PS1 - Part 2

Joe Brown and Morgan Marks

January 29, 2014

0 : SHORTEST SUPERSTRING

0.1 PROVIDING A 2-APPROXIMATION FOR GREEDY

There is a family of strings for which lucky greedy gets "locked out," causing a $\frac{3}{2}$ OPT solution. The fundamental building blocks of this family are:

L : The **L**eft-side lockout character

R : The **R**ight-side lockout character

S : The **S**eparator character

O : The repeated character.

To generate the simplest version of the lockout, we use three strings:

$$\begin{aligned} &LO^k \\ &O^kR \\ &O^{k-1}SO^{k-1} \end{aligned}$$

$$\begin{aligned} Greedy: &\quad LO^kRO^kSO^{k-1} &&\lim_{k\to\infty} = 3k \\ OPT: &\quad LO^kSO^kR &&\lim_{k\to\infty} = 2k \end{aligned}$$

Greedy chooses to merge the first two strings, because they have a value of k, then ends up with the middle string locked out. OPT instead chooses to sandwich the third string.

To further hinder Greedy, we have to add another layer of abstraction. In essence, we multipy the effect of the lockout by multiplying the O^kS portion of the strings.

$$
\begin{array}{l}
L(O^kS)^nO^k \\
(O^kS)^nO^kR \\
O^{k-1}S(O^kS)^{n-1}O^{k-1}
\end{array}
$$

$$
\begin{array}{rll}
Greedy: & L(O^kS)^nO^kRO^{k-1}S(O^kS)^{n-1}O^{k-1} & \lim_{n\to\infty} = 2k \\
OPT: & L(O^kS)^{n+1}O^kR & \lim_{n\to\infty} = k
\end{array}
$$

As n becomes very large, Greedy is a 2·OPT approximation. □

0.2 Exceeding a 2-approximation for Greedy

In all of our sketching and stretching, papers spilling from the couch and the coffee table to the floor, we are left ragged and empty-handed. The amazement at our complete lack of progress resonates in the empty $\frac{11}{23}$, and we are left with the solemn question of how the *(redacted)* can this upper bound of two be so hard to prove?

0.3 Alphabet Restriction Contradiction

The general Shortest Superstring problem (for any alphabet size) can be α-approximated in polynomial time.
Given$_1$: ∃ polynomial-time α-approximation for Shortest Superstring *for instances over a binary alphabet.*
Given$_2$: Any alphabet Σ can be binary encoded with the following formula:

replace the i_{th} character in Σ with $0^i(01)^{k+1-i}1^i$, where k := |Σ|.

By using this encoding to produce Σ', no distinct character in Σ' can overlap at all with any other distinct character in Σ'
The idea shares a few traits of the fundamental building blocks we use in 0.1

- 0*… : The Left-side lockout character
- …1* : The Right-side lockout character
- 01* : The Separator/Uniqueness character

By locking out overlap by anything but identical characters, we are effectively running a multicharacter alphabet Σ emulator in Σ'. Since we know that the algorithm is an α-approximation in (0,1)*, we know that this emulation is also α-approximable. □

1 : O(1)-APPROXIMATION FOR NUMBER OF MOVES IN CHUNK SORTING

The goal of Chunk-Sorting can be expressed as:

> "To create a single sorted chunk"

When you are given π, OPT describes the minimum number of chunk-moves required to turn π into a single sorted chunk.
A helpful first way to look at π is as a set of n pre-sorted chunks, ranging from singletons to $|\pi|$, and the goal of any algorithm is to immediately or indirectly get rid of of (merge) them.

Whatever sorcery guides the hand of the OPT algorithm, the maximum reduction of these sorted chunks that can be attained in one move is 3. This particular type of move happens to be the only move detailed in the Problem Set.

1.1 $\frac{1}{3}$-APPROXIMATION ALGORITHM Φ

1. Examine all elements in π.
2. Group elements into chunks, c_a through c_k, of consecutive integers.
   a) For an arbitrary c_i in π, find a remaining chunk that either continues the consecutive chain of c_i, or leads up to the consecutive chain of c_i, and move c_i there.
   b) if there is no suitable match for c_i, choose another arbitrary chunk and go back to a)
   c) when you move a chunk, go back to 1. At this point, you are effectively recursing on a more-solved problem.
   d) if you cannot find any way to move any chunk beneficially, you are done. π is sorted.

Φ reduces the number of remaining chunks in π by at LEAST one (1) for every chunk-move
OPT reduces the number of remaining chunks by at MOST three (3) per chunk-move.

Φ is a $\frac{1}{3} \cdot$ OPT approximation. □