DigitStringDiv1 - SRM 741, D1, 250-Pointer

Austin Jones

November 23, 2021

University of Tennessee - Knoxville

Overview

Problem Intro

Statement and Constraints

```
class DigitStringDiv1 {
  public:
    long long count(string s, int x);
};
```

- Given: S, string containing characters from '0' through '9' and is at most 47 characters in length.
- Given: X, integer between 0 and 777,444,111.
- A substring of S is a string that results when you erase some subset of indices of S.
- A substring of S does not start with '0'.
- e.g. S is "12356", "13" and "256" are substrings. "61" is not.
- **Count the number of substrings of S that are greater than X.**

Example 1

$$S = 101; X = 9 \rightarrow Answer = 3$$

- 101

Example 1

$$S = 101; X = 9 \rightarrow Answer = 3$$

- 101
- 10

Example 1

$$S = 101; X = 9 \rightarrow Answer = 3$$

- 101
- 10
- 11

Example 1

$$S = 101; X = 9 \rightarrow Answer = 3$$

- 101
- 10
- 11
- 1

Example 1

$$S = 101; X = 9 \rightarrow Answer = 3$$

- $101 > 9$
- $10 > 9$
- $11 > 9$
- $1 \leq 9$

Example 1

$$S = 101; X = 9 \rightarrow Answer = 3$$

- $101 > 9$ Here!
- $10 > 9$ Here!
- $11 > 9$ Here!
- $1 \leq 9$

Example 2

$$S = 471; X = 47 \rightarrow Answer = 2$$

- 471

Example 2

$$S = 471; X = 47 \rightarrow Answer = 2$$

- 471
- 47

Example 2

$$S = 471; X = 47 \rightarrow Answer = 2$$

- 471
- 47
- 41

Example 2

$$S = 471; X = 47 \rightarrow Answer = 2$$

- 471
- 47
- 41
- 71

Example 2

$$S = 471; X = 47 \rightarrow Answer = 2$$

- 471
- 47
- 41
- 71
- 4

Example 2

$$S = 471; X = 47 \rightarrow Answer = 2$$

- 471
- 47
- 41
- 71
- 4
- 7

Example 2

$$S = 471; X = 47 \rightarrow Answer = 2$$

- 471
- 47
- 41
- 71
- 4
- 7
- 1

Example 2

$$S = 471; X = 47 \rightarrow Answer = 2$$

- $471 > 47$
- $47 \leq 47$
- $41 \leq 47$
- $71 > 47$
- $4 \leq 47$
- $7 \leq 47$
- $1 \leq 47$

Example 2

$$S = 471; X = 47 \rightarrow Answer = 2$$

- $471 > 47$ Here!
- $47 \leq 47$
- $41 \leq 47$
- $71 > 47$ Here!
- $4 \leq 47$
- $7 \leq 47$
- $1 \leq 47$

Example 2

$S = 471; X = 47 \rightarrow Answer = 2$

- $471 > 47$ Here!
- $47 \leq 47$ *Notably Not* Here!
- $41 \leq 47$
- $71 > 47$ Here!
- $4 \leq 47$
- $7 \leq 47$
- $1 \leq 47$

Some symbology

- Let i, j be a value 0-indexing into strings.
- Let X_i be the value of X at index i.
- Let S_i be the value of S at index i.
- Let X_D be the count of digits in X.
- Let S_D be the count of digits in S.
- Let X_S be stringified X.

Slow, Dumb Solution

Power Set Enumeration

- Create the power set of the S string.
- Process sets, based on the digit counts of S_i and X.
  - Fewer digits than X, discard.
  - More digits than X, it must be greater.
  - The same number of digits as X, convert S_i to an integer and compare.
- Return count of greater substrings.

Running Time

- Create the power set: $O(S_D * 2^{S_D})$
- Process elements of power set: $O(S_D) * O(2^{S_D})$
  - Fewer digits, discard: $O(1)$
  - More digits, it must be greater: $O(1)$
  - Same number of digits, convert S_i and compare: $O(S_D)$
- Total:
  $O(S_D * 2^{S_D}) + O(S_D * 2^{S_D}) = O(S_D * 2^{S_D})$

Plank’s Fast, Smart Solution

Symbology Refresher

- Let i, j be a value 0-indexing into strings.
- Let X_i be the value of X at index i.
- Let S_i be the value of S at index i.
- Let X_D be the count of digits in X.
- Let S_D be the count of digits in S.
- Let X_S be stringified X.

The Algorithm

- Count all substrings of S with more digits than X.
- Count all substrings of S with the same number of digits as X.

Digits Greater

For each index in S s.t. S_i is non-zero, count all the substrings starting at that index with more digits than X.

- At all indices greater than the number of digits in X perform:

$$CountGreater(S_i) = \sum_{j=X_D}^{i} \binom{i}{j}$$

- The sum goes to i as S_i is fixed.
- Likewise, the sum starts at X_D - one digit is already chosen.

Digits Greater: Running Time

- For all S_i s.t. $i > X_D$: $O(S_D - X_D)$ [1]
- Look at all substring lengths from X_D to i: $O(S_D - X_D)$
- i choose j: $O(j) \rightarrow O(S_D)$
- Total: $O(S_D - X_D) * O(S_D - X_D) * O(S_D) = O((S_D)(S_D - X_D)^2)$

[1] The "$-X_D$" comes back up later.

Digits Equal

For each index in S, look at all substrings equal in length to X. Recursively enumerate the rest of S to a depth of X_D.

Define a routine, $CountEqual(i, j)$, that will compare S_i and X_j.

- On success, spawn $CountEqual(k, j+1)$ for all k from i to S_D.
- Success of $CountEqual(i, X_D)$ returns 1 as a valid substring has been found.
- Success is $S_i \geq X_j$ for $j \neq X_D$
- Success is $S_i > X_j$ for $j = X_D$

Warning: Hand-waving Ahead!

Digits Equal: Running Time

Let $T_0(d)$ be the time/work to run *CountEqual*(i, d). $T_0(0)$ is the exit a condition. $T_0(0)$ is $O(1)$ multiplied by the levels that reach it.

$$\begin{aligned}
T_0(X_D) &= S_D T(X_D - 1) \\
&= S_D S_D T(X_D - 2) \\
&= {S_D}^3 T(X_D - 3) \\
&= {S_D}^{X_D} T(0) \\
&= O({S_D}^{X_D})
\end{aligned}$$

Let $T(S, X)$ be the time/work to run *CountEqual* for the full string. To get all the sums, S_D runs of $T(X_D)$ are required, yielding:

$$T(S, X) = O({S_D}^{X_D+1})$$

This calculation assumes all recursions reach the exit condition. Which would never happen.

Total Running Time/Comparison

The total runtime for this algorithm is:

$$T_{smart}(S, X) = O({S_D}^{X_D+1} + (S_D)(S_D - X_D)^2)$$

Great improvement over:

$$T_{dumb}(S, X) = O(S_D * 2^{S_D}) + O(S_D * 2^{S_D}) = O(S_D * 2^{S_D})$$

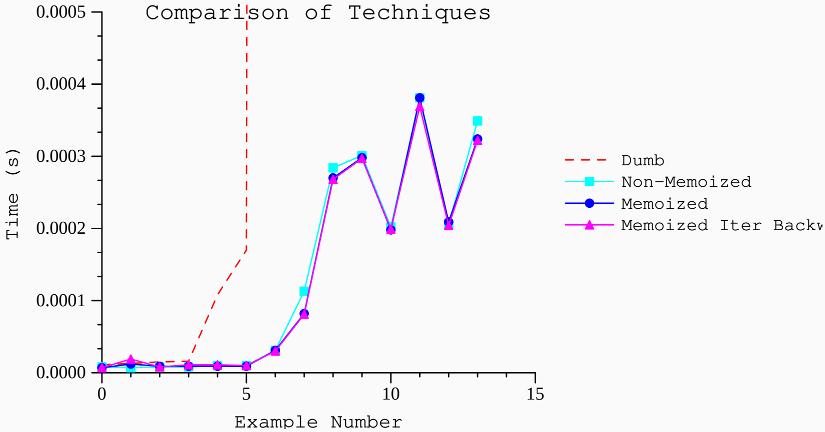
Findings

Speeding up Digits Equal

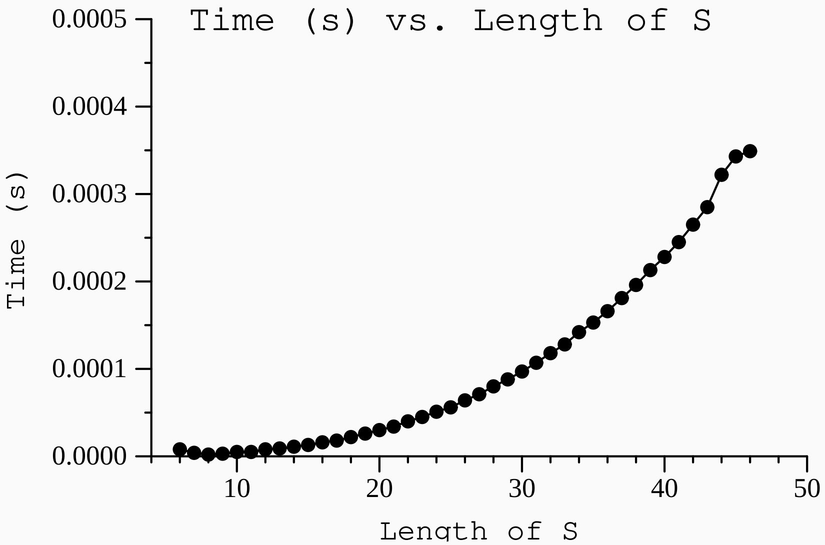
- Memoization:
  $CountEqual(i, j)$ can be memoized on i and j. Making recurring recursion calls free.
- Reverse Iteration:
  The thought here is that iterating backwards would fill up the Memoization Cache faster.
- Early Exit:
  If in a call of $CountEqual(i, j)$, $S_i > X_j$ is found, return:

$$\binom{S_D - i}{X_D - j}$$

Different Methods to the Problem

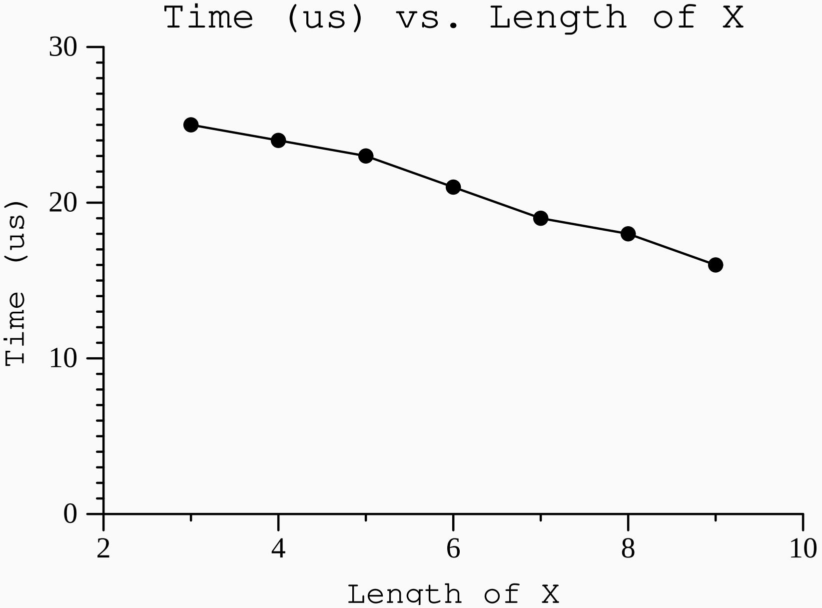


Time vs S



Time (s) vs. Length of S
0.0005
0.0004
0.0003
0.0002
0.0001
0.0000
Time (s)
10
20
30
40
50
Length of S

Time vs X



Wrapping Up

Machine Specs

MacBook Pro (15-inch, 2016)

- CPU: Intel i7-6700HQ (8) @ 2.60GHz
- GPU: AMD Radeon Pro 450
- Memory: 16384 MiB

How did Topcoder Do?

- Problem Given in Topcoder: November, 2018
- Competitors who opened the problem: 99
- Competitors who submitted a solution: 82
- Number of correct solutions: 42
- Accuracy (percentage correct vs those who opened): 49.4%
- Average Correct Time: 24.59
- Best Time: 4:56

Thank You

Questions?