CS/ECE 374 Spring 2017 Lanxiao Bai (lbai5)
Homework 5 Problem 1 Renheng Ruan (rruan2)
Ho Yin Au (hoyinau2)

Let $w \in \Sigma^*$ be a string. We say that $u_1, u_2, \ldots, u_h$ where each $u_i \in \Sigma^*$ is a valid split of w iff $w = u_1 u_2 \ldots u_h$ (the concatenation of $u_1, u_2, \ldots, u_h$). Given a valid split $u_1, u_2, \ldots, u_h$ of w we define its ℓ_3 measure as $\sum_{i=1}^{h} |u_i|^3$.

Given a language $L \subseteq \Sigma^*$ a string $w \in L^*$ iff there is a valid split $u_1, u_2, \ldots, u_h$ of w such that each $u_i \in L$; we call such a split an L-valid split of w. Assume you have access to a subroutine IsStringInL(x) which outputs whether the input string x is in L or not. To evaluate the running time of your solution you can assume that each call to IsStringInL() takes constant time.

Describe an efficient algorithm that given a string w and access to a language L via IsStringInL(x) outputs an L-valid split of w with minimum ℓ_3 measure if one exists.

Solution: First of all, to reach the possible minimal measure ℓ_3, we want to know in what occasion it is the least. And here we claim that

Lemma 1: If there's a set of positive integers $A = \{a_n : n\mathbb{N}\}$

$$\sum_{i=1}^{n} a_i^3 \leq (\sum_{i=1}^{n} a_i)^3$$

for all $n \in \mathbb{N}$.

Proof: We can prove this lemma by mathematical induction.

Base case, when $n = 1$,

$$\sum_{i=1}^{1} a_i^3 = a_1^3 = (\sum_{i=1}^{1} a_i)^3$$

Inductive Hypothesis, suppose for all $n \leq k$,

$$\sum_{i=1}^{n} a_i^3 \leq (\sum_{i=1}^{n} a_i)^3$$

Then when $n = k + 1$,

$$\begin{aligned}
\sum_{i=1}^{k+1} a_i^3 &= \sum_{i=1}^{k} a_i^3 + a_{k+1} \\
&= \sum_{i=1}^{k} a_i^3 + \sum_{i=1}^{1} a_{k+1} \\
&\leq \sum_{i=1}^{k} a_i^3 + \sum_{i=1}^{1} a_{k+1}^3 \\
&\leq (\sum_{i=1}^{k} a_i)^3 + (\sum_{i=1}^{1} a_{k+1})^3 \\
&< (\sum_{i=1}^{k} a_i + a_{k+1})^3 = (\sum_{i=1}^{k+1} a_i)^3
\end{aligned}$$

Hence, we conclude that

$$\sum_{i=1}^{n} a_i^3 \leq (\sum_{i=1}^{n} a_i)^3$$

for all $n \in \mathbb{N}$.

Then, this lemma means that to reach the minimum ℓ_3 measure, we want to have each segment of string is the shortest valid string.

As a result, we can recursively split string into two valid segments in L until they cannot be split any more, of there are more than 1 way of splitting the string, ℓ_3 need to be compared to determine the optimum solution.

Hence, we define the recurrence function that returns the split of string with minimum ℓ_3:

$$\textsc{MinimumL3Split}(i, j) = \begin{cases} \textsc{null} & \text{if } j > n \\ w[i\,..\,j] + \textsc{MinimumL3Split}(j{+}1, j{+}1) & \text{if } j \leq n \text{ and IsStringInL}(w[i\,..\,j]) \\ \textsc{MinimumL3Split}(i, j+1) & \text{if } j \leq n \text{ and not IsStringInL}(w[i\,..\,j]) \end{cases}$$

So with the function, we need to compute $\textsc{MinimumL3}(1,1)$ if $|w| = n$.

We can memoize all function values in a three-dimensional array $\textsc{MinimumL3}[1\,..\,n][1\,..\,n]$. Then we see that each entry $\textsc{MinimumL3}[i, j]$ depends on its up $\textsc{MinimumL3}[i, j+1]$ and the entry on the diagonal of its row. Thus, we can fill the array from left to right, top to bottom. The recurrence gives us the following pseudocode:

```
Split(w[1..n]):
  MinimumL3[n,n] = Null
  for i ← n to 1
      for j ← n to i
          if IsStringInL(w[i, j])
              MinimumL3Split[i, j] ← w[i .. j] + MinimumL3Split(j + 1, j + 1)
          else
              MinimumL3Split[i, j] ← MinimumL3Split(i, j + 1)
  return MinimumL3Split[1, 1]
```

So, obvious the overall runtime of this algorithm is $O(n^2)$.

■