Lecture 4

Shortest Superstring Problem

Example set of strings: {001, 100, 101}

001100101 (Concatenation)

0010100 (Better)

Problem Given a set of strings $\vec{s}_1, \vec{s}_2, \cdots, \vec{s}_n$, find the shortest string $\vec{s}$ such that $\forall\, i = 1, 2, \cdots, n$, $\vec{s}_i$ is a substring of $\vec{s}$.

Example set of strings {CATGC ①, CTAAGT ②, GCTA ③, TTCA ④, ATGCATC ⑤}.

$\vec{s}$: GCTAAGTTCATGCATC

(*) Greedy Algorithm (把可以接在一起的依次接好，重疊部份忽略好。)

Step 1.	① + ⑤	CATGCATC ⑥
Step 2.	② + ③	GCTAAGT ⑦
Step 3.	④ + ⑥	TTCATGCATC ⑧
Step 4	⑦ + ⑧	GCTAAGTTCATGCATC.

(Answer)

Prefix graph G of sequences

(•) overlap $(\vec{s}_i, \vec{s}_j)$

$\vec{s}_i$: ACGGCTAT
$\vec{s}_j$: CTATTAG

overlap $(\vec{s}_i, \vec{s}_j)$ = CTAT

(•) prefix $(\vec{s}_i, \vec{s}_j)$: 去掉 overlap $(\vec{s}_i, \vec{s}_j)$ 之後, $\vec{s}_i$ 的前綴.

上例: prefix $(\vec{s}_i, \vec{s}_j)$ = ACGG (有向)

Definition (Prefix graph)

The prefix graph G of a set of strings $\{\vec{s}_1, \vec{s}_2, \cdots, \vec{s}_n\}$ is a double-weighted complete digraph (22向) such that

$V(G) = \{\vec{s}_1, \vec{s}_2, \cdots, \vec{s}_n\}$, $A(G) = \{(\vec{s}_i, \vec{s}_j) \mid 1 \le i, j \le n\}$ and

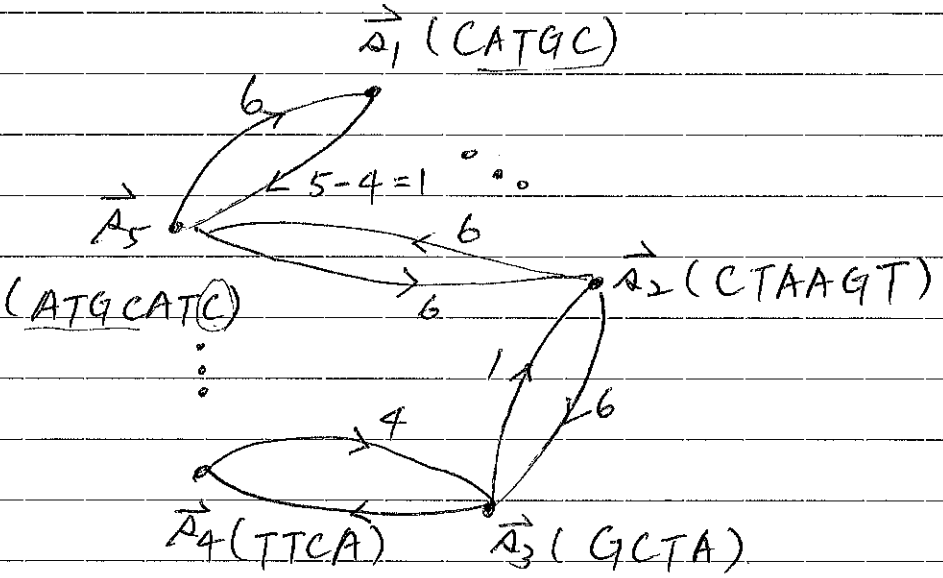
the weight of $(\vec{s}_i, \vec{s}_j) = |\vec{s}_i| -$ overlap $(\vec{s}_i, \vec{s}_j)$, denoted by $w(\vec{s}_i, \vec{s}_j)$.

e.g. (上例)

$w(\vec{s}_i, \vec{s}_j) = 4$, $w(\vec{s}_j, \vec{s}_i) = 0$. (22向分不權重)

Example $\vec{s}_1$ = CATGC, $\vec{s}_2$ = CTAAGT, $\vec{s}_3$ = GCTA, $\vec{s}_4$ = TTCA and $\vec{s}_5$ = ATGCATC.

G



(*) The goal is to find a directed Hamilton path with minimum total weight. (Very hard problem)

An answer

$$\vec{A}_3 \xrightarrow{1} \vec{A}_2 \xrightarrow{5} \vec{A}_4 \xrightarrow{2} \vec{A}_1 \xrightarrow{1} \vec{A}_5$$

$\vec{A}$ = GCTAAGTTCATGCATC (length 16)

$|\vec{A}_3| + |\vec{A}_2| + |\vec{A}_4| + |\vec{A}_1| + |\vec{A}_5| - |\text{overlap}(\vec{A}_3, \vec{A}_2)|$

$- |\text{overlap}(\vec{A}_2, \vec{A}_4)| - |\text{overlap}(\vec{A}_4, \vec{A}_1)| - |\text{overlap}(\vec{A}_1, \vec{A}_5)|$

$= 4 + 6 + 4 + 5 + 7 - 3 - 1 - 2 - 4 = 16.$

Sequencing by Hybridization (SBH Problem)
(雜交定序)

Definition (l-mer composition)

For a string $\vec{s}$ of length n, the l-mer composition, or spectrum, of $\vec{s}$, is the multiset of $n-l+1$ l-mers (consecutive l letters, substring of length l) in $\vec{s}$, denoted by $Spectrum(\vec{s}, l)$

(consecutive $l-1$ letters overlap)

For example $\vec{s}$ = ATGCTGCAAG, $|\vec{s}| = 10$

$l = 3$

$\Rightarrow Spectrum(\vec{s}, l) = \{ATG, TGC, GCT, CTG, TGC, GCA, CAA, AAG\}$

(*) **SBH Problem**

Two l-mers have overlap $l-1$ i.e. $|overlap(\vec{p}, \vec{q})| = l-1$

$\vec{p}$ and $\vec{q}$ ($l-1$ letters)

Let S be a set of l-mers obtained from $\vec{s}$.

Reconstruct $\vec{s}$ from $S = Spectrum(\vec{s}, l)$.

Note: Clearly, we did not know the "order" of elements in S.

Idea: Two consecutive l-mers, $\vec{p}$ and $\vec{q}$ has $overlap(\vec{p}, \vec{q}) = l-1$.

(**) One $\vec{s}$ gives a unique $Spectrum(\vec{s}, l)$ for each $l = 1, 2, \cdots, n-1$.

(***) A given set of l-mers may produce two or more feasible strings, i.e., each of the strings has the same spectrum.

($\Longrightarrow$) 唯一給定。

($\Longleftarrow$) 可能有多個答案。

For example: ATGCGTGGCA $= \vec{s}_1$

ATGGCGTGCA $= \vec{s}_2$

We may obtain the above two distinct strings from the set Spectrum($\vec{s}$?3) = {ATG, TGG, TGC, GTG, GGC, GCA, GCG, CGT}.
v_1 v_2 v_3 v_4 v_5 v_6 v_7 v_8

$\vec{s}_1$: $v_1 - v_3 - v_7 - v_8 - v_4 - v_2 - v_5 - v_6$. ATGCGTGGCA

$\vec{s}_2$: $v_1 - v_2 - v_5 - v_7 - v_8 - v_4 - v_3 - v_6$. ATGGCGTGCA

Observation

SBH problem is a particular (special) case of the Shortest Superstring problem, see it? However, in contrast to SSP, there exists a simple linear-time algorithm for the SBH problem.
(保存得最多的重疊部份)

(*) Construct one $\vec{s}$ not all $\vec{s}$'s.
(Greedy?)

Approach

① Hamiltonian Path problem (Hard problem) ↙ digraph

Let $V(G) = \{\vec{s}_1, \vec{s}_2, \cdots, \vec{s}_n\}$ and $(\vec{s}_i, \vec{s}_j) \in A(G)$ iff overlap$(\vec{s}_i, \vec{s}_j) = l-1$. Then a Hamiltonian path (directed) gives an $\vec{s}$.

② Eulerian Path Problem

From the set of l-mers we obtain a set T of $(l-1)$-mers $\{\vec{t}_1, \vec{t}_2, \cdots, \vec{t}_m\}$. Then, let $\tilde{G}$ be the graph obtained from letting $V(\tilde{G}) = T$ and $\vec{t}_i$ is incident to $\vec{t}_j$ if and only if overlap $(\vec{t}_i, \vec{t}_j) = l-2$ and $\vec{t}_i \wedge \vec{t}_j$ is an l-mer in S.

$\vec{t}_i \wedge \vec{t}_j$ is obtained by combining $\vec{t}_i$ and $\vec{t}_j$ together such that the overlaping portion (suffix of $\vec{t}_i$ and prefix of $\vec{t}_j$) occurs exactly once. For example

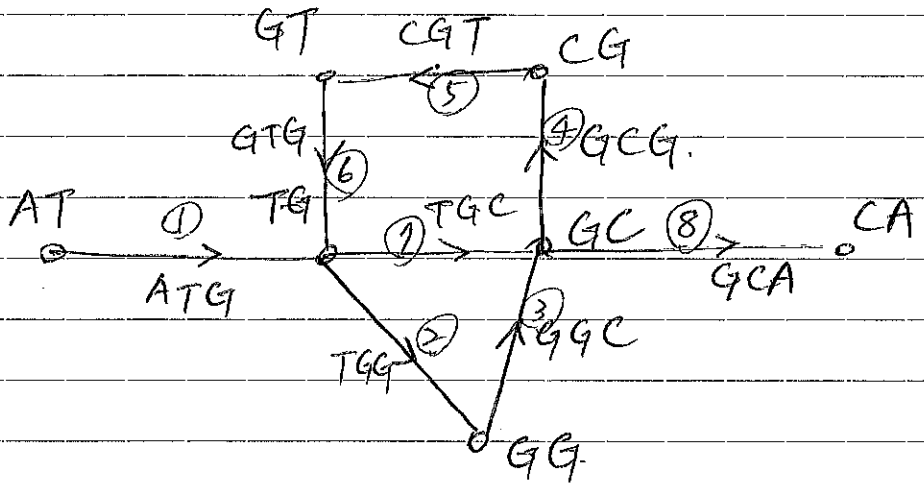
$ATG \wedge TGG = ATGG$. (See 5' for an example.)

(*) Then, an eulerian path (directed) of $\tilde{G}$ gives an $\vec{s}$.

(Note) The number of eulerian circuits in a balanced directed connected digraph can be enumerated. (BEST Theorem)
(directed eulerian graph)

Problem (Open) The number of eulerian circuits in an eulerian graph can also be enumerated, but how?

(**) After the short (500-700) bp DNA reads are sequenced, biologists need to assemble them together to reconstruct the entire genomic DNA sequence. (Fragment Assembly Problem) ← $\leq l$?



① → ② → ③ → ④ → ⑤ → ⑥ → ⑦ → 8

→ ATGGCGTGCA

① → ⑦ → ④ → ⑤ → ⑥ → ② → ③ → 8

→ ATGCGTGGCA

(•) Eulerian path from AT to CA.

(Notable Facts)

1. The error rate in DNA reads produced by modern machines varies from 1% to 3%.

2. One never knowns whether a read came from a target strand DNA sequence or from its Watson-Crick complement (DNA is double-stranded).

3. Repeats in DNA cause a lot of trouble in sequencing DNA. For example, roughly 300 nucleotide Alu sequence is repeated more than a million times throughout the genome, with only 1% to 5% sequence variation.

(Repeats can occur at several different scales.)

4. Fragment assembly algorithms: (Three steps)

(1) Overlap: Finding potentially overlapping reads

(2) Layout: Finding the order of reads along DNA

(3) Consensus: Deriving the DNA sequence from the layout

Review of Graph Theory

1. Any tournament contains a Hamiltonian path.
   The existence of a superstring is warrant.
2. Any even connected graph contains an eulerian circuit.
   (Euler's Theorem)
3. A graph contains an eulerian path if the graph is connected and has at most two odd vertices.
4. (Kotzig, 1968) Let G be a colored connected graph with (edges) even degree of vertices. Then, there is an alternating eulerian circuit in G if and only if every vertex is balanced, i.e. $\forall$ color c, the number of edges incident to v and colored $d_c(v) \le d(v)/2$. ($d(v)$: the degree of v in G)

problem

Proof: 把每一类的相接边分成 pairs of edges with distinct colors.

Corollary: G is bicolored $\Rightarrow d_1(v) = d_2(v)$ for each $v \in V(G)$.

Open problem: Find the number of alternating eulerian circuits in a bicolored eulerian graph.

Set Cover Problem

Let $S_1, S_2, \cdots, S_k$ be subsets of S with $wt(S_i) = w_i$, $i = 1, 2, \cdots$ and $\bigcup_{i=1}^{k} S_i \supseteq S$. Find a set of indices $I \subseteq \{1, 2, \cdots, k\}$ such that (1) $\bigcup_{j \in I} S_j \supseteq S$, and (2) $\sum_{j \in I} w_j$ is minimized.

Example $S = [1, 6]$, $S_1 = \{1, 2\}$, $S_2 = \{1, 3\}$, $S_3 = \{2, 3\}$, $S_4 = \{2, 4, 6\}$, $S_5 = \{3, 5, 6\}$, $S_6 = \{4, 5, 6\}$, $S_7 = \{4, 5\}$, $wt(S_i) = |S_i|$.

Natural weight

	S_1	S_2	S_3	S_4	S_5	S_6	S_7
1	1	1	0	0	0	0	0
2	1	0	1	1	0	0	0
3	0	1	1	0	1	0	0
4	0	0	0	1	0	1	1
5	0	0	0	0	1	1	1
6	0	0	0	1	1	1	0

$S_2 \cup S_3 \cup S_6$

(*) In general S_7 is allowed. ($S_7 \subseteq S_6$) (只是不發生作用而已!)

Hitting set problem (see 8')

(**) So we are looking for some rows $S_{i_1}, S_{i_2}, \cdots, S_{i_t}$ such that $\bigcup_{j=1}^{t} S_{i_j} \supseteq S$ and $\sum_{j=1}^{t} |S_{i_j}|$ is minimized. (=)

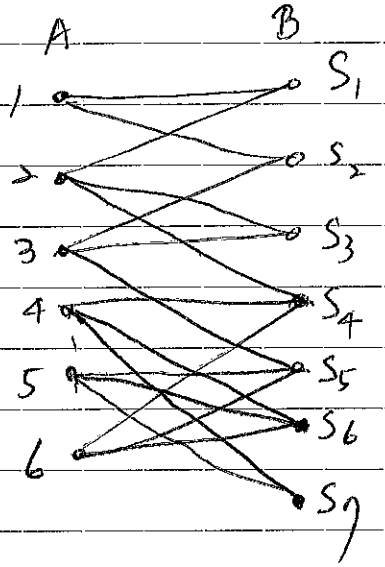
Natural weight

extra!

(***) $S_1 \cup S_2 \cup S_3 \cup S_6 \supseteq S$ and weight $= 9$. (Can you find a better one? Yes!) (=)

Date

Hitting Set Problem : Find a subset X of B in the following bipartite graph $G=(A,B)$ such that $N_G(X) \geq A$.



Find a good X which minimizes $\sum_{x \in X} \deg_G(x)$.

(•) We can also consider subsets of A, Y, such that $N_G(Y) \geq B$.

(*) Weighted version also minimizes $\sum_{x \in X} \deg_G(x)$.

Example (Real)

IBM finds computer viruses (Wikipedia)

S : 5000 known viruses

Subsets (strings) 9000 substrings of 20 or more consecutive bytes from viruses, not found in "good" code.

(*) A set cover of (180) substrings was found. It suffices to search for these 180 substrings to verify the existence of known computer viruses.

Greedy Algorithm

C : represent the set of elements covered so far

$\alpha_{\tilde{S}}$: average cost per newly covered node $\tilde{S}$

Algorithm

1. $C \leftarrow \emptyset$

2. While $C \neq S$ or $C \subsetneq S$ do

Find the set whose cost effectiveness is smallest, say $\tilde{S}$,

Let $\alpha_{\tilde{S}} = \frac{wt(\tilde{S})}{|\tilde{S} \setminus C|} \geq 1$.

$\tilde{S} \setminus C$ 尚未 cover 的部分

$wt(\tilde{S})$ is given

$|\tilde{S}| / |\tilde{S} \setminus C| = \alpha_{\tilde{S}}$

ee值代表效率 ≥ 1

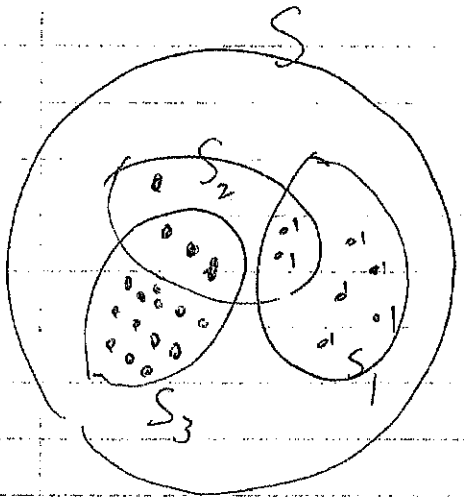
選最大的集合

$\forall\ e \in S \setminus C$, set price$(e) = \alpha$

$C \leftarrow C \cup S$

3. Output picked sets

Example



$wt(S_1) = 7$
$wt(S_2) = 6$
$wt(S_3) = 15$

By "looking", optimal solution is $S_1 \cup S_3$ of weight 22.

By greedy algorithm

1. Choose S_1, $\alpha_{S_1} = \frac{wt(S_1)}{|S_1 \setminus \phi|} = 1$, $C \leftarrow S_1$

2. Choose S_2: $\alpha_{S_2} = \frac{wt(S_2)}{|S_2 \setminus S_1|} = \cdot$, $C \leftarrow S_1 \cup S_2$

3. choose S_3: $\alpha_{S_3} = \frac{wt(S_3)}{|S_3 \setminus (S_1 \cup S_2)|}$

Total: $7 + 6 + 15 = 28$

How good is a greedy algorithm?

(*) In general, it is OK to use, if no better algorithm exists.

We can solve the Set Cover Problem by using Linear Programming.

Definition (Linear Programming)

A "Linear Programming" is the problem of optimizing (minimizing or maximizing) a linear function subject to linear inequality constraints.

Example

Minimize $\sum_{j=1}^{m} c_j x_j$

Subject to $\sum_{j=1}^{n} a_{ij} x_j \geq b_i$, $i = 1, 2, \cdots, m$

$x_j \geq 0$

(Note) 由於答案不一定要 Integer, 有 polynomial time algorithm

"Integer programming is harder."

Set Cover Problem

variable x_j for set S_j $= \begin{cases} 1 & \text{if } S_j \text{ is selected;} \\ 0 & \text{if not.} \end{cases}$

上面例子

$S = [1,6]$, ...

$$\text{Min.} \sum_{j=1}^{k} c_j x_j = OPT$$

$x_j = \begin{cases} 1 & \text{if } S_j \text{ is selected} \\ 0 & \text{if not.} \end{cases}$

subject to

One of S_1 and S_2 must be chosen! →

One of S_5, S_6 and S_7 must be chosen! →

$$\left\{ \begin{array}{l} x_1 + x_2 \geq 1 \\ x_1 + x_3 + x_4 \geq 1 \\ x_2 + x_3 + x_5 \geq 1 \\ x_4 + x_6 + x_7 \geq 1 \\ x_5 + x_6 + x_7 \geq 1 \\ x_4 + x_5 + x_6 \geq 1 \end{array} \right\} ?$$

Consider 1, 2, 3, 4, 5, 6

Each element has to be "covered".

1 : S_1 and S_2

2 : S_1, S_3, S_4,

3 : S_2, S_3, S_5

4 : S_4, S_6, S_7

5 : S_5, S_6, S_7

6 : S_4, S_5, S_6

How to apply ?

Have, $c_j = |S_j|$.

x_i's are in $\{0,1\}$.

$S_1 = \{1,2\}$

$S_2 = \{1,3\}$ ✓

$S_3 = \{2,3\}$ ✓

$S_4 = \{2,4,6\}$

$S_5 = \{3,5,6\}$

$S_6 = \{4,5,6\}$ ✓

$S_7 = \{4,5\}$

Review p. 8

More on "Set Cover Problem"

Problem: Each set of weight (cost) 1.

Target: Find as few sets as possible.

Example: cover [1, 12].

$T_1 = \{1, 2, 3, 4, 5, 6\}$

$T_2 = \{5, 6, 8, 9\}$

$T_3 = \{1, 4, 7, 10\}$

$T_4 = \{2, 5, 7, 8, 11\}$

$T_5 = \{3, 6, 9, 12\}$

$T_6 = \{10, 11\}$

Better?

先選一个；再選尚未cover的集合它增加最多的新元素!

$T_1 \rightarrow T_4 \rightarrow T_5 \rightarrow T_3$

$C = \{T_1, T_3, T_4, T_5\}$

(A solution but not an optimal one!)

Idea of greedy algorithm (Greedy-Set-Cover (X, F)).

(*) At each stage, the greedy algorithm picks the set that covers the greatest number of elements not yet covered.

```
1.  U ← X        ( U = {uncovered elements} )
2.  C ← ∅        ( C = {covered elements} )
3.  while U ≠ ∅
4.  do select an S ∈ F that maximizes |S ∩ U|
5     U ← U \ S.
      Stop at U = ∅
```