

# Search problems: Pattern Matching

Daniel F. Simola

12 May 2005

## 1 Search problems

Grammar is very relevant to computational biology. Genomic data consists of strings created from alphabets (eg  $\Sigma = \{A, T, C, G\}$  is the alphabet of nucleotides). Grammar models are used as the theoretical tools behind things like using context-free grammars to describe RNA secondary structure and hidden Markov models. There is also a relationship between regular grammars and Markov models.

More specifically one might be interested in performing some kind of search over a data set. There are three basic types of search problems you may encounter:

**Pattern:** Given a string and a pattern, find instances of the pattern in the string. A pattern is a regular expression.

**Syntax:** Find instances of the use of a particular rule in the string. Commonly in biology this means something like “find protein coding regions”.

**Semantic:** This is content searching, and is most abstract, since you are searching for meaning, which must be manually added to strings. A query would be something like “find a sequence that codes for a growth factor and acts as a ligand for protein X”.

### 1.1 Pattern matching

#### 1.1.1 Naive algorithm

Slide along all indices of input string  $S$  with a substring  $s$ . Complexity:  $O(|S||s|)$

#### 1.1.2 Z algorithm

A z-box indicates locations of the substrings that match the prefix of the string.  $Z_i(S)$  is the length of the longest substring of  $S$ , starting at index  $i$ , where the substring is a prefix of  $S$ .

Algorithm: Find all instances of  $P$  in  $S$

1.  $|P| = m, |S| = n$
2. Construct a new string  $T = P\%S$ ,  $\%$  is novel symbol  $\notin P, S$
3. Compute Z-boxes efficiently in  $O(m+n)$ ;  $P$  occurs wherever  $Z_i = m$ .

### Possible Cases

1. We have no prior information on the current position -> try to match to prefix

ACAC#GGCACACATTACG

2. We have prior information because we know that the current position is inside a previously computed "z-box" -> look back to the z-scores at the prefix

a. prefix z-score is less than remaining length of the search pattern

ACAC#GGCACACATTACG  
0020

This means that there is no way that a "ACAC" matching pattern can start from the 2<sup>nd</sup> position!

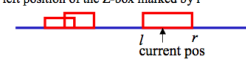
b. prefix z-score is identical to remaining length of the search pattern: Try to see if the match can be extended

ACAC#GGCACACATTACG  
0020

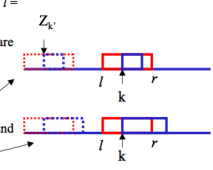
This means that there is a possibility that a match to "ACAC" could be started from the 3<sup>rd</sup> position!

### Computing Z-boxes

Variables  
 $Z(S)$ : As defined  
 $r$ : right most position of any Z-box starting left of current position  
 $l$ : left position of the Z-box marked by  $r$

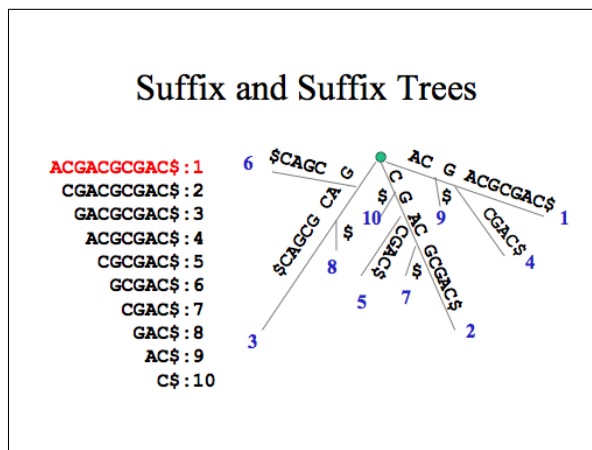


- Compute  $Z_2$  by comparing  $S[2\dots]$  to  $S[1\dots]$ , if  $Z_2 > 0$   $l = 2$ ,  $r = Z_2 + l - 1$ , else  $r = l = 0$
- Suppose we are at position  $k$ , then assess whether we are inside of a Z-box by comparing  $k$  to  $r$ . If outside, do normal comparison to compute  $Z_k$
- If inside, that means the current position matches beginning of the string at  $k' = k - l + 1$  position. Examine  $Z_{k'}$ . If it is less than  $r - k + l$ , then  $Z_k = Z_{k'}$  and  $r$  and  $l$  do not change. If it is greater, then extend by comparing  $S[r+1\dots]$  to  $S[r-k+2\dots]$ . Set  $Z_k$ ,  $l$  and  $r$  accordingly



### 1.1.3 Suffix trees

- List all suffixes of string  $S$
- Draw root and an edge with the longest suffix
- Consider the next longest suffix. If it is a substring of an existing suffix, store the index of end of current suffix in the existing suffix. Else make a new branch with current suffix.
- Repeat



Complexity for constructing a suffix tree:  $O(S)$  Complexity for finding a substring in a suffix tree:  $O(Pn)$ ,  $n$  = number of occurrences of  $P$

## 1.2 Syntactic and semantic matching

This gets into neat things like stochastic models, which are covered elsewhere...