Shortest Common Superstrings.

1 Introduction

In this project, we address the shortest common superstring problem(SCS), which is as follows:

Given a set of strings $S = \{w_1, ..., w_n\}$ over a finite alphabet, output the shortest string s such that for all $w \in S, w$ is a substring of s.

We study a reduction of the problem to a weighted hamiltonian path problem, Ukonnen's greedy algorithm, design an $O^*(2^n)$ exact algorithm and a boosting procedure which tries to improve the quality of solution outputted by the greedy algorithm.

In the next section we look at some definitions and notation that we use in the report.

2 Definitions and Notations

We assume any two strings s and t, such that s is not a substring of t, they can be written as $s = uv$ and $t = vw$ such that u and w are non empty. Here are some definitions that we use in the report.

- *Superstring*: w is called as superstring of S if for all $x \in S$, x is a substring of w.
- *Overlap*: Overlap of s and t is the length of v (denoted by $|v|$). It is denoted by $ov(s,t) = |v|$.
- *Prefix*: Prefix of s w.r.t to t is denoted by $pref(s,t) = u$.
- *Distance*: Distance from s to t is denoted by $d(s,t) = |pref(s,t)|$.
- *Substring free set*: S is *substring free* if for all distinct $x, y \in S$, x is not a substring of y.

The string uvw can be written as $pref(s,t)t$ and it has length $d(s,t) + |t|$.

3 Preprocessing

If there are two strings x and y in S, such that x is a substring of y, then for any superstring w of $S \setminus \{x\}$, x is a substring of w. Therefore, x can be removed from S. This motivates the following preprocessing step:

If there are strings x and $y \in S$ such that x is a substring of y, remove x from S. We repeat this preprocessing step till the resultant set of strings is substring free.

From this point onwards, we assume that the given set of strings is substring free.

Given a permutation $\sigma : \{1, 2...n\} \to \{1, 2...n\}$, we define the shortest common superstring of S w.r.t to permutation σ, denoted by $w_\sigma = \langle w_{\sigma(1)}, ..., w_{\sigma(n)} \rangle$, inductively as follows:

$$\langle w_{\sigma(1)}, w_{\sigma(2)} \rangle = pref(w_{\sigma(1)}, w_{\sigma(2)}) w_{\sigma(2)} \tag{1}$$

$$\langle w_{\sigma(1)}, ..., w_{\sigma(n)} \rangle = \langle \langle w_{\sigma(1)}, ..., w_{\sigma(n-1)} \rangle, w_{\sigma(n)} \rangle \tag{2}$$

In the next two observations, we observe that each superstring induces a permutation on S and each permutation has its own smallest supuerstring that generates the permutation.

Observation 1. *Given a superstring $w = a_1a_2...a_m$ of S, we can find a permutation σ on $\{1, ..., n\}$ such that for $i \neq j$, if $w_{\sigma(i)} = a_ka_{k+1}...a_{k+|w_{\sigma(i)}|-1}, w_{\sigma(j)} = a_la_{l+1}...a_{l+|w_{\sigma(i)}|-1}$ and $\sigma(i) < \sigma(j)$, then $k < l$.*

Proof. We do induction on cardinality of S.

For $|S| = 1$, trivially, $\sigma(1) = 1$. Assume that it is true for substring free sets of strings of cardinality less than n for $n \geq 2$. Therefore, for an arbitrary set $S = \{w_1, ..., w_n\}$ with a superstring $w = a_1...a_m$. w can be written as sw_nt for some strings s and t. If t is empty, then by inductive hypothesis, sw_n induces σ on $\{1, ..., (n-1)\}$ and we can extend σ by assigning $\sigma(n) = n$. Otherwise, the string sw_n is a superstring of $\{w_{i_1}, \ldots, w_{i_k}, w_n\}$ for $k < n$ and w_nt is a superstring of $\{w_{i_{k+1}}, \ldots, w_{i_{n-1}}, w_n\}$. And by induction hyothesis, sw_n induces permutation σ_1 on $\{w_{i_1}, \ldots, w_{i_k}, w_n\}$ and σ_2 on $\{w_{i_{k+1}}, \ldots, w_{i_{n-1}}, w_n\}$. Note that $\sigma_1(k+1) = n$ and $\sigma_2(1) = n$. we can define σ as follows: for $1 \leq i \leq k$, $\sigma(i) = \sigma_1(i)$, $\sigma(k+1) = n$ and for $k+2 \leq i \leq n$, $\sigma(i) = \sigma_2(i-1)$. □

Observation 2. *Given a superstring w on a S which induces an permutation σ on $\{1, ..., n\}$, $|w_\sigma| \leq |w|$.*

Proof. We do induction on the number of strings. If $n = 1$, then $w_\sigma = w_1$ and for any superstring w, $|w_\sigma| \leq |w|$. Let us assume that the proposition is true for $n = k$ where $k > 1$. □

4 Exact Algorithm

4.1 Dynamic Program over Subsets of Set of Strings.

Length of shortest common superstring on S is given by

$$|S| = \sum_{i=1}^{n} |w| - \max_{\pi \in P}\{\sum_{i=1}^{n-1} ov(\pi(i), \pi(i+1))\} \tag{3}$$

where P is the set of all permutations on $\{1, ..., n\}$. This is same as finding minimum weight hamiltonian path in the graph.As $d(u, v) = |u| - ov(u, v), \sum_{i=1}^{n} |w| - \max_{\pi \in P}\{\sum_{i=1}^{n-1} ov(\pi(i), \pi(i+1))\}$ is the weight of the minimum weight Hamiltonian path of G.
Let $OPT[A]$ be the maximum overlap sum of $A \subset S$ over all permutations on A.For $v \in A, OPT_v[A]$ denotes the maximum overlap sum of A over all permutations starting with v.

$$OPT_v[A] = \max_{u \in A \setminus v}\{ov(v, u) + OPT_u[A \setminus v]\}$$
$$OPT[A] = \max_{v \in A}\{OPT_v[A]\}$$

Therefore,

$$OPT[S] = \max_{v \in S}\{OPT_v[S]\}$$

$OPT[S]$ can be calculated by incrementally finding $\forall v \in A, OPT_v[A]$ over all subsets A of S. Total time taken is $O^*(2^n)$.

4.2 Ukonnen's Greedy Algorithm

4.2.1 Graph Formulation

The n-vertex graph $G = (V, E, w)$ with vertex set $V = \{v_1, \ldots, v_n\}$, edge set E and weight function $w \colon V^2 \to \mathbb{Z}^+ \cup \{0\}$ is a weighted, directed complete graph (i.e, between every ordered pair of vertices (u, v), there is a directed edge (u, v) in E) where w is determined by the set of strings S as follows.

$$w((u_i, u_j)) = \begin{cases} d(w_i, w_j) & i \neq j \\ 0 & \text{otherwise} \end{cases}$$

4.2.2 Ukonnen's Greedy Heurestic

Initially, $\mathcal{P} = \phi$.

1. In G, pick the largest weight edge (not yet already picked) in the partial solution $\mathcal{P}$ solution such that $\mathcal{P}$ can be extended to a hamiltonian path.

2. If no such edge remains, return $\mathcal{P}$ as the solution.Else, repeat step 1.

It is conjectured that this greedy algorithm gives a $2-$approximation algorithm and a bound for $4-$approximation has been proven.

4.3 Turbocharging the greedy solution

We present the following method in an attempt to improve the quality of the greedy solution which is also known as "turbocharging" in the literature.

The idea of the turbocharging is as follows. Given a greedy solution $w = \langle w_1, w_2, ..., w_n \rangle$, pick $1 \leq i < j \leq n$ and run the exact algorithm for finding shortest common superstring for the set of strings $\{w_i, ..., w_j\}$ with w_i and w_j as the starting and ending strings.The resulting string with superstring $w = \langle w_1, w_2, ..., w_i, w'_{i+1} ..., w'_{j-1}, w_j, ..., w_n \rangle$ will be atleast as good as the greedy solution.The procedure will improve the solution if $\langle w_i, ..., w_j \rangle$ is not the optimal superstring starting with w_i and ending with w_j in the greedy solution.

In our implementation, we take as a input, a parameter ℓ and then, partition S according to the sequence σ obtained by the greedy solution into n/ℓ sets $\{w_{\sigma(1)}, \ldots, w_{\sigma(\ell)}\}, \ldots, \{w_{\sigma(n-\ell)}, \ldots, w_{\sigma(n)}\}$. Then, we run the exact algorithm for each of these n/ℓ sets and finally concatenate them to get a super string. As we argued in the previous paragraph, the quality of the solution will not be worse than the greedy solution.

As the exact procedure takes exponential time, the parameter ℓ is much smaller compared to n.