Parameters:

\( B \) = # of items per disk block

\( M \) = # of items that fit into main memory

Cost model: Only I/Os (block moves between main memory and disk) are counted.
Exercise

Question
What is the I/O complexity for count queries using
- suffix trees?
- suffix arrays?
- FM-indexes?
The String-B Tree

Problem
Given a set \( S = \{s_0, \ldots, s_{N-1}\} \) of \( N \) strings of total length \( n \). A prefix count query asks for the subset of strings which starts with a given pattern \( P \). Devise a data structure that answers prefix count queries efficiently in the external memory model.

Solution (Ferragina & Grossi, SODA 1996)
String-B Trees (SBT) is a combination of B-trees and Patricia tries. It allows prefix count queries in \( O(\frac{|P|}{B} + \log_B n) \).

- Keys in the SBT are pointers to the text (stored in external memory)
- Keys are stored at leaves; inner nodes store copies of a subset of keys
- Key ordering is determined by lexicographical ordering of the corresponding strings
The String-B Tree

- Each SBT node $v$ is stored in a disk block and contains an ordered string set $S_v \subset S$
- $b \leq S_v \leq 2b$ for $b \in \Theta(B)$
- Denote the leftmost (rightmost) string in $S_v$ by $L(v)$ ($R(v)$)
- Construction:
  - Partition $S$ into groups of $b$ strings (last group may have up to $2n$ strings)
  - Each group is mapped to a leaf such that a left-to-right scan of the SBT gives the strings in lex. order.
  - The longest common prefix $lcp(S_j, S_{j+1})$ is associated with each pair.
  - Each internal node $v$ has $d(v)$ children $u_0, \ldots, u_{d(v)-1}$, with $b/2 \leq d(v) \leq b$ (root node may have 2 to $b$ children).
  - Set $S_v$ is formed by copying the leftmost and rightmost strings of $v$'s children. I.e. $S_v = \{L(u_0), R(u_0), \ldots, L(u_{d(v)-1}), R(u_{d(v)-1})\}$. 
Example

i think that the ruddy widow really wants ripe watermelon and red roses when winter arrives
Example

i think that the ruddy widow really wants ripe watermelon and red roses when winter arrives
String set $S_v$ is represented as a **blind trie**

Build a patricia trie over the binary strings of $S_v$

Example:

blind trie is stored in a succinct tree representation

blind trie matching does not require further IOs

follow blind search until reaching a leaf

check if $P$ occurs at the pointer returned by the blind search in the leaf
Time and space

Time

- Count queries: $\mathcal{O}(|P|/B + \log_B N)$ (assuming that subtree sizes are stored in nodes)
- Locate queries: $\mathcal{O}((|P| + Z)/B + \log_B N)$, where $Z$ is the number of strings in $S$ which are prefixed by $P$

Space

$\mathcal{O}(N/B)$ leaves. Summing up all inner nodes results in $\mathcal{O}(N/B)$ nodes in the SBT. Each node fits into a disk block (which stores $B$ items).
Bibliography

Paolo Ferragina and Roberto Grossi. 
Fast string searching in secondary storage: Theoretical developments and experimental results. 

Full-Text Indexes in External Memory, volume 2625 of Lecture Notes in Computer Science, chapter Chapter 7, pages 149–170. 