Chapter 32: String Matching with Rabin-Karp

Question 1

Without Mod

Say that we are processing text composed of the digits 0 through 9, so we will be using a basis of 10 for this question \((d = 10)\).

For this question you will not be computing the values mod \(q\).

Say that the text \(T\) is ‘1583832647’ and the pattern \(P\) is ‘83832’.

What is \(p\) and show the details of the computation according to Horner’s rule as shown in Section 32.2 of the textbook.

What is \(t_0\) and show the details of the computation just as you did for \(p\).

What is \(t_1, t_2, t_3\) and show the details of the computation according to Equation 32.1 of the textbook.

Question 2

Picking \(q\)

Let’s say that the computer word size is 16 bits. If we just want to store whole numbers \((0,1,2,3,...)\), what is the largest number that can be stored in 16 bits?

The textbook argues what we should pick \(q\) to be prime so that \(10q\) is at most the number that you computed above. Search the internet for a list of prime numbers, and give the largest value of \(q\).

Paraphrase the textbook’s explanation for why this is important.

If we do pick a larger \(q\), what will be the effect on the running time. Make sure you refer to big O notation. Make reasonable assumptions for how arithmetic operations would need to be done if they do not fit inside the computer’s word size. For example, if numbers need 64 bits to be stored, addition might take, 4 operations rather than one. Multiplication might take 16 operations, rather than one.

Note: this discussion is similar to the one that we had for the multiplication method for hash functions.
Question 3

Mod Operations

For this question, you will be doing modulo on negative numbers. \( x \mod c \) is defined as:

find \( r \) such that \( qc + r = x \) where \( r \) is a non-negative integer less than \( c \) and \( q \) is an integer.

What is \(-25 \mod 17\)?

Show the computation of the new value of \( p \) (using the mod operation) starting with your earlier answer.

Show the computation of the new value of \( t_0 \) (using the mod operation) starting with your earlier answer.

For computing subsequent \( t_s \), you need to precompute \( h = d^{m-1} \mod q \) (so that this is not done for each \( t_s \)).

Give the value of \( h \) and show your derivation of it.

Compute the values \( t_1, t_2, t_3 \) according to Equation 32.2, and show your derivation of them.

Chapter 32: String Matching with a Finite Automata

Question 4

Draw a Finite Automata

Draw a Finite Automata that match \( P = ababc \). You can assume that the alphabet is \( \Sigma = \{a, b, c\} \).

Feel free to draw the picture by hand on paper, take a picture of it, and include the picture in your writeup.

There is no need to mention any arc going to the start state, just as the textbook does.

You do not have to justify your answer.

Question 5

Prefixes of the Pattern

Fill in the following table for \( P = ababc \). \( P_i \) is the prefix of \( P \) that includes the first \( i \) characters of \( P \).

<table>
<thead>
<tr>
<th>Prefix</th>
<th>( P_0 )</th>
<th>( P_1 )</th>
<th>( P_2 )</th>
<th>( P_3 )</th>
<th>( P_4 )</th>
</tr>
</thead>
</table>

Question 6: Computing Transitions

For each prefix of the pattern, for possible next character, compute the length of the longest prefix of \( P \) that is a suffix of the resulting string. You will be essentially doing the algorithm: Compute-Transition-Function from the textbook.

Let’s do this step by step. The second column is just what you wrote above. For the 3rd column, take the prefix of \( P \) from column two, and add the character ‘a’ to it, and write down the string. Now this is the confusing part: what is the longest prefix of the pattern \( P \) that is a suffix of this string. Write down the length of that prefix in column 4. You have just wrote down what state you should transition to if the next character is an ‘a’.
Repeat what you did for column 3 and 4, for the next characters of ‘b’ and ‘c’.

The resulting table is in fact the transition table, where column 0 is the state you are coming from, columns 4, 6, and 8 say where to transition to if the next character is an ‘a’, ‘b’, or ‘c’, respectively. This transition table should correspond with the diagram that you drew for the earlier picture.

<table>
<thead>
<tr>
<th>i</th>
<th>$P_i$</th>
<th>Next character</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>a</td>
</tr>
<tr>
<td></td>
<td></td>
<td>longest prefix</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td>string</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Chapter 25: All-Pairs Shortest Paths: Matrix Multiplication**

The textbook defines the matrix $L^{(i)}$ through $l_{ij}^{(k)}$ as the shortest-path (by weight) from vertex $i$ to $j$ using a path of length at most $k$ ($k$ edges). If there is no such path, the weight is $\infty$. If $i = j$, the weight is 0.

**Question 7: Recursive Solution**

In the graph below, what is the shortest path from node 2 to node 1, using just one edge? What is the shortest path from node 2 to node 1 using 2 edges?

Compute $L^{(1)}$

Find the shortest path from node 2 to 1 using at most 2 edges ($i_{2,1}^{(2)}$) using $L^{(1)}$ along with the edge weights. In other words, walk through this example with equation 25.2, filling in actual values.

![Graph](image)

**Question 8: Computing L**

Compute $L^{(2)}$, and $L^{(3)}$ for the above graph.
Chapter 25: All-Pairs Shortest Paths: Floyd-Warshall

The textbook defines the matrix $D^{(k)}$ through $d^{(k)}_{ij}$ as the shortest-path (by weight) from vertex $i$ to $j$ using a path where all intermediate vertices are in $\{1, 2, \ldots, k\}$.

Question 9: L versus D

Which $L$ and $D$ matrixes will be the same (no matter what the graph is). Assume the graph has $n$ vertices.

Question 10: Recursive Definition

In the graph above, what is the weight of the shortest path from node 3 to node 2 in which all intermediate vertices are just from the set {}? How about where all intermediate vertices are from {1}? From {1, 2, 3, 4}? Compute $D^{(0)}$.

Explain how you find the weight of the shortest path going from 3 to 2 in which all intermediate nodes are just from the set {1} using $D^{(0)}$, in other words find $d^{(1)}_{3,2}$. Walk through the calculations using equation 25.5, filling in actual values. Note: the path does not need to go through node 1, it could use no intermediate nodes.

Question 11: Computing D

Compute the rest of the $D$’s: $D^{(1)}$, $D^{(2)}$, $D^{(3)}$ and $D^{(4)}$ for the above graph.

Chapter 24: Single-Source Shortest Paths

Question 12: Bellman-Ford

Run the Bellman-Ford algorithm on the graph above using node 4 as the source. Cycle through the edges in the following order:

(1, 2) (1, 3) (2, 1) (2, 3) (3, 1) (3, 4) (4, 2)

In your answer, write down each time you cycle through the edges in lines 2-4 of the algorithm. And any time that relax is successful, give the edge and its weight, and say which vertex has been updated, and what its new $d$ and $\pi$ values are.

For the loop in lines 5-7, say whether they detect a negative loop.

Note values for $d$ should agree with the 4th row of $D^{(4)}$.

Question 13: DAG Shortest Path

Why can’t the algorithm in section 24.2 not be used on the graph above?

Question 14: Dijkstra’s Algorithm

Again, let the source be node 4 of the graph above. Show the execution of Dijkstra’s algorithm.

Before doing line 5, show what vertices are in the queue, and their priority. Say what vertex is extracted in line 5. Anytime that relax is successful, say what vertex’s priority was changed, and what it was changed to.
Web Resource

This is a continuation of the problem introduced in Homework 7 and 9.

Question 15

Prefix 5
I don’t think it is possible to guarantee retrieval of all entries if you can just retrieve the next 5 words that match the prefix.

Devise a lexicon in which there is at least one word that cannot be extracted. The lexicon does not need to consist of real words, but each word needs to consist of one or more lowercase letters.

Question 16

Verifying your Answer
The code in hw11.py includes create_trie. You will alter this code so that it checks if it did in fact get the full lexicon.

In the lexicon class, create a method called Verify that will take a list as input and check if it matches its lexicon. If it does not, it should return false, as well as print an error message saying where the lists first differed, or if they do not have the same length. Otherwise it should return true.

Add to create_trie so that it gathers up the lexicon in the trie tree via the walk routine, and then calls Verify.

Hand in your code for Verify and create_trie.

Question 17

Retrieving with prefix5_equal
Create create_trie2 similar to create_trie that uses prefix5_equal. You can focus on one of the following:

1. find all entries in the lexicon. This version might make a lot of extra calls (mine made 17474 calls).
   Report how many calls you make. You can refine this by trying to reduce the number of calls, but still finding all entries. My version that does 17474 calls is pretty simplistic. You will hopefully do better.

2. find most of the answers very quickly. Focus on what percentage of the words you can retrieve with how many calls. Try to structure your code is that you just need one algorithm, and you tell it how many calls it can make.

Hand in your code and an explanation of your code, and how well it. Part of your explanation might use debugging output from your algorithm.