

# 2024 年秋季学期算法基础期中考试（答案）

学号 \_\_\_\_\_ 姓名 \_\_\_\_\_

**主定理：** 令  $a \geq 1$  和  $b > 1$  是常数， $f(n)$  是一个函数， $T(n)$  是定义在非负整数上的递归式：

$$T(n) = aT(n/b) + f(n)$$

其中我们将  $n/b$  解释为  $\lfloor n/b \rfloor$  或  $\lceil n/b \rceil$ 。那么  $T(n)$  有如下渐进界：

1. 若对某个常数  $\epsilon > 0$  有  $f(n) = O(n^{\log_b a - \epsilon})$ ，则  $T(n) = \Theta(n^{\log_b a})$ 。
2. 若对整数  $k \geq 0$  有  $f(n) = \Theta(n^{\log_b a} \lg^k n)$ ，则  $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$ 。
3. 若对某个常数  $\epsilon > 0$  有  $f(n) = \Omega(n^{\log_b a + \epsilon})$ ，且对某个常数  $c < 1$  和所有足够大的  $n$  有  $af(n/b) \leq cf(n)$ ，则  $T(n) = \Theta(f(n))$ 。

**一、判断题**（根据表述判断正误，并简要说明理由；每题 5 分，共 30 分；每一题判断答案正确得 2 分，解释理由正确得 3 分。）

1. (T, F) 递归式  $T(n) = 2T(n/5) + \sqrt{n}$  的解满足  $T(n) = \Theta(n^{\log_5 2})$ 。  
F 错误。  
由主定理可得，对于  $\epsilon = 0.5 - \log_5 2$  有  $f(n) = \Omega(n^{\log_5 2 + \epsilon})$ ，则  $T(n) = \Theta(\sqrt{n})$ 。
2. (T, F) 堆排序是一种不稳定的排序算法，给长度为  $n$  的数组 A 建立最大堆的时间复杂度为  $\Theta(n \log n)$ 。  
F 错误。  
给长度为  $n$  的数组 A 建立最大堆的时间复杂度为  $\Theta(n)$ 。
3. (T, F) 插入排序在最后一趟开始之前，所有的元素可能都不在其最终的位置上。  
T 正确。  
在插入排序中，如果待排序列中的最后一个元素其关键字值为最小，则在最后一趟开始之前，前  $n-1$  个排好序的元素都不在其最终位置上，与排好序后的位置相差一个位置。
4. (T, F) 在处理链表而不是数组的数据结构时，快速排序比归并排序更具有优势，因为它不需要额外的空间来进行合并。  
F 错误。  
在链表中使用归并排序不需要额外空间进行合并。同时链表不支持随机访问，因此快速排序无法高效地在链表中选择基准和进行分区，归并排序在链表上更具有优势。
5. (T, F) 某二叉树的后序和中序遍历序列正好一样，则该二叉树中的任何结点一定都无右孩子。  
T 正确。  
后序：L, R, root，中序：L, root, R，要想两个序列一样，一定无 R。
6. (T, F) 对于红黑树中的每一个结点  $x$ ， $x$  的左子树和右子树的高度最多相差 1。  
F 错误。  
题述是二叉平衡树，红黑树每个结点左右子树高度不会超过右子树高度的 2 倍，右子树高度也不会超过左子树高度的 2 倍。

## 二、简答题（根据题目要求写出解答过程；每题 10 分，共 40 分）。

1. 有  $n$  个木块，大小分别为  $a_1, \dots, a_n$ ；另有  $m$  个大小为  $V$  的容器。定义一个容器被填满当且仅当其中放入了两个木块  $a_i, a_j$ ，且  $a_i + a_j \leq V$ 。每个木块只能使用一次。请给出时间复杂度为  $O(n \log n)$  的算法来判断是否存在一种木块分配方案，使得这  $m$  个容器都被填满。给出伪代码，并说明算法的工作原理。

注意，你的算法不需要输出方案，只需要输出是或者否。

解：

先给  $a_1, a_2, \dots, a_n$  排序，然后维护两个指针  $left, right$  和一个计数器  $num$  表示当前  $a_1, a_2, a_{left}$  中有多少个木块没有被使用过，得到最多能够填满多少个容器。基本思想是：从大到小枚举  $right$ ，对于目前最大的  $right$ ，找到使得  $a_{left} + a_{right} \leq V$  成立的最大  $left$ ，那么  $a_1, a_2, a_{left}$  中没有用过的木块和  $a_{right}$  一起就都是一个合法的可以放到容器里的木块配对方案。如果这样的  $a_1, a_2, a_{left}$  存在，那么就任选一个和  $a_{right}$  配对。（这是因为：如果  $a_1, a_2, a_{left}$  可以和  $a_{right}$  配对，那么也一定能和比  $a_{right}$  小的木块配对，且  $a_{right+1}, \dots, a_n$  的配对方案都已经决定过了，那么现在和  $a_{right}$  配对一定是最优解之一。）之后  $num--$ ,  $right--$ ，前一个  $right$  求出的  $left$  值就可以作为初始值继续往上搜索下一个  $right$  对应的  $left$ 。直到  $right \leq left$ ，这表明剩下的  $num$  个没有用到的木块都可以两两组合（这是因为上一轮的搜索保证了  $a_{left} + a_{right+1} \leq V$ ），那么再给答案加上  $\lceil \frac{num}{2} \rceil$ 。最后判断填满的容器数是否  $\geq m$  即可。

时间复杂度：排序时间复杂度  $O(n \log n)$ ，向内搜索时间复杂度  $O(n)$ ，总时间复杂度  $O(n \log n)$ 。

---

### Algorithm 1 向内搜索

---

```
1: sort a
2:  $l = 0$ ;  $r = n$ ;  $num = 0$ ;  $ans = 0$ ;
3: while  $l < r$  do
4:   while  $l + 1 < r$  and  $a_{l+1} + a_r \leq V$  do
5:      $l++$ ;  $num++$ ;
6:   if  $num > 0$  then
7:      $num--$ ;  $ans++$ ;
8:    $r--$ ;
9:  $ans = ans + num/2$ ;
10:
11: if  $ans \geq m$  then
12:   Return True
13: else
14:   Return False
```

---

2. 扩张红黑树使得它能以  $O(1)$  的时间给出动态数集  $Q$  中两个最接近的数之间的差值 (MIN-GAP)。例如， $Q = \{1, 5, 9, 15, 18, 22\}$ ，MIN-GAP 返回  $18 - 15 = 3$ ，因为 15 和 18 是  $Q$  中两个最接近的数。写出该数据结构并说明 MIN-GAP 如何计算。

解：将红黑树的 key 设为动态数集  $Q$  中的值。结点  $x$  的附加值包含以  $x$  为根的子树中两个最接近的数之间的差值 ( $x.min\_gap$ )，以  $x$  为根的子树中的最大值 ( $x.max$ )，和以  $x$  为根的子树中的最小值 ( $x.min$ )。叶子结点的  $min\_gap$  设为  $\infty$ ，非叶子结点  $x$  的  $x.min\_gap = \min\{x.left.min\_gap, x.right.min\_gap, x - x.left.max, x.right.min - x\}$ 。因此直接求根结点的  $min\_gap$  即可在  $O(1)$  的时间给出 MIN-GAP。

3. 对于递归式  $T(n) = T(n/3)/2 + T(2n/3)/2 + \lg n$ ，请给出一个好的渐近上界，并用代入法验证。

解：

每一层的代价可以表示为  $\frac{1}{2^i} \lg \frac{3^{i-1}-1}{2} n^{2^i}$ ，简单放缩后得到每层的代价约为  $O(\lg n)$ ，因此估计它的渐近上界为  $O(\lg^2 n)$ ，用代入法验证：

$$\begin{aligned}
 T(n) &= T(n/3)/2 + T(2n/3)/2 + \lg n \\
 &= \frac{1}{2}c(\lg^2(n/3)) + \frac{1}{2}c(\lg^2(2n/3)) + \lg n \\
 &= c\lg^2 n + (-\lg 3 + \lg 2/3 + (1/c))c\lg n + ((1/2)\lg^2 3 + (1/2)\lg^2(2/3))c \quad (c > 1/(\lg 3 + \lg(3/2))) \\
 &\leq c\lg^2 n
 \end{aligned}$$

其中，常数项  $C = ((1/2)\lg^2 3 + (1/2)\lg^2(2/3))c$  被省略。

同时，使用 Akra-Bazzi 方法也可以求出该解。

4. 对于图一所给的斐波那契堆，黑色节点表示结点的 mark 属性为 true:
- 1) 抽取最小结点，画出操作后最后的结果（需要考虑 CONSOLIDATE(合并) 操作, ）。
  - 2) 在 1) 的基础上，把关键字为 35 的节点的值降低为 4，画出操作后的最后的结果（需考虑 CASCADING-CUT(级联切断) 操作）。

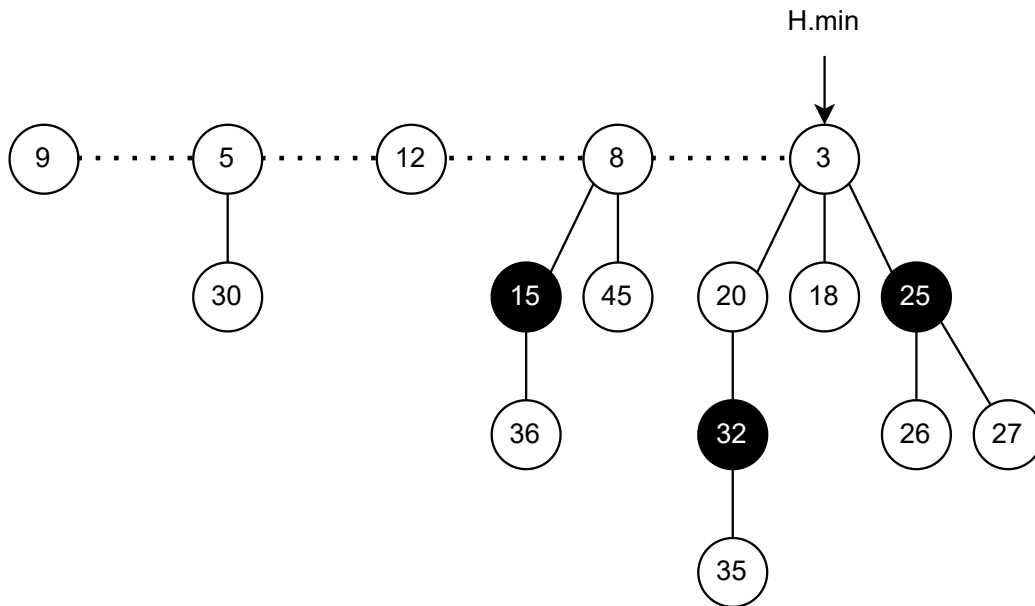
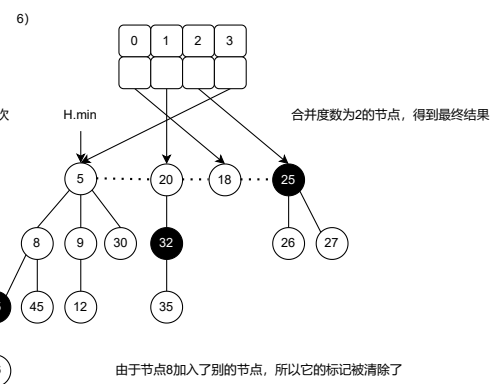
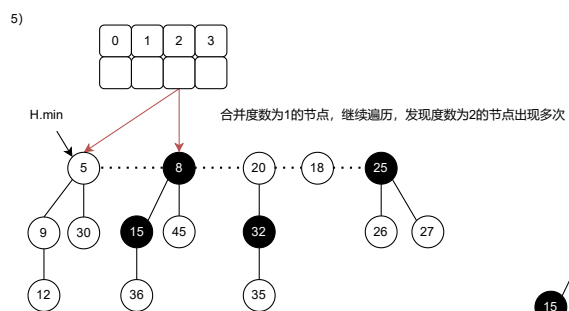
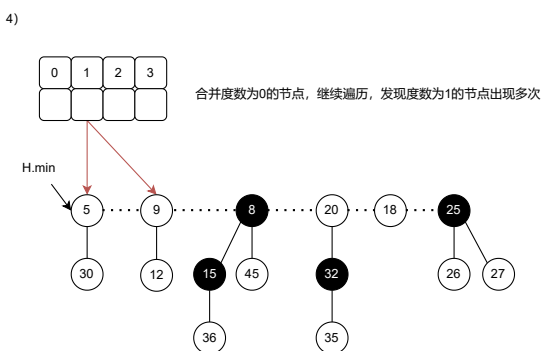
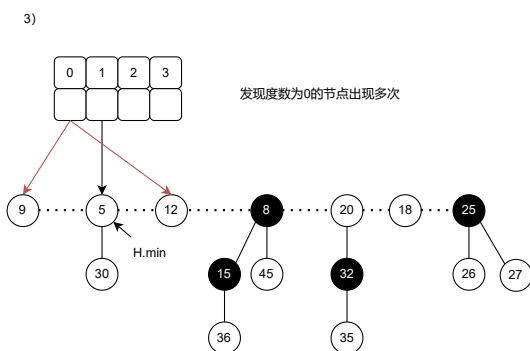
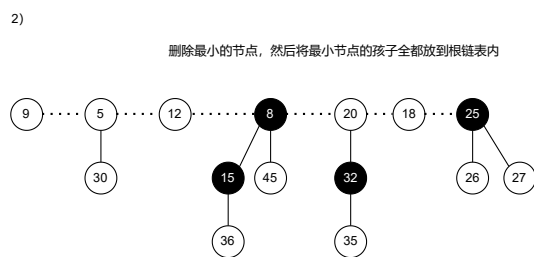
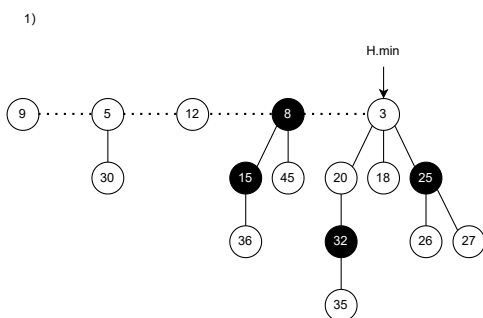
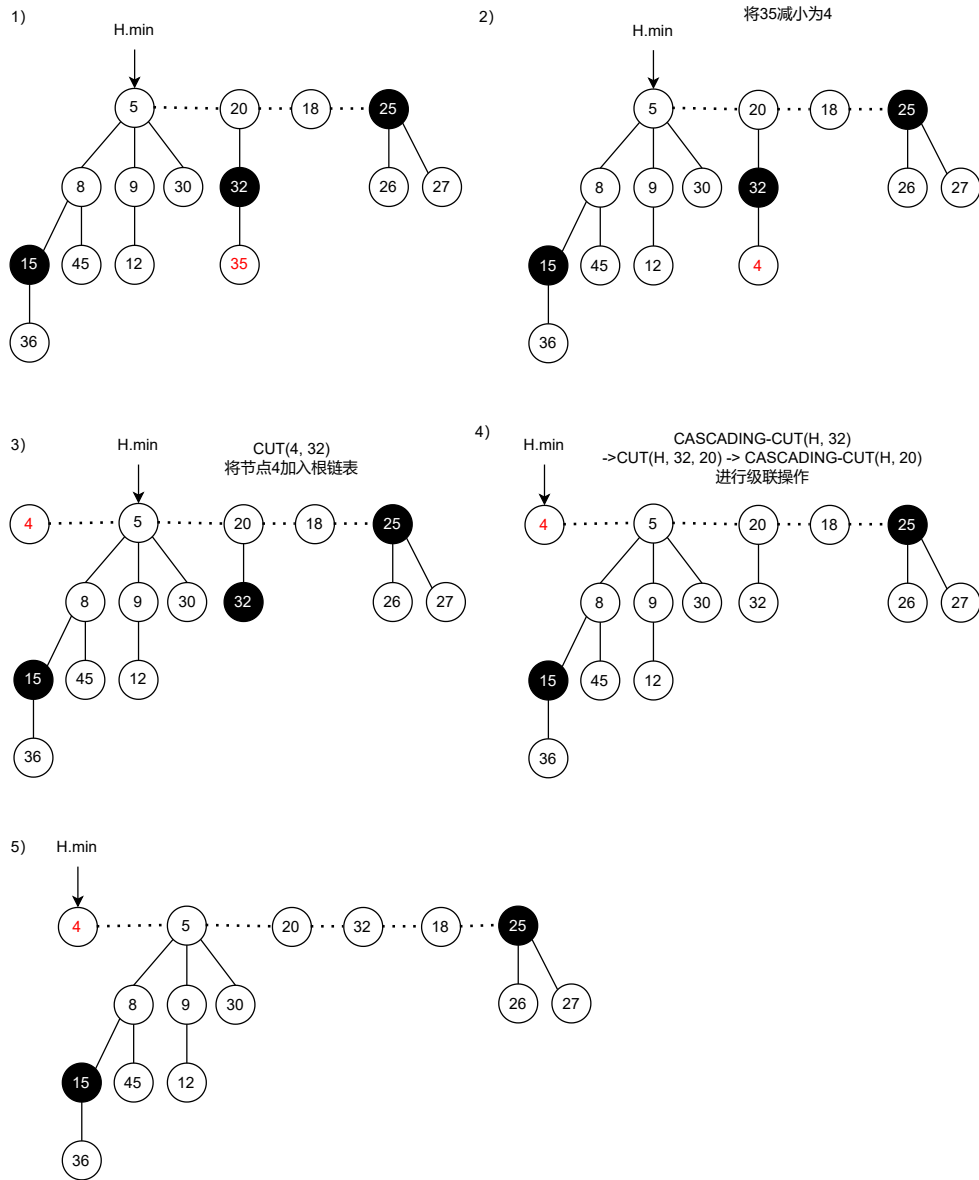


图 1: 斐波那契堆

1)



2)



三、综合题（根据题目要求写出解答过程；每题 15 分，共 30 分）。

1. 快速排序

(a) 假设不是对整个数组排序，而只要确定一个给定的  $n$  个元素数列的第  $m$  个最小者。修改快速排序算法实现这一目的，以避免为进行一次完全排序所需要的额外操作。填写 QuickSelect 算法。

---

**Algorithm 2** QUICKSORT(arr, low, high)

---

```
1: if low < high then
2:   pivotIndex = PARTITION(arr, low, high)
3:   QUICKSORT(arr, low, pivotIndex - 1)
4:   QUICKSORT(arr, pivotIndex + 1, high)
```

---

---

**Algorithm 3** PARTITION(arr, low, high)

---

```
1: pivot = arr[high]
2: i = low - 1
3: for j = low to high - 1 do
4:   if arr[j] <= pivot then
5:     i = i + 1
6:     SWAP(arr[i], arr[j])
7: SWAP(arr[i + 1], arr[high])
8: Return i + 1
```

---

---

**Algorithm 4** QuickSelect(arr, low, high, m)

---

```
1:
2:
3:
4:
5:
6:
7:
8:
9:
10:
11:
12:
13:
```

---

(b) 给出 QuickSelect 的平均键码比较次数  $C_{nm}$  的递推形式表达式。

(c) 当  $m = 1$  时，计算  $C_{n,1}$ 。

解：  
(a)

---

**Algorithm 5** QuickSelect(arr, low, high, m)

---

```
1: if low = high then
2:   Return arr[low]
3: pivotIndex = PARTITION(arr, low, high)
4: k = pivotIndex - low + 1
5: if m = k then
6:   Return arr[pivotIndex]
7: else if m < k then
8:   Return QuickSelect(arr, low, pivotIndex - 1, m)
9: else
10:  Return QuickSelect(arr, pivotIndex + 1, high, m - k)
```

---

(b) 递推式是  $C_{11} = 0$  和当  $n > 1$  时,

$$C_{nm} = n + 1 + \frac{1}{n} \left( \sum_{1 \leq s < m} C_{(n-s)(m-s)} + \sum_{m < s \leq n} C_{(s-1)m} \right)$$

(c) 首先, 根据递推公式, 当  $m = 1, n > 1$  时, 我们有:

$$C_n = n + 1 + \frac{1}{n} \sum_{1 \leq s \leq n-1} C_s$$

可得:

$$nC_n = n^2 + n + \sum_{1 \leq s \leq n-1} C_s$$

$$(n-1)C_{n-1} = n^2 - n + \sum_{1 \leq s \leq n-2} C_s$$

两式相减:

$$nC_n - (n-1)C_{n-1} = 2n + C_{n-1}$$

得到:

$$C_n - C_{n-1} = 2$$

而  $C_1 = 0$  故  $C_n = 2n - 2$

2. 二叉搜索树一定程度上可以提高搜索效率，但在最坏情况下，仍需要  $O(n)$  时间完成一次搜索。为此，有人提出了 Treap 树结构，即二叉搜索树 (tree) 与最小堆 (heap) 的组合。具体而言，该树内的每个结点  $x$  都有一个关键字值  $x.key$ ，另外，还要为每个结点  $x$  指定一个优先级  $x.priority$  (**假设所有结点的优先级都是不同的，而且所有的关键字也是不同的**)。Treap 树的结点被排列成让关键字遵循二叉搜索树的性质，且优先级遵循最小堆顺序性质：

- 如果  $v$  是  $u$  的左孩子，则  $v.key < u.key$ 。
- 如果  $v$  是  $u$  的右孩子，则  $v.key > u.key$ 。
- 如果  $v$  是  $u$  的孩子，则  $v.priority > u.priority$ 。

(a) 给定一个由 (关键字, 优先级) 对组成的结点集合  $\{(x_i.key, x_i.priority) | 1 \leq i \leq n\}$ ，存在多少颗不同的 Treap 树能表示这个集合。

(b) 给出 TREAP-INSERT( $T, x$ ) 的伪代码 (可以调用 LEFT-ROTATE 和 RIGHT-ROTATE)。

(c) 假设 Treap 树中各结点的优先级独立且均匀分布。求解  $n$  结点 Treap 树中的叶子节点的期望数量  $E(n)$ 。

解：

(a) 1

(b)

---

**Algorithm 6** TREAP-INSERT( $T, x$ )

---

```

1:  $key = x.key$ 
2:  $priority = x.priority$ 
3: if  $T$  is NULL then
4:   return NewNode( $key, priority$ )
5: if  $key < T.key$  then
6:    $T.left = \text{TREAP-INSERT}(T.left, x)$ 
7:   if  $T.left.priority < T.priority$  then
8:      $T = \text{RIGHT-ROTATE}(T)$ 
9: else
10:   $T.right = \text{TREAP-INSERT}(T.right, x)$ 
11:  if  $T.right.priority < T.priority$  then
12:     $T = \text{LEFT-ROTATE}(T)$ 
13: return  $T$ 

```

---

(c)  $E(0) = 0, E(1) = E(2) = 1, E(3) = \frac{4}{3}$ ，又有

$$E(n) = \frac{1}{n} \sum_{k=0}^{n-1} (E(k) + E(n-1-k))$$

假设当  $1 < n \leq n_0$  时， $E(n) = \frac{n+1}{3}$ ，则

$$\begin{aligned}
E(n_0+1) &= \frac{1}{n_0+1} \sum_{k=0}^{n_0} (E(k) + E(n_0+1-1-k)) \\
&= \frac{1}{n_0+1} \left( \sum_{k=2}^{n_0-2} \frac{k+1+n_0-k+1}{3} \right) + \frac{2}{n_0+1} \left( 0 + \frac{n_0+1}{3} + 1 + \frac{n_0}{3} \right) = \frac{n_0+2}{3}
\end{aligned}$$

故  $n = n_0 + 1$  时， $E(n) = \frac{n+1}{3}$ 。递推得， $E(n) = \frac{n+1}{3}, (n > 1)$ 。综上，

$$E(n) = \begin{cases} \frac{n+1}{3}, & n > 1 \\ 1, & n = 1 \\ 0, & n = 0 \end{cases}$$

**四、附加题**（根据题目要求写出解答过程，共 10 分）。

小迪作为一名投资者，常常研究不同的投资策略。最近，他提出了“价值区间”分析法：给定某只股票在一定时间范围内的价格  $a = [a_0, a_1, \dots, a_{n-1}]$  ( $a_i > 0$ )，区间大小  $W$ ，以及价值差  $A$ ，若存在  $i < j \in [0, n-1]$ ，且  $|i - j| \leq W$ ， $|a_i - a_j| \leq A$ ，则称该股票存在“价值区间”  $(i, j)$ 。

例如， $a = [1, 3, 5, 10]$ ，若  $W = 2$ ， $A = 2$ ，则存在两个“价值区间”： $(0, 1)$ ， $(1, 2)$ ；若  $W = 2$ ， $A = 1$ ，则不存在“价值区间”。

请你为小迪设计一个时间复杂度为  $O(n)$  的算法 `VALUERANGE`，以判断给定的股票是否存在“价值区间”。给出算法的伪代码，并分析其时间复杂度。（输入： $a, W, A$ ；输出：True or False）

---

**Algorithm 7** `VALUERANGE`( $a, W, A$ )

---

```
1:  $map = \{\}$ 
2:  $size = A + 1$ 
3: for  $i = 0$  to  $n - 1$  do
4:    $idx = \lfloor \frac{a_i}{size} \rfloor$ 
5:   if  $idx$  in  $map$  then
6:     return True
7:    $l = idx - 1$ 
8:    $r = idx + 1$ 
9:   if  $l$  in  $map$  and  $|a_i - map[l]| \leq A$  then
10:    return True
11:  if  $r$  in  $map$  and  $|a_i - map[r]| \leq A$  then
12:    return True
13:   $map[idx] = a_i$ 
14:  if  $i \geq k$  then
15:     $pop\ map[\lfloor \frac{a_{i-k}}{size} \rfloor]$ 
16: return False
```

---

时间复杂度  $O(n)$ ，空间复杂度  $O(k)$ 。