《算法设计与分析》第三次手写作业

2024.11.25 唐治江

1. Distributing Candy Game

1. **建模**:

- **输入**：老师左右手数字(l_0, r_0)，第i个学生左右手数字(l_i, r_i)，总共有n个学生。
- **输出**：获取最多糖果学生能获取到的糖果最小值
- **思路**：

  对于一种学生的排列P，第i个学生获得的糖果可以记作：

$$C_i^P = \frac{\Sigma_j^i l_j}{r_i}$$

  可以证明，需要将n个学生中右手数字最大的学生排列到最后，才能保证最小化拿到最多糖果数量。

  证明：假设右手数字最大的学生标号记为i_{max}，则有如下不等式

$$C_n^P = \frac{\Sigma_j^n l_j}{r_n} \geq \frac{\Sigma_j^{i_{max}} l_j}{r_{i_{max}}} = C_{i_{max}}^P$$

  所以需要将右手数字最大的学生放到最后，否则有更优的答案。同理可以推导出需要将右手数字第二大的位置放到$(n-1)$的位置上，以此类推。

2. **算法描述**:

- 将学生的右手数字r_i由小到大排列
- 将老师作为列头，按照之前的排序结果依次排列学生，分别计算每个学生手里的糖果C_k。
- 输出糖果的最大数量C_{max}。

3. **时间复杂性**：时间复杂度是 $O(nlogn)$ 。排序的时间复杂度为 $O(n \log n)$，计算糖果时的复杂度为 $O(n)$。
4. **n空间复杂性**：空间复杂度是 $O(n)$。需要记录每个学生的糖果。

2. Array Partition

1. **建模**

- **输入**： $2n$ 个整数的数组 `nums`，将其为了n对$(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)$
- **输出**：返回$\sum_i^n min(a_i, b_i)$的最大值

- **思路：**

  两两分组（将最小值与次小值组成一组）可以实现最大化。

2. **算法描述**

- 将数组 `nums` 升序排序。
- 选择排序后的偶数位元素的和作为结果。

3. **时间复杂性**：排序的时间复杂度为 $O(n \log n)$。
4. **空间复杂性**：排序所需额外空间为 $O(1)$。

3. Delete Number Game

1. **建模**

- **输入**：
  - 一个长度为 n 的非负整数，
  - k，表示需要删除的位数。
- **输出**：删除 k 位后形成的最小整数。
- **思路：**

  为了使新整数尽可能小，应该尽量保留较小的数字，并删除较大的数字。

2. **算法描述**

- 初始化一个空栈。
- 遍历整数的每个数字：
  - 如果栈为空或栈顶数字大于当前数字，将当前数字压入栈中。
  - 如果栈顶数字小于当前数字，从栈中弹出数字，直到栈为空或栈顶数字小于当前数字。
  - 如果总的数字达到$n-k$个，则将所有数字压入栈内。
- 遍历结束后，栈中剩余的数字就是应该保留的数字。

3. **时间复杂性**：时间复杂度为 $O(n)$。
4. **空间复杂性**：空间复杂度为 $O(n)$，用于存储栈。

4. Ex. Container Balancing Operations

1. **建模**

- **输入**：一个数组 `containers`，其中 `containers[i]` 表示第 i 个容器中的物品数量。
- **输出** ：使所有容器物品相等所需的最小操作数，若无法实现，返回 -1。
- **思路**：

  总物品数量必须能被容器数量整除，否则返回-1。若可以整除，计算每个容器当前项目数量与期望数量之间的差异，将所有超过期望数量的项目数加起来即为答案。

2. **算法描述**

- 计算数组 `containers` 的平均值 `E`，如果平均值不为整数，则返回-1，否则继续。

- 初始化一个变量`res`来存储所需的总操作次数，设为 0。
- 遍历数组`containers`，计算每个容器当前项目数量与期望数量之间的差异
    - 若`containers[i]-E>0`，则`res+=containers[i]-E`
- 返回总操作次数`res`。

3. **时间复杂性**：时间复杂度为$O(n)$，遍历`containers`。
4. **空间复杂性**：空间复杂度为$O(1)$。