Applied Algorithmics COMP526 – tutorial 7

L.A. Gąsieniec and D. Cartwright

1 Questions

1.1 Burrows-Wheeler Transform

Explain briefly the mechanism used in the Burrows-Wheeler Transform (BWT) and show the result of applying of the BWT on the string $S = baabab$. Comment also on the use of reverse-BWT.

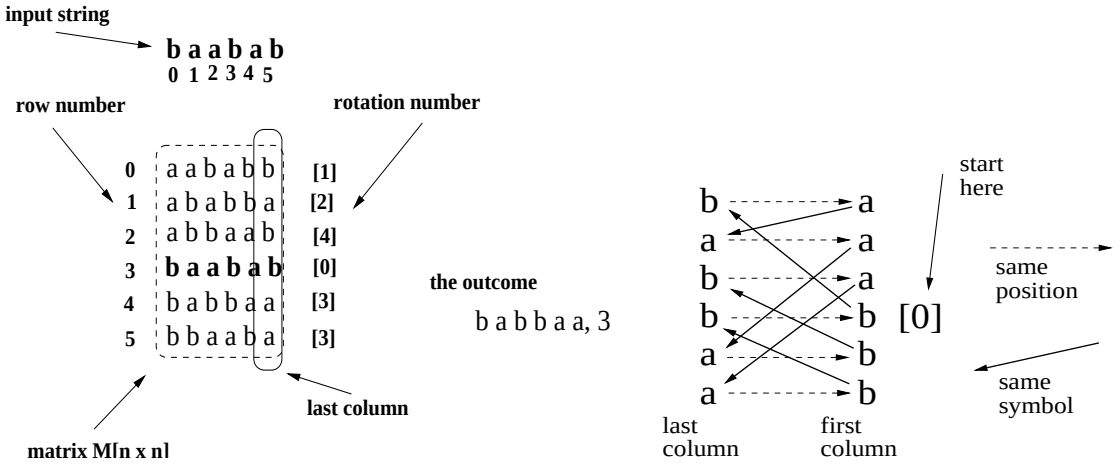
1.2 Huffman Coding

Assume you have 5 symbols A, B, C, D and E forming an alphabet $\mathcal{A}$, i.e., $\mathcal{A} = \{A, B, C, D, E\}$. The distribution of probabilities in the alphabet A is as follows: $P(A) = 0.1, P(B) = 0.4, P(C) = 0.2, P(D) = 0.1$, and $P(E) = 0.2$. Construct the Huffman code for symbols in A, i.e., the tree that minimizes the average number of bits (bit rate) to code a symbol from A.

2 Solutions

2.1 Burrows-Wheeler Transform

The Burrows-Wheeler transform takes all cyclic rotations on an input string of length n, it sorts them according to lexicographical order and place as rows in a $[n \times n]$ matrix M. As the outcome of the transformation we take the last column in the matrix M and the number of a row that contains the original string. See the picture below.



BW Transform **reverse-BW transform**

The reverse transform is based on creation of a directed cycle based on corresponding symbols and positions in the last and the first columns of M. The original word S is recovered by traversing the cycle and listing symbols in the column to the right starting at the row corresponding to the position of the original word in (lexicographically) ordered cyclic rotations of S.

2.2 Huffman Coding

Initially we sort all symbols in A according to their probability (weight) in the non-decreasing order. Each symbol is represented also by a single node, that will eventually become a leaf in the Huffman tree. In due course symbols from A form larger and larger groups where the probability probability of the group is the sum of probabilities of all symbols in it. Groups of symbols are represented as sub trees Huffman tree. During each step of the construction we select two lightest groups G_1 and G_2 (e.g., initially two singletons representing some symbols in A), we form the union of G_1 and G_2 which weight is the sum of probabilities of all symbols in G_1 and G_2. At the same time we form a new subtree with two children subtrees representing G_1 and G_2. The process is finished when the union of all symbols in A is formed. In our case we start with $P(A) = 0.1, P(B) = 0.4, P(C) = 0.2, P(D) = 0.1$, and $P(E) = 0.2$, see figure 1 During each step of the algorithm we have to maintain access to two groups with the smallest weight. This operation is

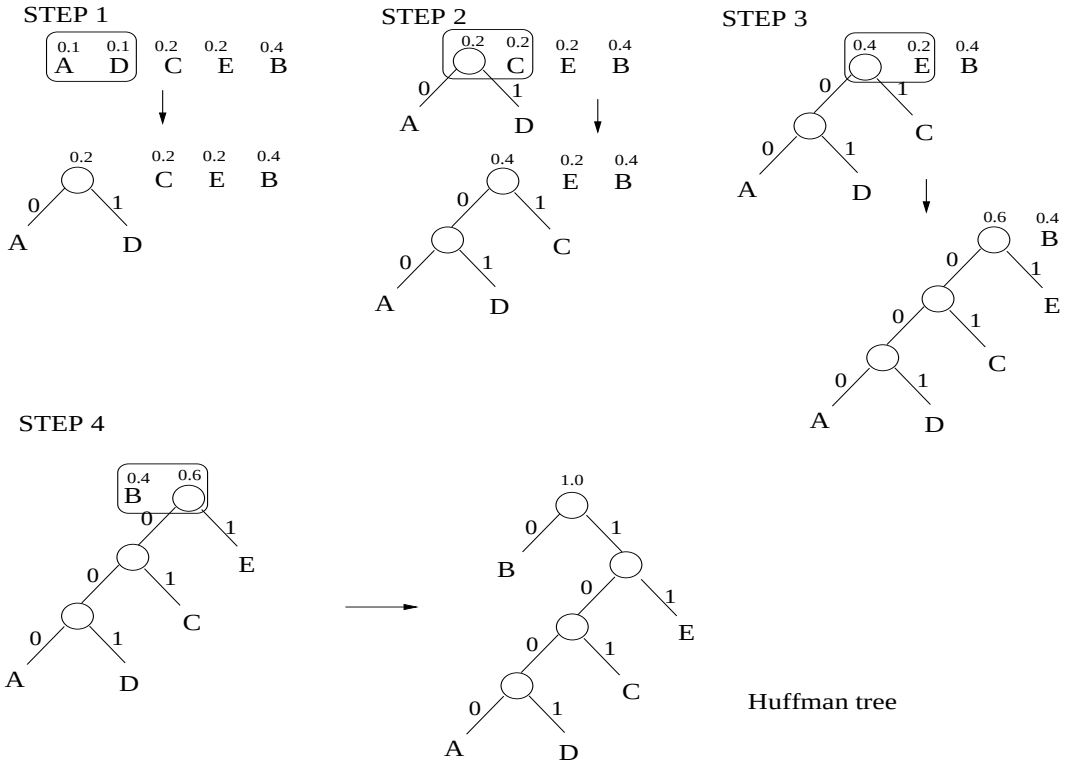


Figure 1: Huffman tree construction step by step

implementable in logarithmic time with the help of priority queue. Thus if the size of the alphabet is n, the total cost of Huffman tree construction is $O(n \log n.)$