p6q1
MR

2003

4 Data structures and algorithms

(1) In the Burrows-Wheeler Block Compression algorithm it is necessary to sort all the suffixes of a vector of possibly tens of millions of bytes.

(a) Explain why Shell sort using a simple character by character string comparison function is unlikely to be satisfactory. [2 marks]

(b) Describe in detail the data structures and algorithms you would use to sort the suffixes and explain why your method is an improvement. You may assume that you have plently of RAM available. [8 marks]

(2) In the Burrows-Wheeler Block Compression algorithm a sequence of small positive integers are transmitted using Huffman encoding.

(a) Describe how the Huffman code is constructed. [4 marks]

(b) Outline how it can be represented compactly in the compressed file. [6 marks]

ANSWER NOTES:

This algorithm is mentioned in the syllabus and covered in detail in the lecture notes.

(1a)
Shell sort is typically much less efficient than quicksort. More importantly string comparison is not a unit cost operations since it depends on the length of string that must be compared before the answer in known. This can be very bad if the file to be compressed contains the same character repeated many millions of times.

(1b) Bookwork
Use a vector of pointers into the block.
Use a second vector of 32-bit words with each word containing four consective chars from the block. This allows word aligned comparison of prefixes 4 chars at a time.
Sort the pointers vector using radix sort, possibly on just the senior 16 bits in the data vector. It may be better to sort on all 32 bits.

If there are any repetitions of the same 4 bytes in the data vector these will need to be sort looking further down the prefix.

Choose the repeated sequence of shortest length and sort properly using quicksort. These values will now be in their true resting position. Put the position number in the least sig 24 bits of the corresponding data words. This will increase the efficiency of late prefix comparisons.

Where the same character is repeated there is a bookwork trick to sort these elements in linear time.

(2a) Bookwork
Sort the symbols and frequencies into increasing order of frequency. Form the huffman tree as described in the notes using basically 2 bits per tree node rather than pointers. Climb over the tree to computer the huffman code length. The encoding can be described by just giving the huffman code length for each symbol.

(2b) Bookwork
Send this table as a sequence of 4-bit nibbles which are essentially instructions for a simple abstract machine that generates the table.
The 16 instructions have name:
-4 -3 -2 -1 M +1 +2 +3 R1 R2 R3 R4 Z1 Z2 Z3 Z4
describe as in the notes.
Typical huffman codes can be described in about 50 bytes this way.