Reading Research

Riley Primeau — 700510762

November 16, 2022

1 Exercise in 7.1

1. Is it a good idea to write a program that plays the classic game of tic-tac-toe with the human user by storing all possible positions on the game's $3 * 3$ board along with the best move for each of them?

   It is not a bad idea to create a program using that method but other methods would be more efficient. The possible moves in a game of tic tac toe would be $9! = 362880$. This is feasible with our modern machines but other methods would be much more efficient. Additionally, if you wanted to expand the game beyond classic rules, this method would not hold up as the game board size increases.

2 Exercises 7.2

1. Consider the problem of searching for genes in DNA sequences using Hor-spool's algorithm. ADNA sequence is represented by a text on the alphabet A, C, G, T, and the gene or gene segment is the pattern.

   (a) Construct the shift table for the following gene segment of your chromo-some 10:

Character	Shift Value
T	1
C	2
A	5
G	10

(b) Apply Horspool's algorithm to locate the above pattern in the following DNA sequence:
TTATAGATCTCGTATTCTTTTATAGATCTCCTATTCTT

Makes 2 comparisons, then shifts pattern 1 indices forward
Makes 1 comparisons, then shifts pattern 2 indices forward
Makes 2 comparisons, then shifts pattern 1 indices forward
Makes 1 comparisons, then shifts pattern 2 indices forward
Makes 8 comparisons, then shifts pattern 1 indices forward
Makes 3 comparisons, then shifts pattern 2 indices forward
Makes 3 comparisons, then shifts pattern 1 indices forward
Makes 1 comparisons, then shifts pattern 2 indices forward
Makes 2 comparisons, then shifts pattern 1 indices forward
Makes 1 comparisons, then shifts pattern 2 indices forward
Makes 1 comparisons, then shifts pattern 1 indices forward
Makes 2 comparisons, then shifts pattern 2 indices forward
Makes 1 comparisons, then shifts pattern 1 indices forward
Makes 10 comparisons, indicating that the pattern was found
Returns "Pattern occurs at index 28"

2. For searching in a text of length n for a pattern of length $m(n >= m)$ with Horspool's algorithm, give an example of

   (a) worst-case input
   To be expected, the best case scenario would occur if the pattern is found at the very beginning of the text. This would mean no shifts would be necessary which would give the algorithm a time complexity of $O(m)$.

   (b) best-case input
   The worst case scenario would occur from searching in a text that repeats the first $m-1$ letters of the pattern you are searching for and the mismatched character landed on has a shift value of 1. This would result in a time complexity of $O(n(m-1))$ or $O(nm)$.

3. If Horspool's algorithm discovers a matching substring, how large a shift should it make to search for a next possible match?

   If Horspool's finds a matching substring, it means that it has successfully matched m number of characters. Therefore it should shift the length of the substring, m, in order to search a new section of the text for the next possible match.

4. Booyer-Moore Algorithm

   (a) Would the Boyer-Moore algorithm work correctly with just the bad-symbol table to guide pattern shifts?
   Yes the algorithm would work correctly with just the bad-symbol table. It could be less efficient though, depending on the input.

   (b) Would the Boyer-Moore algorithm work correctly with just the good-suffix table to guide pattern shifts?
   No it would not work correctly with only a good suffix table. In this scenario the algorithm would not be able to advance if it initially encounters characters that don't match.

5. Implement Horspool's algorithm, the Boyer-Moore algorithm, and the brute-force algorithm of Section 3.2 in the language of your choice and run an experiment to compare their efficiencies for matching

   (a) random binary patterns in random binary texts.
   Brute-force: Occurrences: 5 Time: 0.00644980
   Boyer-Moore: Occurrences: 5 Time: 0.0006224
   Horspools: Occurrences: 5 Time: 0.0000328

   (b) random natural-language patterns in natural-language texts.
   Brute-force: Occurrences: 5 Time: 0.415964
   Boyer-Moore: Occurrences: 5 Time: 0.0046918
   Horspools: Occurrences: 5 Time: 0.0069328

   My results reinforced that brute force is the least efficient approach when it comes to pattern matching. Horspool's and the Boyer-Moore method were quick overall with Horspool out performing Boyer-Moore in the task of searching binary texts. However, the Boyer-Moore method performed better when it came to natural language patterns, likely due to its utilization of suffixes.