The Number of Distinct Alignments of Two Strings

Michael A. Covington

Artificial Intelligence Center

The University of Georgia

Athens, Georgia 30602–7415 U.S.A.

E-mail: mc@uga.edu

Telephone: +1 706 542-0359

Last revised 2003 February 17

1 Introduction

An alignment is a way of pairing up elements of two strings, optionally skipping some elements but preserving the order. For example,

```
a b c        a b c          a b c
| | |          | |          |   |
x y z          x y z        x   y z
```

are three of the many alignments of `abc` with `xyz`. For brevity we will write them as $\binom{\texttt{abc}}{\texttt{xyz}}$, $\binom{\texttt{abc-}}{\texttt{-xyz}}$, and $\binom{\texttt{abc-}}{\texttt{x-yz}}$ respectively. Hyphens mark skips, i.e., places where an element of one string is paired with nothing in the other string.

This paper presents formulae for the number of distinct alignments of two strings of lengths m and n, using various criteria of distinctness. These numbers are of practical interest because they give the size of the search space for inexact string matching (Sankoff and Kruskal 1983, Ukkonen 1985), DNA sequence alignment (Waterman 1995), and the first step in comparative reconstruction of ancient languages (Kay 1964, Covington 1996, 1998).

The last of these is of greatest interest to the linguist. For example, to demonstrate the relatedness of Latin *dō* 'I give' to Greek *didōmi* 'I give', one has to align them as

```
- - d ō - -
d i d ō m i
```

and not

```
d ō - - - -
d i d ō m i

d - - ō - -
d i d ō m i

- - - - d ō
d i d ō m i
```

or numerous other possibilities. The segments of two words may be misaligned because of affixes (living or fossilized), reduplication, and sound changes that alter the number of segments, such as elision or monophthongization. An algorithm to find the best alignment in such situations is given by Covington (1996); this paper works out the size of the search space.[1]

2 Definitions

For some purposes some alignments are equivalent to others. Consider for example the three alignments $\binom{\texttt{abc}}{\texttt{xyz}}$, $\binom{\texttt{a-bc}}{\texttt{xy-z}}$, and $\binom{\texttt{ab-c}}{\texttt{x-yz}}$. They can be handled three ways:

- Treat all three as equivalent because the differences between them do not involve any shifting left or right. All three alignments match `a` with `x` and `c` with `z`. In the middle of the string, they either match `b` with `y` or skip `b` and `y` in tandem, which amounts to the same thing.

- Treat $\binom{\texttt{a-bc}}{\texttt{xy-z}}$ and $\binom{\texttt{ab-c}}{\texttt{x-yz}}$ as equivalent to each other but distinct from $\binom{\texttt{abc}}{\texttt{xyz}}$. That is, skipping two elements in tandem is distinct from matching them, but it does not matter in which order the skips are performed.

- Treat all three as distinct.

These methods of counting (which we will make more precise below) yield what we will call the SMALL SET, MIDDLE SET, and LARGE SET of alignments of any given pair of strings. We

[1] One might think that the dynamic programming algorithm of Ukkonen (1985) makes the size of the search space more or less irrelevant. That is not the case because dynamic programming is not applicable to all searches for alignments. It assumes that the "badness" of any alignment is the sum of the badness of its individual elements, and that only the one best alignment is wanted. Sometimes, however, more than one alignment is wanted, or "badness" must be computed in some other way (e.g., assigning a lower penalty for transpositions than for random mismatches); see Covington (1996, 1998).

I want to thank E. Rodney Canfield for extensive help getting started with this project. I am indebted to many colleagues with whom I discussed the problem; among them are Jeff Clark, Jan Willem Nienhuys, Oscar Lanzi III, Les Reid, and other participants in `sci.math` on the Internet.

will call their cardinalities $a(m,n)$, $\textsc{a}(m,n)$, and $A(m,n)$, respectively, where m and n are the lengths of the strings.

The large set and middle set contain some NULL ALIGNMENTS that do not match up any elements at all, such as $\binom{\texttt{-a-b-c}}{\texttt{x-y-z-}}$ and $\binom{\texttt{---abc}}{\texttt{xyz---}}$. The small set contains no null alignments.

3 Enumerating alignments

Any alignment can be constructed by working through the two strings step by step until all the elements of both strings have been consumed. At each step, make one of three moves:

- MATCH (accept elements from both strings together);
- SKIP-1 (accept an element from the first string only);
- SKIP-2 (accept an element from the second string only).

The number of alignments is the number of different ways this can be done.

4 The large set

The large set of alignments is the easiest to enumerate because it imposes no further restrictions on the construction procedure.

Let m and n be the lengths of the two strings. Recall that a match consumes an element from both strings while a skip consumes an element from one string but not the other. Every complete alignment has to consume all the elements, so any alignment containing k matches must also contain $m-k$ skips on string 1 and $n-k$ skips on string 2.

The number of matches k in turn ranges from 0 to $\min(m,n)$. Thus, the number of possible alignments $A(m,n)$ is:

$$A(m,n) = \sum_{k=0}^{\min(m,n)} \text{number of alignments containing } k \text{ matches}$$

The number of alignments containing k matches is simply the number of ways of partitioning a set of $k + (m - k) + (n - k) = m + n - k$ moves into k matches, $m - k$ skips on string 1, and $n - k$ skips on string 2:

$$A(m, n) = \sum_{k=0}^{\min(m,n)} \frac{(m + n - k)!}{k!(m - k)!(n - k)!}$$

This is the function f of Waterman (1995, p. 187). Values are shown in Table 1. Throughout, $A(m, n) = A(n, m)$. The main diagonal of the matrix is the integer sequence known as M2942 in Sloane and Plouffe (1995).

5 The small set

The small set of alignments treats $\binom{\texttt{abc}}{\texttt{xyz}}$, $\binom{\texttt{a-bc}}{\texttt{xy-z}}$, and $\binom{\texttt{ab-c}}{\texttt{x-yz}}$ as equivalent. That is tantamount to disallowing the latter two because they contain ALTERNATING SKIPS (places where a skip in one string is followed immediately by a skip in the other). More generally, the small set of alignments consists of exactly those that do not contain any alternating skips.

When alternating skips are disallowed, the number of alignments (call it $a(m, n)$, with small a denoting the small set) is no longer a simple partitioning problem because a move can restrict the choice of moves that can be taken next. Specifically, SKIP-2 cannot follow SKIP-1, nor vice versa.

The problem is conveniently approached by constructing a search tree. Figure 1 show a search tree for enumerating all the alignments of `abc` with `xyz`. The tree as shown is not complete; rather, it invokes recursion in the places where all the alignments of two substrings are needed. Note that, of the three possible moves at each juncture, only MATCH leaves the choice of subsequent moves unrestricted. The only places in the tree where recursion can be invoked are therefore the places where matches have been made.

Redrawing and substituting recursive calls to $a(m, n)$ to get the number of alignments, we get the tree in Figure 2 and the equation:

$$a(3, 3) = a(2, 2) + [a(2, 1) + a(2, 0)] + [a(1, 2) + a(0, 2)]$$

Table 1: $A(m, n)$: number of alignments of strings of lengths m and n not discarding any as equivalent to others.

		n										
		0	1	2	3	4	5	6	7	8	9	10
m	0	1										
	1	1	3									
	2	1	5	13								
	3	1	7	25	63							
	4	1	9	41	129	321						
	5	1	11	61	231	681	1683					
	6	1	13	85	377	1289	3653	8989				
	7	1	15	113	575	2241	7183	19825	48639			
	8	1	17	145	833	3649	13073	40081	108545	265729		
	9	1	19	181	1159	5641	22363	75517	224143	598417	1462563	
	10	1	21	221	1561	8361	36365	134245	433905	1256465	3317445	8097453

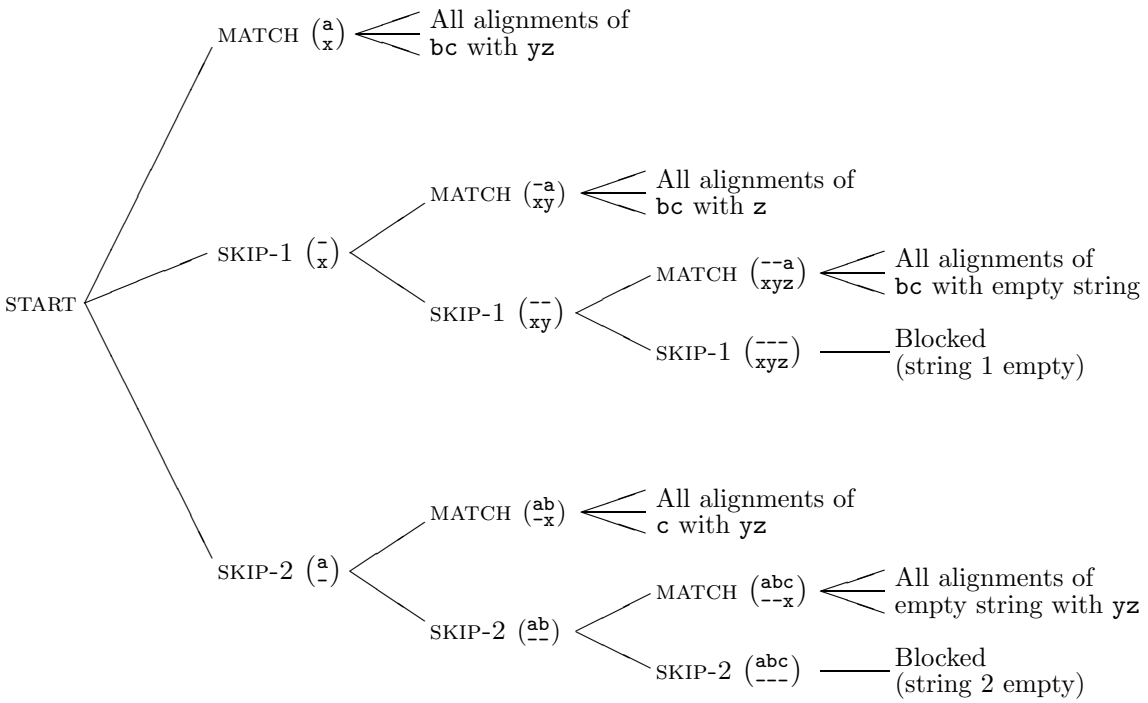

Figure 1: Search tree for aligning `abc` with `xyz` without alternate skips.

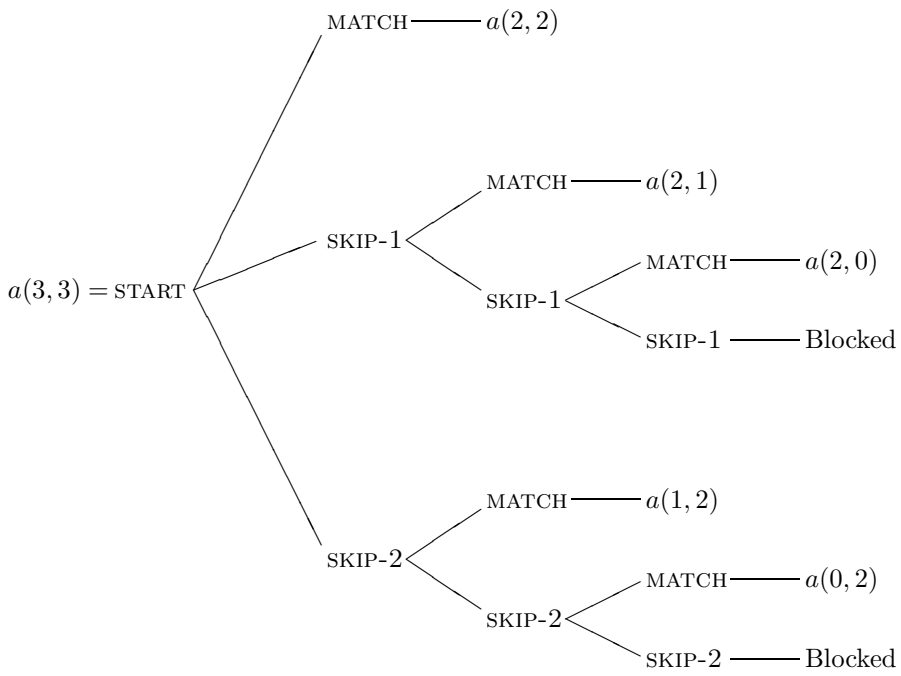

Figure 2: Search tree redrawn to show formulae.

where the bracketed terms reflect the whittling away of string 2 and string 1, respectively, by skips. Here $a(2,0) = a(0,2) = 1$, because the only way to align with an empty string is to skip all the elements of the non-empty string and quit. Likewise, for 4-element strings we get:

$$a(4,4) = a(3,3) + [a(3,2) + a(3,1) + a(3,0)] + [a(2,3) + a(1,3) + a(0,3)]$$

or in general

$$a(m,n) = a(m-1,n-1) + \sum_{i=0}^{n-2} a(m-1,i) + \sum_{i=0}^{m-2} a(i,n-1)$$

with the initial conditions $a(0,n) = a(m,0) = 1$. The three terms represent, respectively, the case where a match is made on the first element; the case where string 2 is whittled away by SKIP-1's; and the case where string 1 is whittled away by SKIP-2's. Table 2 gives computed values, which have been checked by actually enumerating the alignments using a Prolog program.

6 The middle set

The middle set of alignments treats $\binom{\texttt{a-bc}}{\texttt{xy-z}}$ and $\binom{\texttt{ab-c}}{\texttt{x-yz}}$ alike but distinguishes them from $\binom{\texttt{abc}}{\texttt{xyz}}$. The crucial distinction here is that $\binom{\texttt{abc}}{\texttt{xyz}}$ matches `b` with `y`, but $\binom{\texttt{a-bc}}{\texttt{xy-z}}$ and $\binom{\texttt{ab-c}}{\texttt{x-yz}}$ do not; instead they skip `b` and `y` in tandem.

Compared to the small set, the middle set introduces the concept of MISMATCH or DOUBLE SKIP: you can take an element from each of the two strings simultaneously without matching them.[2]

Because double skips are permitted in exactly the same situations as matches, their effect is to double the count at each of the leaf nodes of the search tree in Figures 1 and 2. (Imagine

[2]At first sight it might seem that the middle set could be analyzed by allowing alternating skips in only one direction, e.g., letting a SKIP-2 follow a SKIP-1 but not vice versa. That turns out not to be true. The alignment $\binom{\texttt{a-b-}}{\texttt{-x-y}}$ consists of a SKIP-2, a SKIP-1, a SKIP-2, and another SKIP-1, but it is a legitimate member of the middle set because it first skips `a` and `x` in tandem, then skips `b` and `y` in tandem.

Table 2: $a(m, n)$: number of alignments of strings of lengths m and n disalllowing alternating skips.

		n										
		0	1	2	3	4	5	6	7	8	9	10
m	0	1										
	1	1	1									
	2	1	2	3								
	3	1	3	5	9							
	4	1	4	8	15	27						
	5	1	5	12	24	46	83					
	6	1	6	17	37	75	143	259				
	7	1	7	23	55	118	237	450	817			
	8	1	8	30	79	180	380	755	1429	2599		
	9	1	9	38	110	267	592	1229	2421	4570	8323	
	10	1	10	47	149	386	899	1948	3989	7804	14698	26797

Table 3: $\mathrm{A}(m, n)$: number of alignments of strings of lengths m and n alllowing skips, matches, and double skips.

		n										
		0	1	2	3	4	5	6	7	8	9	10
m	0	1										
	1	1	2									
	2	1	4	8								
	3	1	6	16	36							
	4	1	8	28	72	164						
	5	1	10	44	132	336	764					
	6	1	12	64	224	636	1592	3620				
	7	1	14	88	356	1128	3092	7632	17356			
	8	1	16	116	536	1892	5664	15116	36920	83956		
	9	1	18	148	772	3024	9868	28392	74244	179856	408956	
	10	1	20	184	1072	4636	16456	50932	142240	366108	881080	2003204

a DOUBLE SKIP alongside every MATCH.) This, in turn, doubles the recurrence expression:

$$\text{A}(m,n) = 2 \times (\text{A}(m-1,n-1) + \sum_{i=0}^{n-2} \text{A}(m-1,i) + \sum_{i=0}^{m-2} \text{A}(i,n-1))$$

where $a(0,n) = a(m,0) = 1$ just as before. Because the factor of 2 is within the recurrence, the actual number of alignments is more than doubled compared to the small set. Computed values, verified by actual enumeration, are shown in Table 3.

This middle set of alignments is the best model of the search space for string matching as generally conceived (e.g., by Ukkonen 1985), where the available operations are *insertion* (= SKIP-2), *deletion* (= SKIP-1), *substitution* (= DOUBLE SKIP), and exact matching (= MATCH).

7 A different middle set

Suppose double skips are applied to substrings rather than individual elements. In that case, $\binom{\texttt{a---bcd}}{\texttt{vwxy--z}}$ is equivalent to $\binom{\texttt{ab-c--d}}{\texttt{v-w-xyz}}$ and $\binom{\texttt{abc---d}}{\texttt{v--wxyz}}$ among others; all of these represent the alignment

```
a b  c  d
|       |
v w x y z
```

in which the substrings `bc` and `wxy` are skipped in tandem.

In this situation, an alignment is defined entirely by picking the k elements of each string that are going to be matched. They can only be matched in the order in which they occur, so no further information about order is needed. Accordingly, much as with the large set,

$$\text{A}'(m,n) = \sum_{k=0}^{\min(m,n)} \text{number of alignments containing } k \text{ matches}$$

and in this case:

$$\text{A}'(m,n) = \sum_{k=0}^{\min(m,n)} \binom{m}{k} \binom{n}{k}$$

That is, for each k, choose k elements from the first string, and for each such combination, choose k elements from the second string.[3]

This simplifies mathematically to

$$\textsc{a}'(m,n) = \left(\begin{array}{c} m+n \\ m \end{array} \right) = \left(\begin{array}{c} m+n \\ n \end{array} \right)$$

which can be explained intuitively as follows. Recall that the task is to choose, for some k, a set of k matched elements in each string. Equivalently, we can choose the matched elements in one string and the skipped elements in the other (leaving the rest to be matched). In that case we are choosing k elements from the first string and $n-k$ elements from the second. Regardless of the value of k, this amounts to choosing n elements from the complete set of $m+n$ elements, and the number of ways of doing it is, by definition:

$$\left(\begin{array}{c} m+n \\ n \end{array} \right) = \left(\begin{array}{c} m+n \\ (m+n)-n \end{array} \right) = \left(\begin{array}{c} m+n \\ m \end{array} \right)$$

The numbers here are considerably smaller than those in Table 3. For example, $\textsc{a}(10,10) = 2,003,204$, but $\textsc{a}'(10,10) = \binom{20}{10} = 184,756$. This is the function g of Waterman (1995, pp. 188–189)[4] and of Ewens and Grant (2001, pp. 191–192).

Bibliography

Covington, Michael A. (1996). An algorithm to align words for historical comparison. *Computational Linguistics, 22,* 481–496.

Covington, Michael A. (1998). Alignment of multiple languages for historical comparison. In *COLING-ACL '98: 36th Annual Meeting of the Association for Computational*

[3] The notation $\binom{a}{b}$ denotes the number of combinations of a things taken b at a time, equal to $\frac{a!}{b!(a-b)!}$.

[4] The expression $\binom{n+m}{k}$ at the top of Waterman's page 189 is apparently a misprint; it should be $\binom{n+m}{n}$.

Linguistics and 17th International Conference on Computational Linguistics — Proceedings of the Conference (pp. 275–280). San Francisco: Kaufmann.

Ewens, Warren J. & Grant, Gregory R. (2001). *Statistical methods in bioinformatics.* New York: Springer.

Kay, Martin. (1964). *The logic of cognate recognition in historical linguistics.* Memorandum RM-4224-PR. Santa Monica, CA: The RAND Corporation.

Sankoff, David & Kruskal, Joseph B., eds. (1983). *Time warps, string edits, and macromolecules: The theory and practice of sequence comparison.* Reading, MA: Addison-Wesley.

Sloane, N. J. A. & Plouffe, Simon. (1995). *The encyclopedia of integer sequences.* San Diego, CA: Academic Press.

Ukkonen, Esko. (1985). Algorithms for approximate string matching. *Information and Control, 64,* 100–118.

Waterman, Michael S. (1995). *Introduction to computational biology.* London: Chapman & Hall.