CS222 Homework 4

Exercises for Algorithm Design and Analysis by Li Jiang, 2016 Autumn Semester

5140309507 林禹臣 yuchenlin@sjtu.edu.cn

1. Given a non-empty integer array, find the minimum number of moves required to make all array elements equal, where a move is incrementing a selected element by 1 or decrementing a selected element by 1.

   You may assume the array's length is at most 10,000.

   Input:

   int A[]: the input array.

   int N: length of A.

   Output:

   int minMoves.

   Solution. Let us formulate this problem first. Function $F(T)$ to indicate the total movements when the final equal number is T. So we have:

$$F(T) = \sum_{i=1}^{N} |x_i - T|$$

   The problem is to find the minimum of the $F(T)$. This reminds me of one of the most important properties of median: **Median minimizes the sum of Absolute Deviations.** I would like to prove it later. Now if we all agree with the statement, what we should do next is very simple:

   Step 1. Find the median M of the array.
   Step 2. Calculate $F(M)$.
   Step 3. Return $F(M)$ as the final result.

   The time complexity of *Step 1* can be $O(n)$ with a divide-and-conquer strategy, which I have mentioned in the *Assignment 2*. The time complexity of *Step 2* is obviously $O(n)$ and *Step 3* is $O(1)$. Therefore, the total time complexity is $O(n)$.
   Now, let us prove the above-mentioned statement that Median minimizes the sum of Absolute Deviations.

   First of all, I would like to point out some wrong or weak statements. Some say that we must choose the final T from the elements in the array to minimize the $F(T)$, which is definitely wrong for that if the length is even and then we can choose any number between the middle two elements as the final T. Also, some say that choosing average can cause wrong output, then median is the optimal choice. It is just a guess, not a mathematical proof.

   Following are my two proofs.

Proof 1. Here, I would firstly use derivatives to illustrate why median is optimal choice and provide another proof with basic maths. We all know:

$$\frac{d|x|}{dx} = sgn(x)$$

Thus,

$$\frac{dF}{dx} = \sum_{i=1}^{N} sgn(x_i)$$

We should notice that in derivative is 0 only if the number of positive elements is equal to the number of negative elements. Meanwhile, $x = median$ can make sure that the number of elements which are less than median and the number of elements which are greater than median is equal. Plus, if $x < median$ and then the derivative is negative; if $x > median$ and then the derivative is positive.
Thus, median is an optimal solution.

Proof 2. is as follows: Suppose the length is odd and $T \leq M$. We can conclude that:

$$x_1 \leq x_2 \leq ... \leq x_s \leq ... \leq T \leq ... \leq x_{t-1} \leq x_t = M \leq x_{t+1} \leq ... \leq x_N \quad (t = \frac{n-1}{2})$$

$$\begin{aligned}
F(T) &= \sum_{i=1}^{N}(x_i - T) \\
&= \sum_{i=1}^{s}(T - x_i) + \sum_{i=s+1}^{N}(x_i - T) \\
&= [\sum_{i=1}^{t-1}(T - x_i) - \sum_{i=s+1}^{t-1}(T - x_i)] + [\sum_{i=s+1}^{t-1}(x_i - T) + 0 + \sum_{i=t+1}^{N}(x_i - T)] \\
&= [\sum_{i=1}^{t-1}(T - M + M - x_i) - \sum_{i=s+1}^{t-1}(T - x_i)] + [\sum_{i=s+1}^{t-1}(x_i - T) + \sum_{i=t+1}^{N}(x_i - M + M - T)] \qquad (1) \\
&= \sum_{i=1}^{t-1}(M - x_i) + (t-1)(T - M) + 2\sum_{i=s+1}^{t-1}(x_i - T) + \sum_{i=t+1}^{N}(x_i - M) + (n-t)(M - T) \\
&= \sum_{i=1}^{t-1}(M - x_i) + \sum_{i=t+1}^{N}(x_i - M) + 2\sum_{i=s+1}^{t}(x_i - T) + (n - 2t + 1)(M - T) \\
&= \sum_{i=1}^{t-1}(M - x_i) + \sum_{i=t+1}^{N}(x_i - M) + 2\sum_{i=s+1}^{t}(x_i) - 2(t-s) * T
\end{aligned}$$

Thus, if we want to minimize the $F(T)$, we just need to maximize T, which means when $T = M$ we can get the minimum of $F(T)$.It is similar when $T \geq M$ and we can insert a certain number between the middle two numbers to make the array odd.

□

2. Given a string that consists of only uppercase English letters, you can replace any letter in the string with another letter at most k times. Find the length of a longest substring containing all repeating letters you can get after performing the above operations.

   Note:Both the string's length and k will not exceed 10^4.

   Input:

   string s;

   int k;

   Output:

   return the length of the longest substring.

   Solution. Intuitively, if there is no restraint k, we will first find out the most common char and replace all other char to the most common one to achieve the longest substring. Thus, the times of replacement will be $n - t$, assuming that the most common char occurs t times.

   Now we have a restriction that the time of replacement should be less than or equal to k. Accordingly, we can conclude that if the original substring of the final result substring is $s[L, H]$ then we have $H - L - t \leq k$.

   Thus, our task is to find the largest $H - L$ satisfying hat $H - L - t \leq k$. It is a typical Sliding Window Problem. We can consider $[L, H]$ is a window which satisfies the condition. We always intend to expand this window and return the largest length as the output result. Specifically, we just iterate the H from 0 to N, and in each loop we try to keep expand the windows according to the condition and if the condition cannot be satisfied we just need to increase our L.

   As to the time complexity, we can find that it's evidently $O(n)$ for it only have one layer loop.
   Python code is as follows:

```python
def characterReplacement(self, s, k):
    L = H = 0
    counter = collections.Counter()
    for H in range(1, len(s)+1):
        counter[s[H-1]] += 1
        t = counter.most_common(1)[0][1]
        if H - L - t > k:
            counter[s[L]] -= 1
            L += 1
    return H - L
```

□

3. You are given an array x of n positive numbers. You start at point (0,0) and moves x[0] metres to the north, then x[1] metres to the west, x[2] metres to the south, x[3] metres to the east and so on. In other words, after each move your direction changes counter-clockwise.

   Write a one-pass algorithm with O(1) extra space to determine, if your path crosses itself, or not.