Sorting Strings By Reversals and By Transpositions

David A. Christie and Robert W. Irving

Outline

- Introduction.
- Terminology and notation.
- Reversal distance between binary strings.
- Transposition distance between binary strings.
- NP-completeness of reversal distance.
- Conclusion.

Introduction

- The problems of sorting by reversals and sorting by transpositions have been studied because of their applications to genome comparison.
- In the context of genome comparisons, duplicate genes can occur, so that the permutation model is not always the appropriate one.
- Ex: ATCGAATCG

- In this paper, we define reversal distance and transposition distance on strings and investigate these new problems, which are of interest in their own right, focusing primarily on the case of a binary alphabet.
- Ex: $S=010001110101$
- We say that S and T are related if T is a rearrangement of S.

Reversals and transpositions over strings

For strings S and T

- the *reversal distance* $d_r(S,T)$ between S and T is the minimum number of reversals required to transform S into T.
- the *reversal distance* $d_t(S,T)$ between S and T is the minimum number of transpositions required to transform S into T.
- Ex: $S = abaab, T = ababa,$
  $d_r(S,T) = 1.\ \ S = abaab \rightarrow ababa = T$
  $d_t(S,T) = 1.\ \ S = abaab \rightarrow ababa = T$

Terminology and notation

1. The ith symbol of a string S is denoted by $S(i)$.
   Ex: S = 0110100. $S(1) = 0$, $S(2) = 1$.
2. A reversal on a string will be represented by enclosing in brackets the substring to be reversed.
   Ex: 0[1010110]1 = 0011010 11.
3. Two adjacent substrings are bracketed will be used to describe transpositions.
   Ex: 0[10][1011]01 = 0101110 01.

- 0^k: represent a string of zeros of length k.
- 1^k: represent a string of ones of length k.
  Ex: $0^3 = 000$, $1^4 = 1111$.
- In general, S^k represent the string obtained by concatenating k copies of S.
  Ex: $S = 011$, $S^3 = 011011011$.
- $S \cdot T$ or ST: denote the concatenation of strings S and T.
  Ex: $S = 01001$, $T = 10111$, $ST = 0100110111$.

- S^+: represent the concatenation of one or more copies of S.
- S^*: represent the concatenation of zero or more copies of S.
- B_n: the set of binary strings of length n.
  Ex: n = 2, $B_2 = \{00, 01, 10, 11\}$
- The diameter of B_n is the maximum distance between any two related binary strings of length n.

- $E_k = 0^k 1^k$, $C_k = (10)^k$
  Ex: $E_4 = 00001111$, $C_4 = 10101010$.
- $\bar{S}$: represent the string derived from S by switching ones and zeros.
- S^R: represent the string S in reverse order.
  Ex: $S = 0100110001$,
  $\bar{S} = 1011001110$,
  $S^R = 1000110010$,
  $\bar{S}^R = 0111001101$.

- String X and Y are isomorphic to string S and T if
  (1) $X = S$ and $\mathrm{Y} = T$, or $X = T$ and $\mathrm{Y} = S$ or
  (2) $X = \bar{S}$ and $\mathrm{Y} = \bar{T}$, or $X = \bar{T}$ and $\mathrm{Y} = \bar{S}$ or
  (3) $X = S^R$ and $Y = T^R$, or $X = T^R$ and $\mathrm{Y} = S^R$ or
  (4) $X = \bar{S}^R$ and $Y = \bar{T}^R$, or $X = \bar{T}^R$ and $\mathrm{Y} = \bar{S}^R$.
- In other words, the pair $\{ X, Y \}$ and $\{ S, T \}$ are isomorphic if one pair can be obtained from the other by a fixed permutation of the alphabet, followed by an optional complete reversal of both strings over an arbitrary alphabet.

- If X and Y are isomorphic to S and T, then $d_r(X,Y) = d_r(S,T)$ and $d_t(X,Y) = d_t(S,T)$.
- $lcp(S,T)$: the length of the longest common prefix of S and T.
  Ex: $S = rearranging, T = rearrangement$
  $lcp(S,T) = 8.$
- $lcs(S,T)$: the length of the longest common suffix of S and T.
  Ex: $S = rearranging, T = sorting$
  $lcs(S,T) = 3.$

- A block of zeros is a maximal length substring that consists only of the character 0.
- A block of ones is defined similarly.
- $b(S)$ denote the total number of blocks in S.
- $z(S)$ denote the number of blocks of zeros in S.
  Ex: $S = 001110101$, $b(S) = 6$ and $z(S) = 3$.
- Finally, we represent by $\cdots$ an arbitrary substring of length ≥ 0.
  Ex: if S has prefix 01, a substring 00, and suffix 11, then we could write
  $S = 01 \cdots 00 \cdots 11$.

Reversal distance between binary strings.

- Lower bound
- Upper bound
- Reversal diameter of B_n
- Sorting by reversals.

Breakpoint

- For permutations, two elements form a breakpoint if they are adjacent in π but not adjacent in the identity permutation.
  Ex: $\pi = (1\ 3\ 2\ 4)$, 13 is a breakpoint.
- Substrings of length two represent adjacencies in strings S and T.

- For example, $S = 00011$, $T = 00110$.
  S contains more 00 substrings than T, then each extra 00 must be broken, by a reversal, at some time in the transformation from S into T.
- Each extra 00 in S is an example of a reversal breakpoint.
- Breakpoints also occur for 01, 10 and 11 substrings as well.
- However, reversals can convert 01 into 10, these substrings must be counted together when consider reversal breakpoints.

- Breakpoints can also be contributed from the beginning and end of the strings. Ex: $S = 011$, $T = 110$, the position one and end are breakpoints.
- For the above case, we extend S and T by adding α at the beginning and ω at the end of both strings.

- The number of times the substring ab occurs in S is denoted by $f_{ab}(S)$, where $a, b \in \{ \alpha, 0, 1, \omega \}$. Ex: $S = 001011$, $f_{01}(S) = 2$, $f_{\alpha 0}(S) = 1$.
- For convenience, $\alpha < 0 < 1 < \omega$.
- The number of reversal breakpoints between S and T,
  $b_r(S,T) =$
  $\sum_{\alpha \le a < b \le \omega} \delta(f_{ab}(S) + f_{ba}(S) - f_{ab}(T) - f_{ba}(T)) +$
  $\sum_{0 \le a \le 1} \delta(f_{aa}(S) - f_{aa}(T))$.

  where $\delta(x) = x$ if $x > 0$ and 0 otherwise.

- Ex1 : if $S = T$, then $b_r(S,T) = 0$.
- Ex2: $S = 100101$ and $T = 101001$

$$b_r(S,T) = \delta(f_{\alpha 1}(S) + f_{1\alpha}(S) - f_{\alpha 1}(T) - f_{1\alpha}(T)) +$$
$$\delta(f_{01}(S) + f_{10}(S) - f_{01}(T) - f_{10}(T)) +$$
$$\delta(f_{1\omega}(S) + f_{\omega 1}(S) - f_{\omega 1}(T) - f_{1\omega}(T)) +$$
$$\delta(f_{00}(S) - f_{00}(T)) + \delta(f_{11}(S) - f_{11}(T))$$
$$= \delta(1 + 0 - 1 - 0) + \delta(2 + 2 - 2 - 2) +$$
$$\delta(1 + 0 - 1 - 0) + \delta(1 - 1) + \delta(0 - 0) = 0$$

- $b_r(S,T) = b_r(T,S)$

- Lemma 3.1 Suppose that S' is obtained from S by a single reversal. Then $b_r(S', T) \geq b_r(S, T) - 2$.
- Proof.
  The number of breakpoints can be reduced by at most two as a result of such a reversal.
  Ex: $S = 10101$, $S' = 10[10]1 = 10011$, $T = 10011$

$$b_r(S', T) = \delta(f_{01}(S') + f_{10}(S') - f_{01}(T) - f_{10}(T)) + \delta(f_{00}(S') - f_{00}(T)) + \delta(f_{11}(S') - f_{11}(T))$$
$$= \delta(1 + 1 - 1 - 1) + \delta(1 - 1) + \delta(1 - 1) = 0$$
$$b_r(S, T) = \delta(f_{01}(S) + f_{10}(S) - f_{01}(T) - f_{10}(T)) + \delta(f_{00}(S) - f_{00}(T)) + \delta(f_{11}(S) - f_{11}(T))$$
$$= \delta(2 + 2 - 1 - 1) + \delta(0 - 1) + \delta(0 - 1) = 2$$

Lower bound

- Theorem 3.2 Let S and T be related binary strings. Then

$$d_r(S,T) \geq \left\lceil \frac{b_r(S,T)}{2} \right\rceil$$

- Proof. From the lemma 3.1, it easy to check.

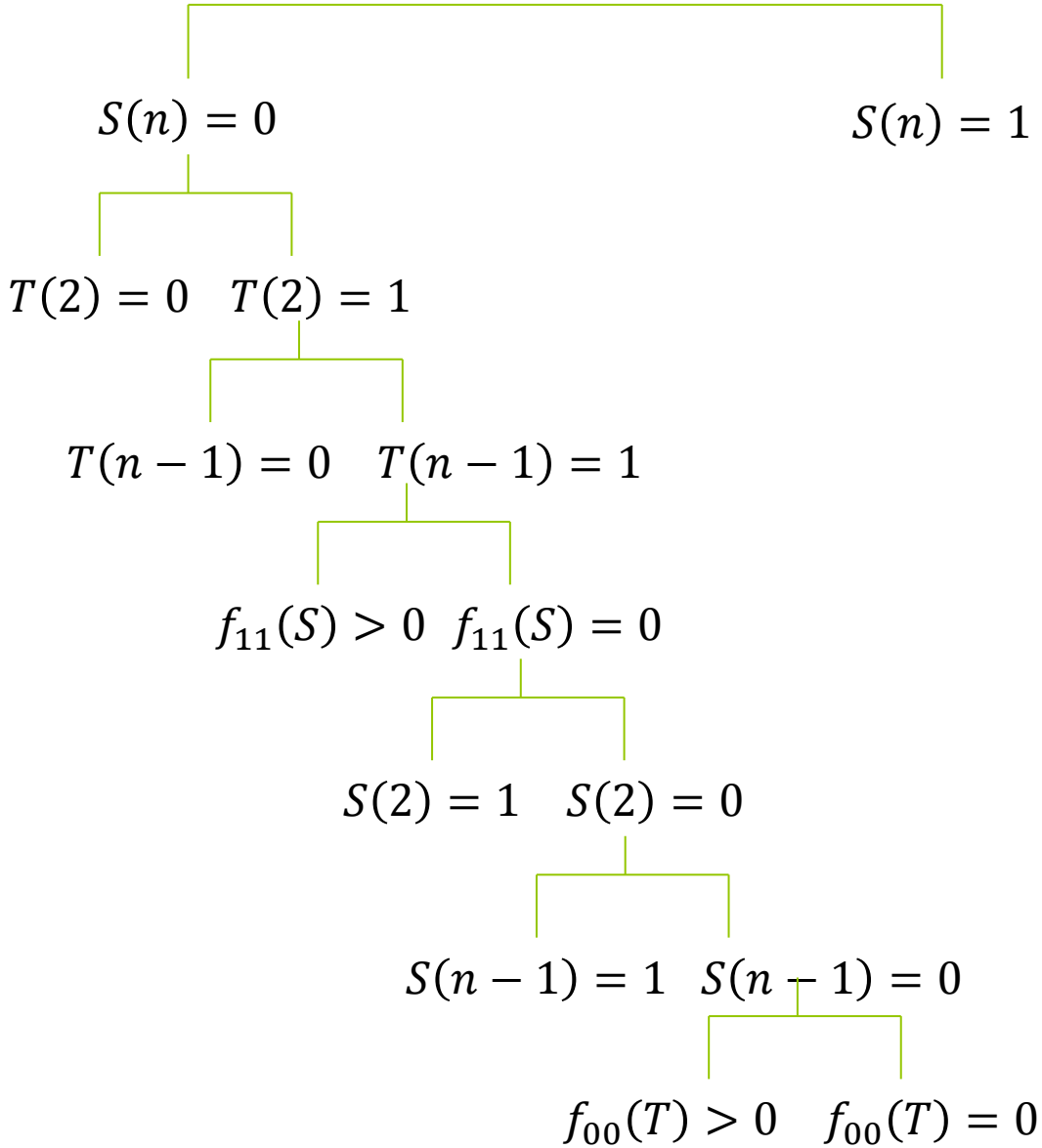
- The lower bound is not tight.
  Ex: $S = 0011000111$ and $T = 1110011000$
  then $b_r(S,T) = 2$ and $d_r(S,T) = 2 \neq \left\lceil \frac{b_r(S,T)}{2} \right\rceil$.

- Lemma 3.3 Let S and T be related strings of length n such that $S \neq T$. Then it is possible either
  (a) to apply a reversal on S resulting in the string S', such that
  $lcp(S', T) + lcs(S', T) \geq lcp(S, T) + lcs(S, T) + 2$

  (b) to apply a reversal on T resulting in the string T' such that
  $lcp(S, T') + lcs(S, T') \geq lcp(S, T) + lcs(S, T) + 2$

Proof.

- 1. WLOG, it can be assume that $S(1) = 0$.
- 2. Assume that $S(1) \neq T(1)$, and $S(n) \neq T(n)$.
- 3. We describe seven cases and show a reversal with the require property in each case.


$S(n) = 0$
$S(n) = 1$
$T(2) = 0$
$T(2) = 1$
$T(n-1) = 0$
$T(n-1) = 1$
$f_{11}(S) > 0$
$f_{11}(S) = 0$
$S(2) = 1$
$S(2) = 0$
$S(n-1) = 1$
$S(n-1) = 0$
$f_{00}(T) > 0$
$f_{00}(T) = 0$

- Case(i) $S(n) = 0$:
  then $S = 0 \cdots 1$ and $T = 1 \cdots 0$,
  so take $S' = [0 \cdots 1]$.
- Case(ii) $T(2) = 0$:
  then $S = 0 \cdots 0$ and $T = 10 \cdots 1$,
  so take $S' = [0^+1] \cdots 0$.
- Case(iii) $T(n-1) = 0$:
  then $S = 0 \cdots 0$ and $T = 11 \cdots 01$
  so consider S^R and T^R, similar to Case(ii).
- Case(iv) $f_{11}(S) > 0$:
  then $S = 0 \cdots 11 \cdots 0$ and $T = 11 \cdots 11$
  so take $S' = [0^+(10^+)^*11] \cdots 0$.

- Case(v) $S(2) = 1$:
  then $S = 01 \cdots 0$, $T = 11 \cdots 11$,
  consider $\bar{T}$ and $\bar{S}$, similar to case(ii).
- Case(vi) $S(n-1) = 1$:
  then $S = 00 \cdots 10$ and $T = 11 \cdots 11$
  consider $\bar{T}^R$ and $\bar{S}^R$, similar to case(ii)
- Case (vii) $f_{00}(T) > 0$
  then $S = 00 \cdots 00$, $T = 11 \cdots 00 \cdots 11$
  consider $\bar{T}$ and $\bar{S}$, similar to case(iv)
- Case (viii) $f_{00}(T) = 0$
  it is impossible since $S = 00 \cdots 00$, $T = 11 \cdots 11$
  $f_{11}(S) = 0$, $f_{00}(T)$=0

Upper bound

- Theorem 3.4 Let S and T be related binary strings of length n. Then

$$d_r(S,T) \leq \left\lfloor \frac{n}{2} \right\rfloor$$

- Proof.
  By lemma 3.3, we can increase the combined length of the common prefix and suffix of S and T by at least two using a single reversal.

- Example.

  $S = 010101010$ and $T = 110000011$

$$
\begin{aligned}
S &= 010101010 \\
&\rightarrow 01[01]01010 \\
&\rightarrow 01100[1010] \\
&\rightarrow 011000[10]1 \\
&\rightarrow [011]000011 \\
&\rightarrow 110000011 = T
\end{aligned}
$$

Reversal diameter of B_n

- The reversal diameter, $D_r(n)$, of B_n is defined to be the maximum value of $d_r(S,T)$ over all related binary string S and T of length n.

$$D_r(n) = \max\{\, d_r(S,T) : S, T \text{ are related binary strings } S \text{ and } T \text{ of length } n\}$$

- Lemma 3.5 $\forall k \geq 1$, $d_r(E_k, C_k) = k$ and $d_r(0 \cdot E_k, 0 \cdot C_k) = k$.

- Proof.
  Review: $E_k = 0^k 1^k$ and $C_k = (10)^k$
  By theorem 3.2 and 3.4

$$\left\lceil \frac{b_r(S,T)}{2} \right\rceil \leq d_r(S,T) \leq \left\lfloor \frac{n}{2} \right\rfloor$$

  $b_r(E_k, C_k) = 2k$, therefore $d_r(E_k, C_k) = k$.

- Theorem 3.6 $\forall n \geq 1$, $D_r(n) = \left\lfloor \frac{n}{2} \right\rfloor$

- Proof.
  By theorem 3.4 and lemma 3.5,
  $d_r(S,T) \leq \left\lfloor \frac{n}{2} \right\rfloor$ and $d_r(E_k, C_k) = k = \left\lfloor \frac{n}{2} \right\rfloor$
  done!!

- Theorem 3.7 Let S and T be related binary strings of length $2n \geq 6$. Then $d_r(S,T) = n$ if and only if S and T are isomorphic to C_n and E_n.

- Proof.
  We prove this theorem by induction.
  When n = 3, it may be verified that $d_r(S,T) = 3$ if and only if S and T are isomorphic to E_3 and C_3.

- Suppose that the theorem holds when $n \leq k$.
- Let S and T be strings of length $2k + 2$ such that $d_r(S, T) = k + 1$.
- WLOG, we suppose $S(1) = 0$, and use the similar proof in lemma 3.3.
- It must be that $lcp(S', T) + lcs(S', T) = 2$ and $d_r(S', T) = k$, since any alternative would contradict $d_r(S, T) = k + 1$.

- Let S'_e and T_e be the strings S' and T excluding any common prefix and suffix.
- By the induction hypothesis, S'_e and T_e must be isomorphic to E_k and C_k.
  There are four cases:
  (a) $S'_e = E_k$, $T_e = C_k$ (b) $S'_e = C_k$ and $T_e = E_k$
  (c) $S'_e = E_k^R$, $T_e = E_k^R$ (d) $S'_e = C_k^R$ and $T_e = E_k^R$
- By the proof of Lemma 3.3, that the reversal can be applied to S, as typified by Cases (i), (ii), and (iv) in that proof.

- Case (i): $S = 0 \cdots 1, T = 1 \cdots 0.$
  (a) $S = 0 \cdot E_k^R \cdot 1 = 0 \cdot 1^k 0^k \cdot 1$ and $T = 1 \cdot C_k \cdot 0.$
  consider $S'' = [011]1^{k-2} \cdot 0^k \cdot 1$, $d_r(S'', T) < k.$

  (b) $S = 0 \cdot C_k^R \cdot 1 = 0 \cdot (01)^k \cdot 1$ and $T = 1 \cdot E_k \cdot 0 = 1 \cdot 0^k \cdot 1^k \cdot 0.$ it's isomorphic to(a).

  (c) $S = 0 \cdot E_k \cdot 1 = E_{k+1}$ and $T = 1 \cdot C_k^R \cdot 0 = C_{k+1}$

  (d) $S = 0 \cdot C_k \cdot 1 = C_{k+1}^R$ and $T = 1 \cdot E_k^R \cdot 0 = E_{k+1}^R.$

- Case(ii): $S = 0 \cdots 0$ and $T = 10 \cdots 1$
  The reversal results in a string S' that has prefix 10. Therefore S'_e and T_e must be suffixes of S' and T.
  (b) $S' = 10 \cdot C_k = 10 \cdot (10)^k$ and $T = 10 \cdot E_k = 10 \cdot 0^k \cdot 1^k$.
  Consider $S'' = [01101]0 \cdot (10)^{k-2}$, $d_r(S'', T) < k$.
  (c) $S' = 10 \cdot E_k^R = 10 \cdot 1^k 0^k$ and $T = 10 \cdot C_k^R = 10 \cdots (01)^k$
  $\Rightarrow S = 01 \cdot 1^k 0^k$, $S'' = 01 \cdot 1^{k-1}[10^k]$, $d_r(S'', T) < k$

- Case (iv): $S = 0 \cdots 11 \cdots 0$ and $T = 11 \cdots 11$.
  (b) $S' = 11 \cdot C_k = 11 \cdot (01)^k$ and $T = 11 \cdot E_k = 11 \cdot 0^k \cdot 1^k$.
  However, then the reversal on S could not have moved the first 11 substring in S.
- Theorem 3.7 describes the strings of length n that achieve the reversal diameter when n is even. When n is odd, significantly more pairs of strings achieve the reversal diameter.

Sorting by reversals

- Let S_I denote the string that is related to S and consists only of a block of zeros, followed by a block of ones.
- Ex: $S = 01100110$, then $S_I = 00001111$
- Lemma 3.8 Let S' be a string obtained from S by a single reversal. Then

$$z(S') \geq z(S) - 1$$

Review: $z(S)$: the number of blocks of zeros contained in S.

- Theorem 3.9 For any binary string S,

$$d_r(S, S_I) = \begin{cases} z(S) - 1 & if\ S(1) = 0 \\ z(S) & otherwise \end{cases}$$

- Proof.
  By Lemma 3.8, $d_r(S, S_I) \geq z(S) - 1$.
  Case(i) if $S(1) = 0$
  then $z(S) - 1$ reversals of the form
  $0^+[1^+0^+]1 \cdots$ or $0^+[1^+0^+]$ transform S into S_I

- Proof.
  Case(ii) if $S(1) = 1$, then an extra reversal is required because it is impossible to change the first symbol to 0 and also reduce the value of z.

Transposition distance between binary strings.

- Lower bound
- Upper bound
- Transposition diameter of B_n
- Sorting by transpositions.

Breakpoints

- Transposition breakpoints are defined in a similar way to reversal breakpoints.
- The number of transposition breakpoints is

$$b_t(S,T) = \sum_{a,b \in A} \delta(f_{ab}(S) - f_{ab}(T))$$

  where $A = \{\alpha, 0, 1, \omega\}$, $\delta(x) = x$ if $x > 0$.
- A crucial difference between reversal breakpoints and transposition breakpoints is that 01 and 10 substrings are counted separately.

- Ex1: $S = T$, then $b_t(S,T) = 0$.

- Ex2: $S = 101001, T = 100101$

$$b_t(S,T) = \delta(f_{\alpha 1}(S) - f_{\alpha 1}(T)) + \delta(f_{00}(S) - f_{00}(T)) + \delta(f_{01}(S) - f_{01}(T)) + \delta(f_{10}(S) - f_{10}(T)) + \delta(f_{1\omega}(S) - f_{1\omega}(T))$$

$$= \delta(1-1) + \delta(1-1) + \delta(2-2) + \delta(2-2) + \delta(1-1) = 0$$

- Lemma 4.1 Suppose that S' is obtained from S by a single transposition. Then

$$\mathrm{b_t(S', T)} \geq \begin{cases} b_t(S,T) - 3 & if\ S(1) \neq S'(1) and\ S(n) \neq S'(n) \\ b_t(S,T) - 2 & otherwise \end{cases}$$

- Proof.
  The transposition must have the form

$$\cdots a[b \cdots c][d \cdots e]f \cdots$$
$$\rightarrow \cdots a\,[d \cdots e][b \cdots c]f \cdots$$

  Consider ab, cd, ef and ad, eb, cf.

- Claim: if $a \neq \alpha$ or $f \neq \omega$, then at least one of the substrings $ab, cd\ or\ ef$ is the same as one of the substrings $ad, eb\ or\ cf$.
- Proof.

  Suppose not, we have $a \neq e, b \neq f, c \neq a, d \neq f$.
  Consider $\alpha \neq a$, then $c = e$.
  However, $ef = cf$ (a contradiction.)
  Similar if $f \neq \omega$, then $b = d$, implies $ab = ad$.

- Case(i) a transposition moves the first and last symbol of S.
  There are at most three substrings of length two may change as a result of the transposition.
- Case(ii) don't move the first and last symbol of S
  There are at most two substrings of length two may change as a result of the transposition.

Lower bound

- Theorem 4.2 Let S and T be related binary strings of length n. Then

$$d_t(S,T) \geq \begin{cases} \lceil b_t(S,T) \rceil & if\ S(1) = T(1), or\ S(n) = T(n) \\ \lceil (b_t(S,T) - 1)/2 \rceil & otherwise \end{cases}$$

- Proof.
  By Lemma 4.1, done.

- The lower bound is not exact.
- Ex: $S = 011100110001$ and $T = 100011001110$ then the bound is 1, but $d_t(S,T) = 2$.

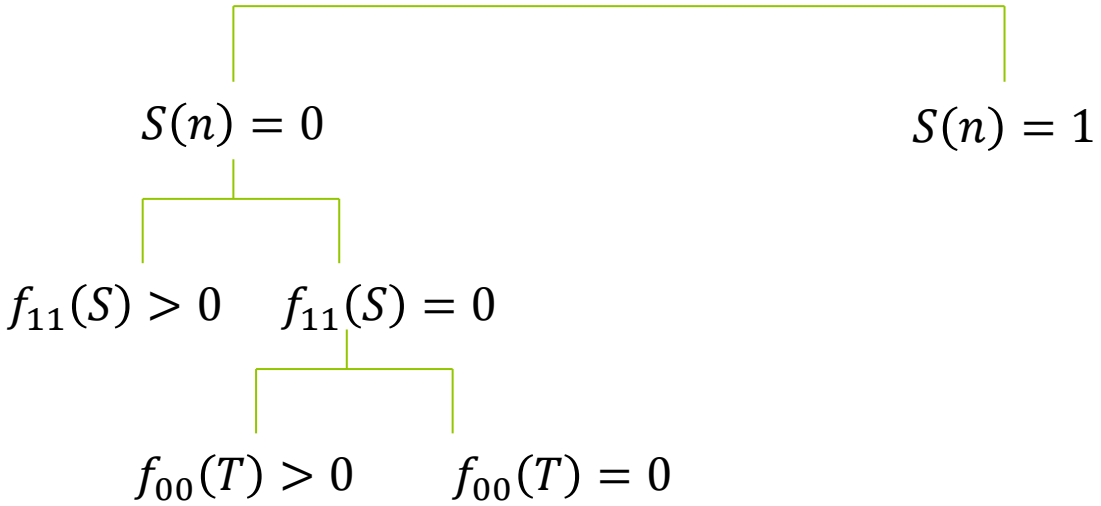
- Theorem 4.3 Let S and T be related strings of length n, over an alphabet of size > 2. Then

$$d_t(S,T) \geq \left\lceil \frac{b_t(S,T)}{3} \right\rceil$$

- Lemma 4.4 Let S and T be related strings of length n such that $S \neq T$. Then it is possible either
  (a) to apply a transposition to S resulting in a string S' such that
  $lcp(S', T) + lcs(S', T) \geq lcp(S, T) + lcs(S, T) + 2$ or
  (b) to apply a transposition to T resulting in a string T' such that
  $lcp(S, T') + lcs(S, T') \geq lcp(S, T) + lcs(S, T) + 2$

Proof.

- WLOG, assume that $S(1) = 0$
- $S(1) \neq T(1)$, $S(n) \neq T(n)$
- The following four cases are described.



- Case(i) $S(n) = 1$:
  then $S = 0 \cdots 1$ and $T = 1 \cdots 1$,
  so take $S' = [0^+][1 \cdots]$.
- Case(ii) $f_{11}(S) > 0$:
  then $S = 0 \cdots 11 \cdots 0$ and $T = 1 \cdots 1$,
  so take $S' = [0^+(10^+)^*1][1 \cdots 0]$.
- Case(iii) $f_{00}(T) > 0$:
  then $S = 0 \cdots 0$ and $T = 1 \cdots 00 \cdots 1$,
  consider $\bar{T}$ and $\bar{S}$, similar to Case(ii)
- Case(iv)
  then $S = 0 \cdots 0$ and $T = 1 \ldots 1$ and $f_{11}(S) = 0$, $f_{00}(T) = 0$, it is impossible since S is related to T.

Upper bound

- Theorem 4.5 Let S and T be related binary strings of length n. Then

$$d_t(S,T) \leq \left\lfloor \frac{n}{2} \right\rfloor$$

- Proof.
  By Lemma 4.4, we can increase the combined length of the common prefix and suffix of S and T by at least two using a single transposition.

Transposition diameter of B_n

- The transposition diameter, $\mathrm{D}_t(n)$, of B_n is the maximum value of $d_t(S,T)$ taken over all related binary strings of length n.

$$D_t(n) = \max\{\, d_t(S,T) : S,T \text{ are related binary strings of length } n \,\}$$

- Lemma 4.6 $\forall k \geq 1, d_t(E_k, C_k) = k \text{ and } d_t(0 \cdot E_k, 0 \cdot C_k) = k$
- Proof. By Theorem 4.2 and 4.5.
- Theorem 4.7 $\forall n \geq 1, D_t(n) = \left\lfloor \frac{n}{2} \right\rfloor$
- Proof. By Theorem 4.5 and Lemma 4.6.

- Theorem 4.8 Let S and T be related binary strings of length $2n \geq 4$. Then $d_t(S,T) = n$ if and only if S and T are isomorphic to C_n and E_n
- Proof.
  We prove it by induction.
  When n=2, it may be verified that $d_t(S,t) = 2$ if and only if S and T are isomorphic to E_2 and C_2.

- Suppose that the theorem holds when $n \leq k$.
- Let S and T be strings of length $2k + 2$ such that $d_t(S, T) = k + 1$.
- WLOG, $S(1) = 0$, and use the similar proof in Lemma 4.4.
- It must be that $lcp(S', T) + lcs(S', T) = 2$ and $d_t(S', T) = k$.

- Let S'_e and T_e be the string S' and T excluding any common prefix and suffix.
- By induction hypothesis, S'_e and T_e must isomorphic to E_k and C_k.
  There are four case:
  (a) $S'_e = E_k$, $T_e = C_k$ (b) $S'_e = C_k$ and $T_e = E_k$
  (c) $S'_e = E_k^R$, $T_e = E_k^R$ (d) $S'_e = C_k^R$ and $T_e = E_k^R$
- By the proof of Lemma 4.4, the transposition can be applied to S, as typified by Case (i) and (ii) in that proof.

- Case(i) $S = 0 \cdots 1, T = 1 \cdots 0.$
  (a) $S' = 1 \cdot E_k \cdot 0 = 1 \cdot 0^k \cdot 1^k \cdot 0$ and $T = 1 \cdot C_k \cdot 0 = 1 \cdot (01)^k \cdot 0.$
  Therefore $S = 01 \cdot 0^k \cdot 1^k$, $S'' = 0[100][0^{k-2} \cdot 1^k]$, $d_r(S'', T) < k.$
  (b) $S' = 1 \cdot C_k \cdot 0 = 1 \cdot (01)^k \cdot 0$ and $T = 1 \cdot E_k \cdot 0 = 1 \cdot 0^k \cdot 1^k \cdot 0.$
  Then $S = 001 \cdot (10)^{k-1} \cdot 1$, $S'' = [001 \cdot (10)^{k-1}][1]$, $d_r(S'', T) < k.$

(c) $S' = 1 \cdot E_k^R \cdot 0 = E_{k+1}^R$ and
$T = 1 \cdot C_k^R \cdot 0 = C_{k+1} \Rightarrow S = E_{k+1}$.

(d) $S' = 1 \cdot C_k^R \cdot 0 = C_{k+1}$ and
$T = 1 \cdot E_k^R \cdot 0 = E_{k+1}^R \Rightarrow S = C_{k+1}^R$.

- Case(ii): $S = 0 \cdots 11 \cdots 0$, $T = 1 \cdots 1$

(c) $S' = 1 \cdot E_k^R \cdot 1 = 1 \cdot 1^k \cdot 0^k \cdot 1$ and
$T = 1 \cdot C_k^R \cdot 1 = 1 \cdot (01)^k \cdot 1$

Then $S = 0^+ \cdot 1^{k+2} \cdot 0^+$, $S'' = 0^+ \cdot 1^k[11][0^+]$, $d_t(S'', T) < k$.

Sorting by transposition

- Lemma 4.9 Let S' be a string obtained from S by a single transposition. Then
$$z(S') \geq z(S) - 1$$

- Theorem 4.10 For any binary string S,
$$d_t(S, S_I) = \begin{cases} z(s) - 1 & if\ S(1) = 0 \\ z(S) & otherwise \end{cases}$$

- Proof.
  By Lemma 4.9, $d_t(S, S_I) \geq z(S) - 1$.
  Case(i) $S(1) = 0$
  $z(S) - 1$ transpositions of the form
  $0^+[1^+][0^+]1\cdots$ or $0^+[1^+][0^+]$ transform S into S_I

  Case(ii) $S(1) = 1$
  An extra transposition is required.

NP-completeness of reversal distance

- We begin with a definition of the reversal distance problem as a decision problem:
  Instance: Related strings S and T of length n, over an alphabet of size $t \geq 2$, and a bound $d \in Z^+$.
  Question: Is $d_r(S,T) \leq d$?
- Theorem 5.1 The above problem is NP-complete, even when $t = 2$.
- Proof:
  Use 3-Partition problem to prove it.

- 3-partision:
  Instance: A set A of $3m$ elements, a bound $B \in Z^+$, and a size $s(a) \in Z^+$ for each $a \in A$ such that $B/4 < s(a) < B/2$ and such that $\sum_{a \in (A)} s(a) = mB$.
  Question: Can A partitioned into m disjoint sets $A_1, A_2, \cdots, A_m$ such that, for $1 \leq i \leq m$, $\sum_{a \in A_i} s(a) = B$?

Conclusion

- Find the reversal distance between two strings is NP-hard, even when the strings re drawn from a binary alphabet.
- The complexity of find the transposition between two strings remains open.
- The lower bound, upper bound and diameter of B_n of reversal and transposition.

Reference

[1] V. Bafna and P. A. Pevzner, *Genome rearrangements and sorting by reversals*, SIAM J. Comput., 25 (1996), pp. 272–289.

[2] V. Bafna and P. A. Pevzner, *Sorting by transpositions*, SIAM J. Discrete Math., 11 (1998), pp. 224–240.

[3] P. Berman and S. Hannenhalli, *Fast sorting by reversals*, in Combinatorial Pattern Matching, Lecture Notes in Comput. Sci. 1075, Springer, Berlin, 1996, pp. 168–185.

[4] A. Caprara, *Sorting by reversals is difficult*, in Proceedings of the First International Conference on Computational Molecular Biology (RECOMB'97), ACM Press, Santa Fe, NM, 1997, pp. 75–83.

[5] A. Caprara, *Sorting permutations by reversals and Eulerian cycle decompositions*, SIAM J. Discrete Math., 12 (1999), pp. 91–110.

[6] D. A. Christie, *Sorting permutations by block-interchanges*, Inform. Process. Lett., 60 (1996), pp. 165–169.

[7] D. A. Christie, *A 3/2-approximation algorithm for sorting by reversals*, in Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, San Francisco, 1998, pp. 244–252.

[8] D. A. Christie, *Genome Rearrangement Problems*, Ph.D. thesis, Department of Computing Science, University of Glasgow, Glasgow, Scotland, 1998.

[9] M. R. Garey and D. S. Johnson, *Complexity results for multiprocessor scheduling under resource constraints*, SIAM J. Comput., 4 (1975), pp. 397–411.

[10] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, San Francisco, 1979.

[11] S. Hannenhalliand P. A. Pevzner, *Transforming cabbage into turnip (polynomial algorithm for sorting signed permutations by reversals)*, in Proceedings of the 27th Annual ACM Symposium on Theory of Computing, Las Vegas, 1995, pp. 178–189.

[12] S. Hannenhalliand P. A. Pevzner, *Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals*, J. ACM, 46 (1999), pp. 1–27.

[13] H. Kaplan, R. Shamir, and R. E. Tarjan, *Faster and simpler algorithm for sorting signed permutations by reversals*, in Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, New Orleans, 1997, pp. 344–351.

[14] J. Kececioglu and D. Sankoff, *Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement*, Algorithmica, 13 (1995), pp. 180–210. Downloaded