Problem 22 - Names Scores

Gautam Manohar

23 June 2018

This document originally appeared as a blog post on my website. Find it at gautammanohar.com/euler/22.

1 Problem Statement

You are given around five-thousand names. Begin by sorting them by alphabetical order. Then working out the alphabetical value of each name, multiply this value by the alphabetical position of the name in the list.

For example, suppose `COLIN` is the 983rd element of a sorted list of names. It is worth $3 + 15 + 12 + 9 + 14 = 53$, so this name has a score of $53 \cdot 983 = 49714$.

Give the name score of the given names.

2 My Algorithm

We need two things to solve this problem: a way to get the score of a letter and a way to get the alphabetical position of a name in the list. We can make a string `ABCDEFHGHIJKLMNOPQRSTUVWXYZ` containing all the letters of the alphabet. For each letter in a given name, we find the index of that letter in our string and add one; this is the score of that letter. We sum the scores and multiply it by the alphabetical position of a name, which is `i + 1`, where `i` is its index in the sorted list. In Python, sorting a list of strings is done in alphabetical order.

We sort the list of n names in $O(n \log n)$ time, and we find the index of a name in the list by linearly searching it in $O(n)$ time. The alphabetical score is computed in $O(26) = O(1)$ time. This solution has time complexity $O(n \log n + Qn)$.

3 Other Solutions

Because we sort the list of names, we could use binary search to find the alphabetical position of a given name. This would improve our time complexity to $O(n \log n + Q \log n)$. In the HackerRank problem, $Q < n$, but even if Q were greater than n, then some names would be queried more than once. In this case, instead of recalculating the score, we could cache each score as it is calculated. And so binary search reduces the complexity of this solution to just the sorting time, $O(n \log n)$.