Problem discussion

February 3, 2016

Hamming distance (D)

Find any string that minimizes the sum of Hamming distances to 3 given strings.

The hamming distance is the number of characters that are different between 2 strings.

D: Solution

Each character contributes independently to the sum, so at each character we need to choose the best possible option. It’s clear that we’ll choose one of the characters already present in one of the 3 strings.

If all 3 have different characters at position i, it won’t matter which one we chose, since the cost will be +2 no matter what.

But if at least two of them agree on one (have the same character at that position) we’ll chose that one.

Secret information (H)

Reach target bit string from initial bit string through minimum number of steps, where a step is flipping the bits in a substring.

example: 010101000 to 010010001

H: Solution

Observation: The optimal solution can be made from intervals with no overlap. We can ignore the overlapping parts, because flipping two times returns the substring to the original form.

0101010 -> 0010010
0010010 -> 0001100 has the same effect as:
0101010 -> 0001010
0001010 -> 0001100

Solution: Go once through both strings and flip all continuous substrings where the values are different.

The best statement (J)

You have the following function:

```
F(IntArray a1, IntArray a2, int K):
    for i in range(a1)
        if(a1[i] >= K) return a2[i]
```

With the 2 arrays fixed, find a K such that it maximizes the function.

J: solution

We could actually implement the function, and chose each possible K, and then get the maximum, but this will not work since there are at most 10^9 possible K.

To solve this we need to look at what positions are possible to be selected. If we're at position i, we will be able to select it if and only if there's a K that is smaller than or equal to A[i], but bigger than all A[j], with j < i;

This means that we can select the ith element, if A[i] is a the maximum in the interval A[j], with j <= i.

So the solution follows: for all A[i] that are maximum so far, choose the biggest B[i].

Similar Strings (B)

Number of pairs of words that are equal through an alphabet bijection.

An alphabet bijection is a one to one mapping from {a,b,...z} to {a,b,....z} (different letters are mapped to different values).

B: solution

- How to determine if 2 strings are ‘similar’ (example: DDKSD and ZZXYH)
  - go through both strings
  - if new character in first string, add mapping: D→ Z, K → X, S→ Y
  - if character already appeared, check that the mapping is correct: D→ Z ok, D→ H incorrect
- Wrong approach: Compare every pair → not time efficient
- Correct approach: Find a transformation that turns similar strings in equal strings and count the occurrences of each transformed string
  - efficient ways for counting include maps and sorting
- How to find the transformation:
  - Since we are basically looking at the first occurrence of each character, we can map the first new character to ‘A’, second new character to ‘B’ etc.
  - DDKSD → AABCA, ZZXYH → AABCD so they are not similar
  - TETE → ABAB, MAMA → ABAB so they are similar