

MA3259 Lecture 16

# Genome Mapping, Assembly and Sequencing Part V

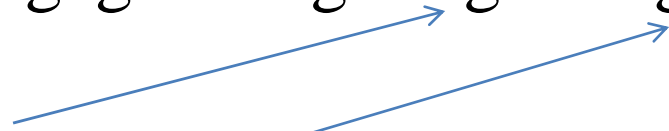
LX Zhang  
Department of Mathematics  
National University of Singapore  
matzlx@nus.edu.sg

# Shortest Superstring Problem

---

S: agcgttttttcgtttctgcaataggct

V: tgcaatag



S is a superstring of V;

S is a superstring of a set of strings if it is a superstring of all strings.

**Example** Let  $\mathcal{F} = \{agt, gta, act\}$ .

*gtaagtact, agtgtaact, agtact*

are superstrings of  $\mathcal{F}$ .

## Shortest Superstring Problem

INSTANCE: A set  $\mathcal{F}$  of  $k$  strings.

QUESTION: Find a shortest superstring of  $\mathcal{F}$ .

**Fact** Let  $\mathcal{F}$  is a set of  $k$  strings and let  $S \in \mathcal{F}$ . If  $S$  is a substring of another of  $\mathcal{F}$ , then, a superstring of  $\mathcal{F} - \{S\}$  is also one of  $\mathcal{F}$ .

In the following discussion , any string set is “substring free”.

**Definition:**

- (1) For any string  $s$ , let  $|s|$  denote the length of  $s$ .
- (2) For two strings  $x$  and  $y$ , let  $w(x, y)$  be the length of the longest match (overlap) of a suffix of  $x$  and a prefix of string  $y$ .

For two strings  $s_1 = a_1a_2 \cdots a_m$  and  $s_2 = b_1b_2 \cdots b_n$ . Let  $w$  be  $w(s_1, s_2)$ , the *length* of the longest overlap of a suffix of  $s_1$  and a prefix of  $s_2$ . Then,  $a_{m-w+i} = b_i$  for  $i \leq w$  and we define

$$s_1 \circ s_2 = a_1a_2 \cdots a_{m-w}b_1b_2 \cdots b_n = a_1a_2 \cdots a_m b_{w+1} \cdots b_n.$$

called **the overlap string of string  $s_1$  and  $s_2$ .**

$s_1$ :	a b a c d
$s_2$ :	a c d e f
$s_1 \circ s_2$ :	a b a c d e f

## Greedy Algorithm

INPUT: A non-empty substring-free set  $\mathcal{S}$  of strings.

REPEAT until  $\mathcal{S}$  contains just one string

    Select a pair of strings  $s$  and  $t$  that has the longest overlap;

    Remove  $s$  and  $t$  from  $\mathcal{S}$  and add  $s \circ t$  into  $\mathcal{S}$ .

Remarks: (1). When several pairs have the longest overlap at the start of a step, choose an arbitrary pair to merge.

(2). Merging strings in different order may give different output.

Example:

$S = \{aaaxbbb, bbbxccc, cccxddd, bbxaaax, ccxbbbx, ddxcccx\}$



Step 1: Merger bbxaaax and aaaxbbb

$\{bbxaaaxbbb, bbbxccc, cccxddd, ccxbbbx, ddxcccx\}$



Step 2: Merger ccxbbbx and bbbxccc

$\{bbxaaaxbbb, ccxbbbxccc, cccxddd, ddxcccx\}$



Step 3: Merger ddxcccx and cccxddd

$\{bbxaaaxbbb, ccxbbbxccc, ddxcccxddd\}$



Step 4: Merge the first and second strings

$\{bbxaaaxbbbcxbbbxccc, ddxcccxddd\}$



bbxaaaxbbbcxbbbxcccddxcxcxddd

## Greedy Algorithm

INPUT: A non-empty substring-free set  $\mathcal{S}$  of strings.

REPEAT until  $\mathcal{S}$  contains just one string

    Select a pair of strings  $s$  and  $t$  that has the longest overlap;

    Remove  $s$  and  $t$  from  $\mathcal{S}$  and add  $s \circ t$  into  $\mathcal{S}$ .

Remarks: (1). When several pairs have the longest overlap at the start of a step, choose an arbitrary pair to merge.

(2). Merging strings in different order may give different output.

Does the algorithm perform well?

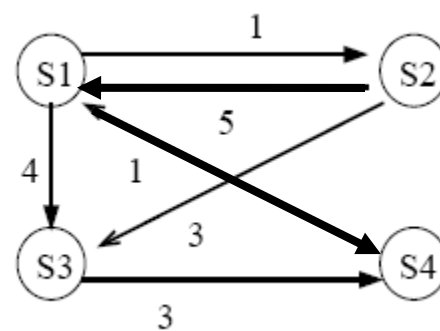
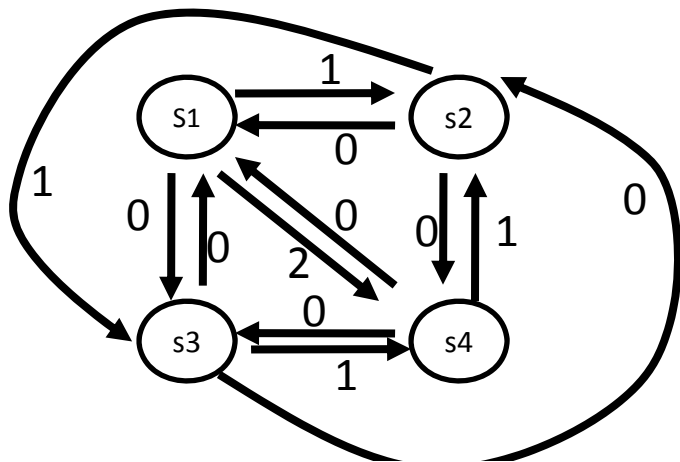
# Overlap graphs

**Definition** The overlap graph  $G(\mathcal{F})$  of a set  $\mathcal{F}$  of strings is a directed graph where each node corresponds to a string of  $\mathcal{F}$  and there is an arc from  $S_1$  to  $S_2$  of weight  $w(S_1, S_2)$  for any  $S_1$  and  $S_2$ .

**Example 1.** The overlap graph of

$$\{S_1 = tagca, S_2 = aggg, S_3 = gtaaac, S_4 = caga\}$$

is the graph (a) in the figure.



$$\{S_1 = cbadef, S_2 = fcbade, S_3 = adefcd, S_4 = fcdafeb\}$$

Recall that a directed path  $P$  is a list of nodes in which two neighboring nodes are connected by a directed edge.

**Fact 1:** *Each directed path*

$$P = s_1 s_2 \cdots s_k$$

*in the overlap graph  $G(\mathcal{F})$  gives rise to a superstring*

$$S(P) = s_1 \circ s_2 \circ \cdots \circ s_k$$

*of the strings appearing in  $P$ ; hence, a Hamiltonian path corresponds to a superstring of  $\mathcal{F}$ .*

**Fact 2:** *Let  $P$  be a directed path in  $G(\mathcal{F})$ , and let  $\mathcal{A}$  be the set of strings appearing in  $P$ . Then, if  $\mathcal{A}$  is substring-free,*

$$|S(P)| = \|\mathcal{A}\| - w(P)$$

*where  $\|\mathcal{A}\| = \sum_{s \in \mathcal{A}} |s|$  is the sum of lengths of strings in  $\mathcal{A}$  and  $w(P)$  is the weight of  $P$ , the sum of weights of edges in  $P$ .*

Each Hamiltonian path  $P$  contains involves all the strings in  $\mathcal{F}$  and gives a superstring of  $\mathcal{F}$ .

In addition,

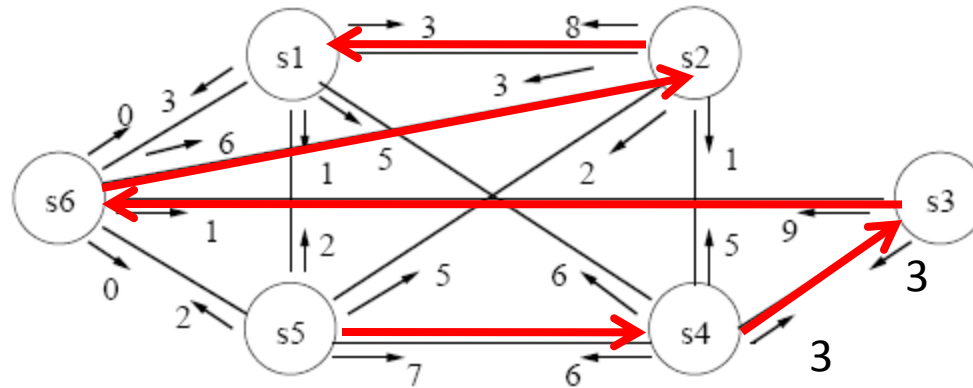
$$|S(P)| + w(P) = \|\mathcal{F}\|$$

**Fact 3** *Each shortest superstring of  $\mathcal{F}$  corresponds one-to-one to a Hamiltonian path with the maximum weight in the overlap graph  $G(\mathcal{F})$ .*

# Graph version of the greedy algorithm

*Scan the edges in non-increasing order of weight and select an edge  $(u, v)$  if any edge of the form  $(u, x)$  or  $(y, v)$  have not been chosen and if the collection of paths constructed so far does not include a path from  $v$  to  $u$ .*

**Example** On the following graph, the greedy algorithm selects the edges  $(s3, s6)$ ,  $(s2, s1)$ ,  $(s5, s4)$ ,  $(s6, s2)$ ,  $(s4, s3)$  in order.



s3s6,	s2s1,	s5s4,	s4s1,	s6s2,	s4s2,	s1s4,	s5s2,	s1s6,	s1s2,	s2s6,	s3s4,	s4s3,	.....
9	8	7	6	6	5	5	5	3	3	3	3	3	

**Theorem 5.7.1** *Let  $G(\mathcal{F})$  be an overlap graph of a substring-free string set  $\mathcal{F}$  and let a maximum Hamiltonian path in  $G(\mathcal{F})$  has weight  $W_{opt}$ . Then, the greedy algorithm always outputs an Hamiltonian path of weight at least  $W_{opt}/2$ .*

In terms of compression measure, the greedy algorithm has an approximation ratio  $\frac{1}{2}$ .

**Proof.** Let  $\mathcal{F}$  have  $s$  strings and let

$$e_1, e_2, \dots, e_{s-1}$$

are all the edges chosen by the algorithm listed in the order in which they were selected. Then the Hamiltonian path  $P_{greedy}$  output from the algorithm has weight  $W_{greedy} = \sum_{i=1}^{s-1} w(e_i)$ , where  $w(e_i)$  is the weight of edge  $e_i$ . We need to prove

$$W_{opt} \leq 2W_{greedy}.$$

For  $i = 1, \dots, s - 1$ , define  $E_i = \{e_1, e_2, \dots, e_i\}$ . We also define  $E_0 = \phi$ . Then,

$$E_0 \subset E_1 \subset E_2 \subset \dots \subset E_{s-1}.$$

Let  $H_i$  be the set of all the Hamiltonian paths that contain the arcs in  $E_i$ . Then,

$$H_0 \supset H_1 \supset H_2 \supset \dots \supset H_{s-1}$$

and  $H_0$  contains all the Hamiltonian path in the graph while  $H_{s-1}$  contains the unique Hamiltonian path consisting of  $e_1, e_2, \dots, e_{s-1}$ .

Let  $W_i = \max_{P \in H_i} W(P)$ . If, for  $i = 1, \dots, s - 1$ ,

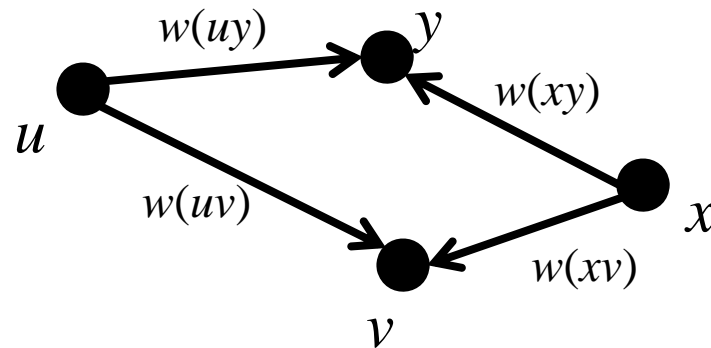
$$W_{i-1} \leq w(e_i) + W_i,$$

then,  $W_{opt} = W_0 \leq \sum_{i=1}^{s-1} w(e_i) + W_{s-1} = 2W_{greedy}$ . This finishes the proof.

Now we prove that  $W_{k-1} \leq w(e_k) + W_k$  for any  $1 \leq k \leq s - 1$  using the following fact.

**Lemma 5.7.1** *Let  $u, v, x, y$  be four nodes in an overlap graph. If  $w(uv) = \max\{w(uv), w(uy), w(xv), w(xy)\}$ , where  $w(uv)$  denote the weight of the directed edge  $(u, v)$ , then,*

$$w(uv) + w(xy) \geq w(uy) + w(xv).$$



**Lemma 5.7.2**  $W_{k-1} \leq w(e_k) + W_k$  for  $k = 1, \dots, s - 1$ .

**Proof.** Fix a  $k$ . Let

$$P = v_1 v_2 \cdots v_s$$

be an Hamiltonian path has weight  $W_{k-1}$  in  $H_{k-1}$ ,  
i.e., it contains  $e_1, e_2, \dots, e_{k-1}$ .

Let  $e_k = (u, v)$ .

P is a Hamiltonian path  u and v appear in P.

If  $e_k$  is an edge in P,

$$H_{k-1} \supseteq H_k \xrightarrow{\text{blue arrow}} w(P) = \max_{P \in H_{k-1}} w(P) = \max_{P \in H_k} w(P)$$

$$\xrightarrow{\text{blue arrow}} W_{k-1} = w(P) = W_k \leq w(e_k) + W_k$$

If  $e_k$  is not an edge in P, we consider the following three cases.



**Case 2**  $u = v_i$  and  $v = v_j$ ,  $i < j$ . See Assignment 3.

Then, we have that  $(v_i, v_{i+1}), (v_{j-1}, v_j) \notin E_{k-1} = \{e_1, e_2, \dots, e_{k-1}\}$ .

Otherwise,  $e_k = (u, v)$  could not be selected after edges  $e_i$ ,  $i \leq k-1$ .

We consider the following Hamiltonian path

$$P' = v_1 v_2 \cdots v_i v_j v_{j+1} \cdots v_s v_{i+1} \cdots v_{j-1}.$$

Since  $P'$  includes  $e_1, e_2, \dots, e_k$  and  $e(k) \geq w(v_i v_{i+1}), w(v_{j-1} v_j)$ ,

$$\begin{aligned} & w(e_k) + W_k \\ & \geq w(e_k) + w(P') \\ & = 2w(e_k) + w(v_s v_{i+1}) + w(P) - w(v_i v_{i+1}) - w(v_{j-1} v_j) \\ & \geq w(P) = W_{k-1} \end{aligned}$$

