Database Tuning, ITU, Spring 2008

Rasmus Pagh

April 3, 2008

1 Inverted indexes versus full-text indexes

An alternative to a full-text index is to supplement an inverted index with an index on the vocabulary (set of words occurring in the documents). A search would first find the words that match one or more search terms, using the full-text index. Then the inverted index would be used to retrieve the matching documents for each search term.

a) How does the time to search for documents containing the two strings `inverted` and `index` in this way relate to the time spent in a full-text index (e.g., a suffix tree)?

b) Give an example of a search that could not be handled in this way (in fact, could not be handled by a simple inverted index) but is supported in a full-text index.

2 Space exploration

Disclaimer: Solving this problem will hardly get you a job at Rovsing.

Consider a space probe on its way to the equator of a remote rock planet. Upon impact it will need to determine its exact position along the equator. A previous space probe has gathered altitude data for all of the planet. The altitude data for the equator is a sequence of $40,000,000$ integers in the range 0 to $10,000$. The idea is to let the space probe gather altitude information in its near surroundings and search for this sequence in the data. As a small example, if the altitude data is

$$731, 731, 749, 748, 746, 748, 747, 749, 748, 750, 733$$

the sequence $(749, 748)$ fits two locations (and thus cannot be used to locate the probe). On the other hand, the sequence $(749, 748, 746)$ has a unique position. The space probe is equipped with hard drives that can hold all of the data, but its main memory (dated 1975) can hold only a few disk blocks. You are given the task of choosing an index for the data, such that for any sequence $(a_1, a_2, \ldots, a_i)$ of altitudes, it can efficiently be determined

whether this sequence matches a unique position in the data (and if so, what this position is).

a) Suppose it is known that the sequence looked up does not "wrap around" the end of the data (as e.g. $(750, 733, 731)$ in the above data). Suggest an efficient index that works in this setting.

b) Extend the solution such that "wrap around" sequences are also efficiently handled.