Homework 12: Algorithms and Data Structures

Digdarshan Kunwar

May 2019

!! NOTE !!
Implementation of different algorithms are in different folders.
References for doing the homework are given at the bottom of this sheet.

Problem 12.1

Backtracking: n Horses Problem

(a) In the folder Backtrack in the zip file.
The implementation is in Backtrack.cpp
$: make all
$: make run
$: make clean

Problem 11.2

Rabin-Karp String-Matching

(a) Here the Rabin-Karp algorithm matches the hash values of the pattern with the hash value of the current substring the string that we are searching on,so if the hash value of the substring and the pattern string that we are searching for matches then only the algorithm starts matching the individual characters.
For example:
Assumming:
Let the pattern be = eda
Let the string be = abeda

Let the prime = 3
And the values of a,b,c,d,e,f.... be 1,2,3,4,5,6..... respectively for now.

$x = oldhash - \mathbf{val}(old\_char)$
$x = \frac{x}{prime}$
$newhash = x + prime^{m-1} * val(new\_char)$ `where m = length of pattern`

So,

a	b	e	d	a

Here we have :
Initially $oldhash = 1 * 3^0 + 2 * 3^1 + 5 * 3^2 = 52 \implies$ **MISMATCH OF HASH**

Moving On to the next Sub String

a	b	e	d	a

Now we have :

$$
\begin{aligned}
oldhash &= 52 \\
x &= oldhash - val(old - char) \\
&= 52 - 1 = 51 \\
x &= x/prime = 51/3 = 17 \\
newhash &= x + prime^{m-1} * val(newchar) \\
&= 17 + 3^2 * val(d) = 53
\end{aligned}
$$

Checking with the original hash of the pattern $\Longrightarrow$ **MISMATCH OF HASH**

Moving On to the next Sub String

a	b	e	d	a

Now we have :

$$
\begin{aligned}
oldhash &= 53 \\
x &= oldhash - val(old - char) \\
&= 53 - 2 = 51 \\
x &= x/prime = 51/3 = 17 \\
newhash &= x + prime^{m-1} * val(newchar) \\
&= 17 + 3^2 * val(a) = 26
\end{aligned}
$$

Checking with the original hash of the pattern $\Longrightarrow$ **MATCH OF HASH** Here we check the characters individually if the hash values of both the sub string and the string matches. $\Longrightarrow$

a	b	e	d	a

$\Longrightarrow$ **PATTERN FOUND**

Thus we return the index of this position.
In this way we check the hash efficiently and check for the occurence of the pattern string within the orginal string.

(b) In the folder Rabin-Karp in the zip file. In the folder Backtrack in the zip file.
The implementation is in Rabin-Karp.cpp
$: make all
$: make run
$: make clean

References :

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C. (n.d.). Introduction to algorithms.
https://www.geeksforgeeks.org/
https://en.wikipedia.org/wiki/Rabin%E2%80%93Karp_algorithm
Discusion with friends for the approach to solve the algorithm.
https://www.geeksforgeeks.org/the-knights-tour-problem-backtracking-1/
https://algorithms.tutorialhorizon.com/backtracking-knights-tour-problem/