid
        r = ____5____
    else:
        l = mid + 1
print(res)
```

(1) 下列说法正确的是_____(单选)。

A. 得分不可能达到 k 时，程序输出 -1。

B. 程序的时间复杂度为 $\mathcal{O}\ n^2 \log n$ 。

C. 程序中使用了队列这一数据结构。

D. 程序中使用了栈这一数据结构。

(2) 请补全空缺处的代码。

模拟小卷 5 单调队列与单调栈

答题卷

姓名：_____

一、非选择题(共 3 小题，第 1 题 8 分，第 2 题 8 分，第 3 题 14 分，共 30 分)

1. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   (3) _____ (2 分)
2. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
3. (1) _____ (2 分)
   (2) 第一空：_____ (2 分)
   第二空：_____ (3 分)
   第三空：_____ (2 分)
   第四空：_____ (3 分)
   第五空：_____ (2 分)

模拟小卷 6 树相关

试题卷

一、选择题(共 5 小题，每题 3 分，共 15 分)

1. 已知一棵二叉树有 2013 个节点，则其中至多有(　　)个节点。

A. 1006　　B. 1007　　C. 1023　　D. 1024

2. 对于一棵二叉树，独立集是指两两互不相邻的节点构成的集合。例如，图 6-1 有 5 种不同的独立集（1 个双点集合、3 个单点集合、1 个空集），那么，图 6-2 有(　　)种不同的独立集。

图 6-1

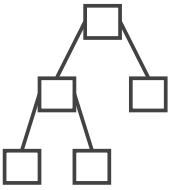


图 6-2

A. 12　　B. 14　　C. 16　　D. 18

3. 一颗二叉树的前序遍历序列是 ABCDEFG，后序遍历序列是 CBFEGDA，则根结点的左子树的结点个数可能是(　　)。

A. 0　　B. 2　　C. 4　　D. 6

4. 前序遍历和中序遍历相同的二叉树为且仅为(　　)。

A. 只有 1 个点的二叉树

B. 根结点没有左子树的二叉树

C. 非叶子结点只有左子树的二叉树

D. 非叶子结点只有右子树的二叉树

5. 如果根的高度为 1，具有 61 个结点的完全二叉树的高度为(　　)。

A. 5　　B. 6　　C. 7　　D. 8

二、非选择题(共 1 小题，每题 15 分，共 15 分)

6. 对于一棵二叉树的每一个节点，都满足该节点左子树的所有节点不大于该节点，该节点右子树的所有节点大于该节点，就称该二叉树为排序二叉树。

将一个列表中的元素依次插入到一个排序二叉树中，再对其进行中序遍

历，就可以得到列表元素从小到大排序的结果。

小 R 利用如上性质编写了如下代码，用于给 a 数组排序。

```
a = list(map(int, input().split()))
tree = None
for i in a: # 将列表元素插入排序二叉树中
    if tree == None:
        tree = [i, None, None]
    else:
        ___1___
        while u != None:
            fa = u
            if ___2___:
                op = 1
            else:
                op = 2
            u = fa[op]
        ___3___
st = [[tree, 0]]
while len(st):
    u, op = st.pop()
    if op == 1:
        print(u)
    else:
        ①st.append([u[0], 1])
        ②if u[1] != None:
            st.append([u[1], 0])
        ③if u[2] != None:
            st.append([u[2], 0])
```

(1) 请补全空缺处的代码。

(2) Else 语句中的编号的三段代码顺序有误，正确顺序为_____。

模拟小卷 6 树相关

答题卷

姓名：_____

一、选择题(共 5 小题，每题 3 分，共 15 分)

题号	1	2	3	4	5
答案					

二、非选择题(共 1 小题，每题 15 分，共 15 分)

6. (1) 第一空：_____ (3 分)

第二空：_____ (4 分)

第三空：_____ (4 分)

(2) _____ (4 分)

模拟小卷 7 排序

试题卷

一、选择题(共 2 小题，每题 4 分，共 8 分)

1. 体育课的铃声响了，同学们都陆续地奔向操场，按老师的要求从高到矮站成一排。每个同学按顺序来到操场时，都从排尾走到排头，找到第一个比自己高的同学，并站在他的后面。这种站队的方法类似于(　　)算法。

A. 快速排序　　B. 插入排序　　C. 冒泡排序　　D. 归并排序

2. 冒泡排序算法的伪代码如下：

输入：数组 L，$n \geq k$。输出：按非递减顺序排序的 L。
算法 BubbleSort：

```
1. FLAG ← n //标记被交换的最后元素位置
2. while FLAG > 1 do
3.     k ← FLAG -1
4.     FLAG ← 1
5.     for j=1 to k do
6.         if L(j) > L(j+1) then do
7.             L(j) ↔ L(j+1)
8.             FLAG ← j
```

对 n 个数用以上冒泡排序算法进行排序，最少需要比较(　　)次。

A. n^2　　B. $n-2$　　C. $n-1$　　D. n

二、非选择题(共 2 小题，每题 11 分，共 22 分)

3. 希尔排序是一种改进版的插入排序，时间复杂度为 $\mathcal{O}\left(n^{\frac{4}{3}}\right)$(不同写法快慢不同)。其基本思想是：按下标除以 k 的余数把原数列分成 gap 组，再对这 gap 组分别进行插入排序。不断减小 gap 直至减小到 1，原数列便完成排序。

(1) 根据以上原理补全空缺处的代码。

```
def shellSort(arr):
    import math
    gap=1
```

```
    while(gap < len(arr)/3):
        gap = gap*3+1
    while gap > 0:
        for i in ___1___:
            ___2___
            j = i-gap
            while j >=0 and arr[j] > temp:
                ___3___
                j-=gap
            arr[j+gap] = temp
        gap = math.floor(gap/3)
    return arr
print(shellSort([1,4,6,3,4,2,2,7,5,7,9,10]))
```

(2) 当 $gap = 1$ 时，程序变成普通的插入排序，也就是说，程序在普通插入排序的基础上还做了多次分组的插入排序，为什么时间效率更高了？试解释原因。

4. 明明想在学校中请一些同学一起做一项问卷调查，为了实验的客观性，他先用计算机生成了 N 个 1 到 1000 之间的随机整数 $N \leq 100$，对于其中重复的数字，只保留一个，把其余相同的数去掉，不同的数对应着不同的学生的学号。然后再把这些数从小到大排序，按照排好的顺序去找同学做调查。请你协助明明完成“去重”与“排序”的工作。

(1) 这道题可以用冒泡排序解决，请补全空缺处的代码：

```
def bubbleSort(arr):
    n = len(arr)
    for i in range(1, n):
        for j in ___1___:
            if arr[j] < arr[j - 1]:
                arr[j], arr[j - 1] = arr[j - 1], arr[j]
    return arr


def unique(arr):
    ___2___
    n = len(arr)
```

```
    for i in range(1, n):
        if ____3____:
            new.append(arr[i])
    return new
```

(2) 这道题可以用计数排序解决，请补全空缺处的代码：

```
def countSort(arr):
    count = [0] * 1001
    for i in arr:
        ____1____
    new = []
    for i in ____2____:
        if count[i] > 0:
            new.append(i)
    return new
```

模拟小卷 7 排序

答题卷

姓名：_____

一、选择题(共 2 小题，每题 4 分，共 8 分)

题号	1	2
答案		

二、非选择题(共 2 小题，每题 11 分，共 22 分)

3. (1) 第一空：_____ (2 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
   (2)* _____ (5 分)
4. (1) 第一空：_____ (3 分)
   第二空：_____ (2 分)
   第三空：_____ (2 分)
   (1) 第一空：_____ (2 分)
   第二空：_____ (2 分)

参考答案

模拟小卷 1 双指针

题目总览

1	A-B 数对	双指针	-
2	逛画展	双指针，桶	USACO
3	排列排序	双指针	EAEC-10

一、非选择题

1. (1) 3
   (2) 第一空： range(n-1,i,-1)
   第二空： nums[top]=nums[i]
   第三空： i >= 0 and nums[j] – nums[i] < c
   第四空： cnt[i] * cnt[j]
   (3) $\mathcal{O}\ n^2$
2. (1) $a = 2, b = 7$
   (2) 第一空： r<n-1
   第二空： l+=1
   第三空： cnt[artist[l]]==0
   (3) $\mathcal{O}\ n$

3 (1) 7
   (2) 第一空： a[l]==l+1
   第二空： mx=max(a[r], mx)
   第三空： l = r + 1

模拟小卷 2 二分

题目总览

1	借教室	差分，二分答案	NOIP 2012 提高组 Day2 T2
2	养獾	二分答案(二分查找写法)，排序	srm476 div2
3	车站检票	排序，二分答案(二分查找写法)	-

4	-	差分，二分答案	八梅巫

一、非选择题

1. 第一空：diff[t[i] + 1] -= d[i]
   第二空：need[i] = need[i - 1] + diff[i]
   第三空：isok(m)
   第四空：l = mid + 1
   第五空：l
2. (1) C
   (2) 第一空：h[i] + g[i] * (m - 1)
   第二空：sum(z[:m + 1])
   第三空：l = mid + 1
3. (1) 8
   (2) 第一空：range(n - 1, i, -1)
   第二空：t // i
   第三空：L <= R
   第四空：min(ans, mid) 或 mid
4. 第一空：l = 2
   第二空：mid = (l + 2 * r) // 4 或 mid = (l + 2 * r + 2) // 4
   第三空：l = 2 * mid + 2

模拟小卷 3 bfs 和队列

题目总览

1	机器翻译	队列	NOIP 2010 提高组 Day1 T1
2	奇怪的电梯	bfs，SPFA，链表	洛谷
3	棋盘	bfs	NOIP 2017 普及组 T3

一、非选择题

1. (1) 5
   (2) 第一空：not isInQue[i]
   第二空：cnt += 1
   第三空：tail - head > m

2. (1) 3
   (2) 第一空：[y, head[x]]

第二空：　i - k[i] >= 1
第三空：　isInQue[x] = False
第四空：　i = link[i][1]

3. (1) 8
(2) 第一空：　t[1] == m and t[2] == m
第二空：　vis[t[1]][t[2]] = True
第三空：　t[2] + dy[i]
第四空：　t[0] + 2, tx, ty, t[3]

模拟小卷 4 dfs 和栈

题目总览

1	四分树	栈	
2	猫粮规划	dfs，剪枝	洛谷
3	小木棍	dfs，剪枝	

一、非选择题

1. (1) 640
(2) 第一空：　pre_order[pointer]
第二空：　cnt += 1
(3) ④③①②(注：这个顺序不影响 cnt 的答案，但会影响 pixel 的结果)

2. (1) 4
(2) B
(3) 第一空：　ans += 1
第二空：　u + 1, s + w[u]
第三空：　0, 0

3. (1) 6
(2) 第一空：　a[i] < a[j]
第二空：　i
第三空：　i = l
第四空：　i = nxt[i]
第五空：　1, 1, origin - a[1]

模拟小卷 5 单调队列与单调栈

题目总览

1	滑动窗口	单调数据结构	-
2	Look Up S	单调数据结构	USACO 2009
3	跳房子	二分答案，单调数据结构	NOIP 2017 普及组 T4

二、非选择题

1. (1) 第一空：q[head] + m <= i
   第二空：q[tail] = i
   第三空：a[q[head]]
   (2) $O\ n$
2. (1) 3,3,0,6,6,0
   (2) 第一空：h[i] >= h[st[-1]]
   第二空：st[-1]
   第三空：st.append(i)
3. (1) A
   (2) 第一空：max(1, d - g)
   第二空：dp[que[tail]] < dp[canR]
   第三空：canR += 1
   第四空：max(dp[i], dp[que[head + 1]] + val[i])
   第五空：mid - 1

模拟小卷 6 树相关

一、选择题

1-5. ABBDB

二、非选择题

6. (1) 第一空：u = tree
   第二空：i <= u[0]
   第三空：fa[op] = [i, None, None]
   (2) ③①②

模拟小卷 7 排序

一、选择题

1-2. CC

二、非选择题

3. (1) 第一空：range(gap,len(arr))
第二空：temp = arr[i]
第三空：arr[j+gap]=arr[j]

(2)* 提示：希尔排序是基于插入排序的以下两点性质而提出改进方法的：

1. 插入排序在对几乎已经排好序的数据操作时，效率高，即可以达到线性排序的效率；

2. 但插入排序一般来说是低效的，因为插入排序每次只能将数据移动一位；

4. (1) 第一空：range(n - 1, i - 1, -1)
第二空：new = [arr[0]]
第三空：arr[i] != arr[i - 1]

(2) 第一空：count[i] += 1
第二空：range(1001)