算法基础

第六次作业（DDL：2024 年 11 月 21 日 23:59）

解答过程中请写出必要的计算和证明过程

Q1.(20 分) 对于用最少的硬币找 n 美分零钱的问题，假定有 25 美分、10 美分、5 美分、2 美分和 1 美分五种面额的硬币：设计贪心算法求解找零问题，**给出算法思路**，并证明你的**贪心算法的正确性**，不必给出伪代码。

solution:

算法思路：先取 $\lfloor \frac{n}{25} \rfloor$ 个 25 美分硬币，然后相同的方式对 $n - \lfloor \frac{n}{25} \rfloor$ 取最多的 10 美分硬币，依次取 5 美分、2 美分和 1 美分硬币。

正确性证明：引理，若 n 是正整数，则用 25 美分、10 美分、5 美分、2 美分和 1 美分等尽可能少的硬币找出的 n 美分零钱中，至多有 2 个 10 美分、至多有 1 个 5 美分、至多有 2 个 2 美分硬币、至多有 1 个 1 美分硬币，而且不能有 2 个 10 美分和 1 个 5 美分硬币、不能有两个 2 美分和 1 个 1 美分硬币。因此用 10 美分、5 美分、2 美分和 1 美分硬币找出的零钱不能超过 24 美分。如果有超过规定数目的各种类型的硬币，可以用等值的数目更少的硬币来替换。

假设有存在正整数 n 和某种找零方式硬币数小于使用贪心算法所得到的的硬币数，在这种方式中所得到的 25 美分硬币数 $q^{'}$ 一定等于贪心算法所得的对应的 q，贪心算法使用尽可能多的 25 美分硬币，所以 $q^{'} \leq q$，假如 $q^{'}$ 小于 q，需要在这种最优方式中用 10 美分、5 美分、2 美分和 1 美分硬币至少找出 25 美分零钱, 由引理，这不可行。同理可得这种方式下的其他面额硬币个数和贪心算法硬币个数相同。

Q2.(25 分) 给定一个非负整数数组 $A[1:n]$, 每次可以选择一个满足：

$1 \leq l \leq r \leq n$，$min(a_l,\ a_{l+1},\ ...,\ a_r) > 0$ 的区间 $[l, r]$ 进行一次操作，使得 a_l，a_{l+1}，...，a_r 减 1。比如，$n = 5$，$A[1:5] = [1, 2, 3, 4, 5]$，先后选择 $[1, 5]$，$[2, 4]$ 执行操作后，A 数组变成 $[0, 0, 1, 2, 4]$。下一次操作便不可以选择 $l = 1$，2。

问最少需要多少次操作，使得 A 数组变为全 0 数组。请设计一个时间复杂度 $O(n)$ 的算法求解该问题，**给出伪代码和算法思路**、**复杂度说明**，并**证明算法的正确性**。

solution:

算法思路与正确性证明：从 1 到 n 遍历数组，如果后一个数比前一个数大，则答案加上两者差值；否则什么都不做。这是因为：首先对于 $A[1]$，一定要进行 $A[1]$ 次操作将其变成 0。而对于新的 $A[i]$，如果 $A[i]$ 比前一个数小，那么前一个数操作时顺带把 $A[i]$ 也带上就可以了，无需增加操作数；如果 $A[i]$ 比前一个数大，那么前一个数操作时只能连带着将 $A[i]$ 减小至 $A[i]-A[i-1]$，所以还需额外的 $A[i]-A[i-1]$ 次操作。由于只遍历了数组一次，所以时间复杂度是 $O(n)$ 的。

伪代码：

```
MIN-OPERATORS(A)
    operators = A[1]
    for i in [2, n]
        if A[i] > A[i-1]
            operators += A[i] - A[i-1]
    output operators
```

Q3. (30 分) 有一本词典，其中有 n 种单词，各个单词的出现次数分别为 a_1，a_2，...，a_n 。请推广赫夫曼编码到 k 位，使之能生成 k 进制的码字，并用其将词典中的每个单词替换为一个互不为前缀的 k 进制串，使得替换后的词典总长度最小，输出替换后的最小词典总长度（词典总长度定义为 $\Sigma_{i=1}^{n} a_i s_i$，s_i 是最终分配给 a_i 的编码长度）。**给出伪代码和算法思路**，不必证明算法的正确性。

solution: 算法思路：每次选择频率最小的 k 个单词合并即可。为了效率，可以维护一个优先队列用来快速查找最小的 k 个值。要注意的是，如果直接这样做，最后一次合并可能剩下小于 k 个节点，这显然不是最优的。考虑到每次操作会将节点数减少 $k-1$，所以应该一开始不断补 0 节点直到节点数满足 $n\%k-1=1$。

伪代码:

```
MIN-LENGTH(a)
    length = 0
    priority_queue = empty
    for i in [1, n]
        priority_queue.push(a[i])
    while priority_queue.size % (k-1) != 1
        priority_queue.push(0)
    while priority_queue.size >= k
        frequency = 0
        for i in [1, k]
            frequency += priority_queue.top()
            priority_queue.pop()
        length += frequency
        priority_queue.push(frequency)
    output length
```

Q4. (25 分)

1. 假定我们对一个数据结构执行一个由 n 个操作组成的操作序列，当 i 严格为 2 的幂时，第 i 个操作的代价为 i，否则代价为 1。使用聚合分析确定每个操作的摊还代价。

2. 用核算法重做第一题。

3. 使用势能法重做第一题。

solution:

1. 总代价 $T(n) = \Sigma_{i=1}^{n} f(i)$，其中 $f(i) = \begin{cases} i, i = 2^k \\ 1, else \end{cases}$

于是 $T(n) = n + \Sigma_{k=1}^{[log_2 n]} 2^k - 1 = n - [log_2 n] - 2 + 2^{[log_2 n]+1} = O(n)$，摊还代价为 $\frac{T(n)}{n} = O(1)$。

2. 对于正常操作，存储 3 的费用，支付 1 的费用；对于 2^k 的幂的操作，存储 3 的费用，支付 2^k 的费用。这样每个 i 的摊还代价 c_i' 都是 3。而对于每个 k，$2^{k-1} + 1$ 到 2^k 的操作中一共存储了 $3 * 2^{k-1} = 2^{k-1} + 2^k$ 的费用，支

付了 $2^{k-1}-1+2^k$ 的费用，所以信用始终是非负的，摊还代价为 $O(1)$。

3. 定义势能函数 $\Phi(i)=\begin{cases} 0, i=0 \\ 2(i-2^{[log_2 i]}), i \geq 1 \end{cases}$

$\Phi(n)-\Phi(0)=2(n-2^{[log_2 n]}) \geq 2(n-2^{log_2 n})=0$

当 $n=1$ 时，$c'=1+2(1-1)=1$

当 $n=2^k(k \geq 1)$ 时，$2^{[log_2 n]}=n, 2^{[log_2(n-1)]}=\frac{n}{2}$，$c'=n+2(n-n)-2((n-1)-\frac{n}{2})=2$

当 $n!=2^k(k \geq 0)$ 时，$2^{[log_2 n]}=2^{[log_2(n-1)]}$，$c'=1+2n-2(n-1)=3$

故摊还代价为 $O(1)$。