Cs352 — Homework #7
Suffix trees

March 30, 2004

Due Time: 4/13 (9:00PM). Name your file tt suffixtree.c. Submission in pairs is allowed and encouraged.

The input to the exercise is the files `data.inp` containing a sequence of characters, each is one of the four characters a, b, c, d (for example *abaaacadaaabcdaaaaac*). Their number is unbounded. The second file, called `queries.inp`, where each line containing a string, over the alphabet {a,b,c,d}. Your program should first read `data.inp`, treat it as the compressed string R from the slides *suffix.ppt*. After the trie is constructed, your program should read each line from `queries.inp`, and print for each line
The string STING does not appear in data.inp
or
The string STING appears in index 17 in data.inp

For example, if `data.inp` contains *"abababacdca"* and

```
ab
ac
acdc
acc
```

The expected output should be
The string ab appears in index 4 in data.inp
The string ac appears in index 6 in data.inp
The string acdc appears in index 6 in data.inp
The string acc does not appear in data.inp

Finally, your program should print the compressed suffix tree T, in the following format. Your program prints the nodes of the tree in a preorder order. (That is, first the root of the tree is printed, and then your program prints all subtrees of the

root, from left to right, in a preorder order (recursively)). The exact format would be published later.

You can assume that the length of each query line is no more than 40 chars. The length of `data.inp` is not bounded.

Basic algorithms/instructions:

1. Read `data.inp` once, to find its size.
2. Read the file again, and store it in the array R. Create space for this array using malloc command.
3. Create an empty uncompressed trie, and insert into it all suffixes of R. Modify the fields `b_inx` during this process.
4. Call the function `compressed_trie` on the trie. This function compressed the trie. It uses the function `check_path` which accepts a node of the trie, and returns the length of the longest path starting at the node. Recall that the definition of a path is a sequence of nodes, each excluding the last has a **single** child, namely the next node of the path. The last node of the path has either zero, or at least 2 children. Use also the function `IsSingleChild` that accepts a node and returns true iff it has a single child. Remove from each path all but the first node of the path, and update the fields `c_inx, lng` accordingly
5. Reads the queries from `queries.inp`, and answer each appropriately.
6. Call the function `PrintTrie` that prints the tree.

Other comments

1. It is recommended that you check your program fist with the uncompressed trie. The code for the answering a query should work properly with compressed and uncompressed trie. (basically, in the uncompressed version, the 'lng' fields are all 0.
2. Your program should be consisting of the files `st_built.c`, creating the uncompressed trie, the file `st_compress.c` (containing the functions for compressing the trie, and the file file `st_query.c` (containing the part of answering queries and printing the trie). You should submit these files, and the Makefile.
3. After answering all queries, your program should free all allocated cells. Similarly the program should free the cells freed in the compression stage.