Summary

1.

> **Problem: Given a set of strings S and a positive integer K, does S have a superstring of length K?**

2. **Abstract:** A superstring of a set of strings $\{s_1, \ldots s_n\}$ is a string s containing each s_i, $1 \le i \le n$, as a substring. The superstring problem is: Given a set S of strings and a positive integer K, does S have a superstring of length K? The superstring problem has applications to data storage; specifically, data compression. We consider the complexity of the superstring problem. NP-completeness results dealing with sets of strings over both finite and infinite alphabets are presented. Also, for a restricted version of the superstring problem, a linear time Algorithm is given.

3. **Superstring:** A superstring of a set of strings $S = \{s_1, \ldots s_n\}$ is a string s containing each s_i, $1 \le i \le n$, as a substring.

4. **Primitive:** A string is *primitive* if no character appears more than once.

5. First theorem shows the superstring problem to be NP-complete even if for any integer $H \ge 3$, the restriction is made that all strings in the set must be primitive and of length H.

6. For the case $H \ge 8$, a reduction employing a restricted version of the node cover problem appears in Maier and Storer. (1977)

7. The restricted directed Hamilton path problem is the directed Hamilton path problem with the following restrictions:

   (1) There is a designated start node s and a designated end t, with IN(s) = OUT(t) = 0.

   (2) Except for the end node t, all nodes have out-degree greater than 1

8. **<u>Lemma 1.</u> The restricted (by 7.) directed Hamilton path problem is NP-complete.**

9. **<u>Theorem 1.</u>**

   (a) The superstring problem is NP-complete.

   (b) This problem is NP-complete even if for any integer $H \ge 3$, the restriction is made that all strings in the set be primitive and of length H.

Proof:

First aim (1) For nonprimitive strings of length 3

(2) Show how to modify the construction to make all strings primitive and of length *H*.

Let $G = (V, E)$

$V = \{1, \ldots, n\}$, $|E| = m$,

$\Sigma = V \cup B \cup S$, $B = \{\bar{v} \mid v \in V - \{n\}\}$ (扣掉 end 點的 barred symbols),

$S = \{¢, \#, \$\}$: the set of special symbols

barred symbols: local to a node

unbarred symbols: global to the whole graph *G*

Second aim

Create a set of $2 * OUT(v)$ strings: A_v

Let $R_v = \{w_0, \ldots, w_{OUT(v)-1}\}$ be the set of nodes adjacent to *v*

$\therefore A_v = \{\bar{v}w_i\bar{v} \mid w_i \in R_v\} \cup \{w_i\bar{v}w_{i\oplus 1} \mid w_i \in R_v\}$ ($\bar{v}$: local to *v*)

$\oplus$: Addition modulo $OUT(v)$

C_v: A singleton set containing a string of the form $v\#\bar{v}$ called a connector.

Terminal strings: $T = \{¢\#\bar{1},\ n\#\$\}$ (Q)

Let $S = \bigcup_{1 \le i, j < n} (A_j \cap C_i \cap T)$, (Q)

Claim: *G* has a directed Hamilton path if and only if *S* has a superstring of length $2m + 3n$

Suppose *G* has a directed Hamilton path. Let $\{v, w_i\}$ be an edge on the path.

(1) Create a superstring of length $2(OUT(v)) + 2$ for A_v (of the form:

$\bar{v}w_i\bar{v}w_{i\oplus 1}\bar{v} \ldots \bar{v}w_i$) => w_i-*standard superstring* for A_v

($\because w_i$和$\bar{v}$各有 $OUT(v) + 1$ 個)

This superstring is formed by overlapping the strings of A_v in the order:

$\bar{v}w_i\bar{v},\quad w_i\bar{v}w_{i\oplus 1},\quad \bar{v}w_{i\oplus 1}\bar{v},\quad \ldots,\quad \bar{v}w_{i\oplus OUT(v)}\bar{v},\quad w_{i\oplus OUT(v)}\bar{v}w_i$ (共 $2(OUT(v))$ 個)

Each successive pair has an overlap of length 2

(2) The set of w_i -standard superstrings for A_v is in one-to-one correspondence with the cyclic permutations of the integers 0 through $OUT(v) - 1$(剛好也是 $2(OUT(v))$ 個)

(w_i -standard superstrings 與 $0 \sim OUT(v) - 1$ 有著 1-1 對應的關係)

(3) Let $(u_1, u_2, \ldots, u_n)$ denote the directed Hamilton path

$u_1 = 1$ and $u_n = n$

(4) Abbreviate(縮寫) the u_j-standard superstrings for A_{u_i} as $STD(\bar{u}_i, u_j)$

(5) We can form a superstring for S by overlapping the standard superstrings and the strings in S but not in A_v in the order: (Q)

$$\underbrace{¢\#\bar{1}}, STD(\bar{1},u_2), u_2\#\bar{u}_2, STD(\bar{u}_2,u_3), u_3\#\bar{u}_3, \mathbf{K}, u_{n-1}\#\bar{u}_{n-1}, STD(\bar{u}_{n-1},n), \underbrace{n\#\$}$$

令起點為 1 終點為 n

$\boxed{\text{terminal strings}}$

The superstring has length $\sum_{i=1}^{n-1}(2*OUT(i)+2)+(n-2)+4=2m+3n$

$\boxed{\text{Since } u_2\#\bar{u}_2 \sim u_{n-1}\#\bar{u}_{n-1} \text{ are } (n-2) \text{ items}}$ $\boxed{¢,\bar{1},n,\$}$

“#”只是連接符號，不算在內

而 $\sum_{i=1}^{n-1}2*OUT(i)=\sum_{i=1}^{n}2*OUT(i)=2|E|=2m$

$\boxed{\text{Since degree of } n\text{-th node is entirely “indegree”}}$

(6) To prove the converse, we show that $2m + 3n$ is a lower bound on the size of a superstring for S and then show that this lower bound can only be achieved if the superstring **encodes a directed Hamiltonian path.**

(7) There are a total of $2m + n$ strings, with a total length of $3(2m + n)$.

$$3(2m+n)-2(2m+n-1)=2m+n+2$$

$\boxed{\text{Since Overlap of length 2}}$

$n - 2$ connectors can only have overlaps of length 1 on either side.

(Since no string begins or ends with #)

Terminal strings can overlap at most one symbol on only one side.

Therefore we obtain a lower bound: $(2m + n + 2) + 2(n - 2) + 2 = 2m + 3n$

on the length of a superstring for S

(8) Let x be the string between the two #’s, all substrings of x except the first and the last must have overlaps of length two on both sides. Furthermore, all

strings in A_v except two must have overlaps of length 2 on both sides.

$\therefore$ Every string in A_v but one must occur contiguously in order

And $\because$ x contains one string from A_v, $\therefore$ It must contain them all.

<u>Thus, x is the w_j - standard superstring for A_v.</u>

(9) We can recover a directed Hamilton path by looking at the symbols next to #. ($\because$The barred and unbarred symbol of each connector correspond to the same node in G.)

(10) Restrict: All strings are primitive and exactly length $H \geq 3$

(11) Include $\{\hat{a} \mid a \in V\}$ to Σ.

(12) For $H = 3$,

Replace strings of the form $\bar{v}a\bar{v}$ by the strings $\bar{v}a\hat{v}$, $\hat{a}\hat{v}a$, $\hat{v}\hat{a}\bar{v}$

Replace strings of the form $a\bar{v}b$ by $\hat{a}\bar{v}b$ (爲了跟$\hat{v}\hat{a}\bar{v}$能相接)

(13) For $H \geq 4$,

Let y be a primitive string over an alphabet disjoint from Σ of length $H-4$.

Let y' *be a* primitive string over an alphabet disjoint from Σ of length $H-2$

(14) Replace the # in all connectors and terminals by y'.

(15) Replace strings of the form $\bar{v}\hat{a}\bar{v}$ by $\bar{v}ay\hat{a}\hat{v}$ and those of the form $a\bar{v}b$ by $\hat{a}\hat{v}y\bar{v}b$

剛好長度爲 H

(16) There is an integer k such that the theorem holds. And we can also check that the superstring problem is in NP and the above reductions can be done in polynomial time.

The proof is done.

---

10. **<u>Definition 3.</u>** For a directed graph $G = (V, E)$, if $G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k)$ are the loosely connected components of G

$$PATH(G) = \sum_{i=1}^{k} \max\left\{1, \sum_{v \in V_i} \frac{|IN(v) - OUT(v)|}{2}\right\}.$$ (It is just the number of paths in a minimal path-decomposition of a directed graph G)

11. **<u>Definition:</u>** A path-decomposition of G is **a partition of E** into edge disjoint paths.

∴ ***PATH*(*G*) = 在一個有向圖 *G* 的 minimal path-decomposition 中，path 的個數**

12. **<u>Lemma 2.</u>** The number of paths in a minimal path-decomposition of a directed graph G is given by *PATH*(*G*).

Theorem 2 and its corollary present a linear time algorithm to find a minimal length superstring for a set of strings of length less than or equal to 2

13. **<u>Algorithm 1</u>**

WHILE there exists a node *v* in *G* with *IN*(*v*) < *OUT*(*v*) ***DO***

Starting at *v*, traverse edges at random until a node with no outgoing edges is reached, delete the edges traversed from *G*, and add this path to *P.*

WHILE *G* is not empty ***DO***

IF there exists a cycle ***c*** which intersects a path ***p*** in ***P***

THEN Delete ***c*** from *G* and "splice" it into ***p***.

ELSE Delete a cycle from *G* and add it to *P*.

<u>*Proof:*</u>

Each time a path ***p*** is deleted from *G* and added to *P* in the first *WHILE* loop

The outdegree of the start node of ***p*** and the indegree of the end node of ***p*** are reduced by 1 respectively, and so do other nodes ***v*** of ***p*** .

∴ $|IN(v) - OUT(v)|$ is unchanged.

ð This loop produces $\sum_{v \in V} \dfrac{|IN(v) - OUT(v)|}{2}$ paths.

(除以 2 是因爲某些邊的 destination 等於別的邊的 starting point)

The second WHILE loop adds a new path to *P* only when <u>a loosely connected component,</u> consisting entirely of cycles (i.e., *IN*(*v*) = *OUT*(*v*) for all nodes *v* in this component), <u>is encountered for the first time.</u>

14. **<u>Theorem 2.</u>** For a set of strings *S* = {s_1 ,…, s_n} and an integer *K*, if $|w_i| \le 2$, $1 \le i \le n$, then there is a linear time and space algorithm (on a RAM) to decide if *S* has a superstring of length *K*.

Applications: (1) Storing Huffman trees for encoding letter pairs.

(2) Storing a directed graph G.

Proof:

We can assume that all strings in S have length exactly 2

∵Strings of length 1 are either a substring of a string of length 2 or are a unique character not appearing anywhere else in S. (∵字串長度不是 2 就是 1)

We can also assume all strings in *W* to be primitive

∵For a nonprimitive string $s_i = aa$ in S, if the character 'a' does not appear anywhere else in S, then S has a superstring of length K if and only if $S - \{s_i\}$ has a superstring of length $K-2$

o.w., S has a superstring of length K if and only if $S-\{s_i\}$ has a superstring of length $K-1$.

We can associate a directed graph $G = (V, E)$ with S by letting $V = \Sigma$ $(= V \cup B \cup S)$ and $(a, b) \in E$ when $ab \in S$

∴ S has a superstring of length K if and only if $PATH(G) \leqq K - |S|$ and $PATH(G)$ can be computed using linear time and space.

15. **Corollary 2.1.** There is a linear time and space algorithm to find a minimal length superstring for a set of strings of length less than or equal to 2.

16. **Corollary 2.2** For a multiset of strings S over alphabet $\sum$, algorithm exist to find a minimal length superstring for S which use the following amounts of time and space:

(1) Linear expected time and linear space.
(2) $o(\| S \| \text{LEN}_2 | S |)$ time and linear space.
(3) *Linear time and* $o(| S | + | \sum |^2)$ space.

Proof:

(1) Use hashing techniques
(2) Use dictionary techniques
(3) Strings of length 1: Can be dealt with as in Corollary 2.1

Strings of length 2: May be tabulated in linear time by using an $o(|\Sigma| \times |\Sigma|)$ matrix. It can be effectively initialized to all zeros in linear time by employs an $o(|W|)$ stack and "hand shaking" protocol.

Bounded Size Alphabets

We can take an alphabet $\Sigma = \{a_1, \ldots, a_m\}$ and encode a_i, $1 \le i \le m$, over the alphabet $\Sigma' = \{0, 1, a\}$ by writing ai as $\bar{i}a$ where $\bar{i}$ denotes i written in binary using $LEN_2(m)$ bits.

17. **<u>Theorem 3.</u>** The superstring problem is *NP*-complete even if for any real number $h > 1$, the problem is restricted to instances S, K where S is written over the alphabet {0, 1} and all strings in S have length $\lceil h \, LEN_2 \| S \| \rceil$.

18. **<u>Conclusion:</u>** Since the superstring problem has many practical applications, the *NP*-completeness results presented in this paper should not discourage future research regarding the superstring problem. Rather, they should provide the impetus for studying approximation algorithms and heuristics for finding a minimal length superstring.

19.

Summary

1.

> **Problem: Given a set of strings S and a positive integer K, does S have a superstring of length K?**

2. **Abstract:** A superstring of a set of strings $S = \{s_1, \ldots, s_n\}$ is a string s containing each s_i, $1 \le i \le n$, as a substring. The superstring problem is: Given a set S of strings and a positive integer K, does S have a superstring of length K? The superstring problem has applications to data storage; specifically, data compression. We consider the complexity of the superstring problem. NP-completeness results dealing with sets of strings over both finite and infinite alphabets are presented. Also, for a restricted version of the superstring problem, a linear time Algorithm is given.

3. **Superstring:** A superstring of a set of strings $S = \{s_1, \ldots, s_n\}$ is a string s containing each s_i, $1 \le i \le n$, as a substring.

4. **Primitive:** A string is ***primitive*** if no character appears more than once.

5. First theorem shows the superstring problem to be NP-complete even if for any integer $H \ge 3$, the restriction is made that all strings in the set must be primitive and of length H.

6. For the case $H \ge 8$, a reduction employing a restricted version of the node cover problem appears in Maier and Storer. (1977)

7. The restricted directed Hamilton path problem is the directed Hamilton path problem with the following restrictions:

   (1) There is a designated start node s and a designated end t, with $\mathrm{IN}(s) = \mathrm{OUT}(t) = 0$.

   (2) Except for the end node t, all nodes have out-degree greater than 1

8. **<u>Lemma 1.</u> The restricted (by 7.) directed Hamilton path problem is NP-complete.**

9. **<u>Theorem 1.</u>**

   (a) The superstring problem is NP-complete.

   (b) This problem is NP-complete even if for any integer $H \ge 3$, the restriction is made that all strings in the set be primitive and of length H.

Proof:

<u>First aim</u> (1) For nonprimitive strings of length 3

(2) Show how to modify the construction to make all strings primitive and of length *H*.

Let $G = (V, E)$

$V = \{1, \ldots, n\}, |E| = m,$

$\Sigma = V \cup B \cup S$, $B = \{\bar{v} \mid v \in V - \{n\}\}$ (扣掉 end 點的 barred symbols),

$S = \{¢, \#, \$\}$: the set of special symbols

barred symbols: local to a node

unbarred symbols: global to the whole graph *G*

<u>Second aim</u>

Create a set of $2 * OUT(v)$ strings: A_v

Let $R_v = \{w_0, \ldots, w_{OUT(v)-1}\}$ be the set of nodes adjacent to *v*

$\therefore A_v = \{\bar{v}w_i\bar{v} \mid w_i \in R_v\} \cup \{w_i\bar{v}w_{i\oplus 1} \mid w_i \in R_v\}$ ($\bar{v}$: local to *v*)

$\oplus$: Addition modulo $OUT(v)$

C_v: A singleton set containing a string of the form $v\#\bar{v}$ called a connector.

Terminal strings: $T = \{¢\#\bar{1},\ n\#\$\}$ (Q)

Let $S = \bigcup_{1 \le i, j < n} (A_j \cap C_i \cap T)$, (Q)

<u>Claim:</u> *G* has a directed Hamilton path if and only if *S* has a superstring of length $2m + 3n$

Suppose *G* has a directed Hamilton path. Let $\{v, w_i\}$ be an edge on the path.

(1) Create a superstring of length $2(OUT(v)) + 2$ for A_v (of the form:

$\bar{v}w_i\bar{v}w_{i\oplus 1}\bar{v} \ldots \bar{v}w_i$) => w_i*-standard superstring* for A_v

($\because w_i$和$\bar{v}$各有 $OUT(v) + 1$ 個)

This superstring is formed by overlapping the strings of A_v in the order:

$\bar{v}w_i\bar{v},\quad w_i\bar{v}w_{i\oplus 1},\quad \bar{v}w_{i\oplus 1}\bar{v},\quad \ldots,\quad \bar{v}w_{i\oplus OUT(v)}\bar{v},\quad w_{i\oplus OUT(v)}\bar{v}w_i$ (共 $2(OUT(v))$ 個)

Each successive pair has an overlap of length 2

(2) The set of w_i-standard superstrings for A_v is in one-to-one correspondence with the cyclic permutations of the integers 0 through $OUT(v)-1$(剛好也是 $2(OUT(v))$ 個)

(w_i-standard superstrings 與 0 ~ OUT(v) – 1 有著 1-1 對應的關係)

(3) Let $(u_1, u_2, \ldots, u_n)$ denote the directed Hamilton path

$u_1 = 1$ and $u_n = n$

(4) Abbreviate(縮寫) the u_j-standard superstrings for A_{u_i} as $STD(\bar{u}_i, u_j)$

(5) We can form a superstring for S by overlapping the standard superstrings and the strings in S but not in A_v in the order: (Q)

$$\underline{¢\#\bar{1}}, STD(\bar{1},u_2), u_2\#\bar{u}_2, STD(\bar{u}_2,u_3), u_3\#\bar{u}_3, \mathbf{K}, u_{n-1}\#\bar{u}_{n-1}, STD(\bar{u}_{n-1},n), \underline{n\#\$}$$

令起點爲 1 終點爲 n

$\boxed{\text{terminal strings}}$

The superstring has length $\sum_{i=1}^{n-1}(2*OUT(i)+2)+(n-2)+4=2m+3n$

$\boxed{\text{Since } u_2\#\bar{u}_2 \sim u_{n-1}\#\bar{u}_{n-1} \text{ are } (n-2) \text{ items}}$ $\boxed{¢,\bar{1},n,\$}$

"#"只是連接符號，不算在內

而 $\sum_{i=1}^{n-1}2*OUT(i)=\sum_{i=1}^{n}2*OUT(i)=2|E|=2m$

$\boxed{\text{Since degree of } n\text{-th node is entirely "indegree"}}$

(6) To prove the converse, we show that $2m + 3n$ is a lower bound on the size of a superstring for S and then show that this lower bound can only be achieved if the superstring **encodes a directed Hamiltonian path.**

(7) There are a total of $2m + n$ strings, with a total length of $3(2m + n)$.

$$3(2m+n)-2(2m+n-1)=2m+n+2$$

$\boxed{\text{Since Overlap of length 2}}$

n – 2 connectors can only have overlaps of length 1 on either side.

(Since no string begins or ends with #)

Terminal strings can overlap at most one symbol on only one side.

Therefore we obtain a lower bound: $(2m + n + 2) + 2(n - 2) + 2 = 2m + 3n$ on the length of a superstring for S

(8) Let x be the string between the two #'s, all substrings of x except the first and the last must have overlaps of length two on both sides. Furthermore, all

strings in A_v except two must have overlaps of length 2 on both sides.

$\therefore$ Every string in A_v but one must occur contiguously in order

And $\because$ x contains one string from A_v, $\therefore$ It must contain them all.

Thus, x is the w_j - standard superstring for A_v.

(9) We can recover a directed Hamilton path by looking at the symbols next to #. ($\because$The barred and unbarred symbol of each connector correspond to the same node in G.)

(10) Restrict: All strings are primitive and exactly length $H \geq 3$

(11) Include $\{\hat{a} \mid a \in V\}$ to Σ.

(12) For $H = 3$,

Replace strings of the form $\bar{v}a\bar{v}$ by the strings $\bar{v}a\hat{v}$, $a\hat{v}\hat{a}$, $\hat{v}\hat{a}\bar{v}$

Replace strings of the form $a\bar{v}b$ by $\hat{a}\bar{v}b$ (爲了跟 $\hat{v}\hat{a}\bar{v}$ 能相接)

(13) For $H \geq 4$,

Let y be a primitive string over an alphabet disjoint from Σ of length $H-4$.

Let y' *be a* primitive string over an alphabet disjoint from Σ of length $H-2$

(14) Replace the # in all connectors and terminals by y'.

(15) Replace strings of the form $\bar{v}\hat{a}\bar{v}$ by $\bar{v}ay\hat{a}\hat{v}$ and those of the form $a\bar{v}b$ by $\hat{a}\hat{v}y\bar{v}b$

剛好長度爲 H

(16) There is an integer k such that the theorem holds. And we can also check that the superstring problem is in NP and the above reductions can be done in polynomial time.

The proof is done.

---

10. **Definition 3.** For a directed graph $G = (V, E)$, if $G_1 = (V_1, E_1), \ldots, G_k = (V_k, E_k)$ are the loosely connected components of G

$$PATH(G) = \sum_{i=1}^{k} \max\left\{1, \sum_{v \in V_i} \frac{|IN(v) - OUT(v)|}{2}\right\}.$$

(It is just the number of paths in a minimal path-decomposition of a directed graph G)

11. **Definition:** A path-decomposition of G is **a partition of E** into edge disjoint paths.

∴ ***PATH*(*G*) = 在一個有向圖 *G* 的 minimal path-decomposition 中，path 的個數**

12. **<u>Lemma 2.</u>** The number of paths in a minimal path-decomposition of a directed graph G is given by *PATH*(*G*).

Theorem 2 and its corollary present a linear time algorithm to find a minimal length superstring for a set of strings of length less than or equal to 2

13. **<u>Algorithm 1</u>**

WHILE there exists a node *v* in *G* with *IN*(*v*) < *OUT*(*v*) ***DO***

Starting at *v*, traverse edges at random until a node with no outgoing edges is reached, delete the edges traversed from *G*, and add this path to *P.*

WHILE *G* is not empty ***DO***

IF there exists a cycle ***c*** which intersects a path ***p*** in ***P***

THEN Delete ***c*** from ***G*** and "splice" it into ***p***.

ELSE Delete a cycle from ***G*** and add it to ***P***.

<u>*Proof:*</u>

Each time a path ***p*** is deleted from ***G*** and added to ***P*** in the first *WHILE* loop

The outdegree of the start node of ***p*** and the indegree of the end node of ***p*** are reduced by 1 respectively, and so do other nodes *v* of ***p*** .

∴ $|IN(v)-OUT(v)|$ is unchanged.

ð This loop produces $\sum_{v\in V}\frac{|IN(v)-OUT(v)|}{2}$ paths.

(除以 2 是因爲某些邊的 destination 等於別的邊的 starting point)

The second WHILE loop adds a new path to *P* only when <u>a loosely connected component</u>, consisting entirely of cycles (i.e., *IN*(*v*) = *OUT*(*v*) for all nodes *v* in this component), <u>is encountered for the first time.</u>

14. **<u>Theorem 2.</u>** For a set of strings $S = \{s_1, \ldots, s_n\}$ and an integer *K*, if $|w_i| \leq 2$, $1 \leq i \leq n$, then there is a linear time and space algorithm (on a RAM) to decide if *S* has a superstring of length *K*.

Applications: (1) Storing Huffman trees for encoding letter pairs.

(2) Storing a directed graph G.

Proof:

We can assume that all strings in S have length exactly 2

∵Strings of length 1 are either a substring of a string of length 2 or are a unique character not appearing anywhere else in S. (∵字串長度不是 2 就是 1)

We can also assume all strings in *W* to be primitive

∵For a nonprimitive string $s_i = aa$ in S, if the character 'a' does not appear anywhere else in S, then S has a superstring of length K if and only if $S - \{s_i\}$has a superstring of length $K-2$

o.w., S has a superstring of length K if and only if $S-\{s_i\}$ has a superstring of length $K-1$.

We can associate a directed graph $G = (V, E)$ with S by letting $V = \Sigma$ $(= V \cup B \cup S)$ and $(a, b) \in E$ when $ab \in S$

∴ S has a superstring of length K if and only if $PATH(G) \leqq K - |S|$ and $PATH(G)$ can be computed using linear time and space.

15. **Corollary 2.1.** There is a linear time and space algorithm to find a minimal length superstring for a set of strings of length less than or equal to 2.

16. **Corollary 2.2** For a multiset of strings S over alphabet $\sum$, algorithm exist to find a minimal length superstring for S which use the following amounts of time and space:

(1) Linear expected time and linear space.
(2) $o(\| S \| \text{ LEN}_2 | S |)$ time and linear space.
(3) *Linear time and* $o(| S | + | \sum |^2)$ space.
Proof:
(1) Use hashing techniques
(2) Use dictionary techniques
(3) Strings of length 1: Can be dealt with as in Corollary 2.1
Strings of length 2: May be tabulated in linear time by using an $o(|\Sigma| \times |\Sigma|)$ matrix. It can be effectively initialized to all zeros in linear time by employs an $o(|W|)$ stack and "hand shaking" protocol.

Bounded Size Alphabets

We can take an alphabet $\Sigma = \{a_1,\ldots, a_m\}$ and encode a_i, $1 \le i \le m$, over the alphabet $\Sigma' = \{ 0, 1, a \}$ by writing ai as $\bar{i}a$ where $\bar{i}$ denotes i written in binary using $LEN_2(m)$ bits.

17. **<u>Theorem 3.</u>** The superstring problem is *NP*-complete even if for any real number $h > 1$, the problem is restricted to instances S, K where S is written over the alphabet {0, 1} and all strings in S have length $\lceil h\, LEN_2 \| S \| \rceil$.

18. **<u>Conclusion:</u>** Since the superstring problem has many practical applications, the *NP*-completeness results presented in this paper should not discourage future research regarding the superstring problem. Rather, they should provide the impetus for studying approximation algorithms and heuristics for finding a minimal length superstring.

19.