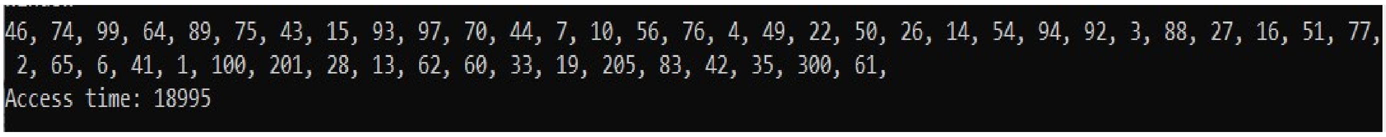
CSE225 Data Structures, 2020(FALL) 29.11.2020
Project#1 Report
Text Representation with Binary Trees
Zeynep Ferah Akkurt - 150119824

a) Root is the first word in the output. This is the output of binary search tree of words as in-order traversal.

******BST with the Word******
algorithm
Ankara
bag
board
book
bus
car
city
class
clock
club
compiler
computer
country
department
Dubai
economics
excel
faculty
game
grade
group
head
kitchen
lab →

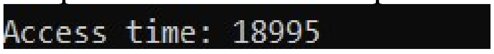
→ library
meeting
memory
mouse
name
New York
news
pencil
people
plane
population
professor
room
society
software
sports
student
teacher
team
television
text
traffic
university
visit
window

In-order frequencies of words:
46, 74, 99, 64, 89, 75, 43, 15, 93, 97, 70, 44, 7, 10, 56, 76, 4, 49, 22, 50, 26, 14, 54, 94, 92, 3, 88, 27, 16, 51, 77, 2, 65, 6, 41, 1, 100, 201, 28, 13, 62, 60, 33, 19, 205, 83, 42, 35, 300, 61



```
46, 74, 99, 64, 89, 75, 43, 15, 93, 97, 70, 44, 7, 10, 56, 76, 4, 49, 22, 50, 26, 14, 54, 94, 92, 3, 88, 27, 16, 51, 77,
 2, 65, 6, 41, 1, 100, 201, 28, 13, 62, 60, 33, 19, 205, 83, 42, 35, 300, 61,
Access time: 18995
```

b) The access time of this BTS is 18995.
Output of the code for b part

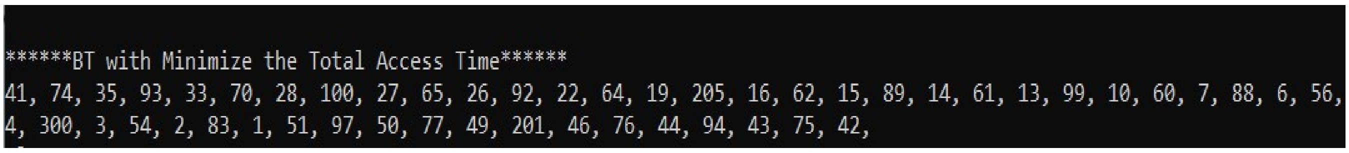


```
Access time: 18995
```

c) To minimize the total access time, the word that has the maximum frequency should be the root. And children of it must be the words that has the second and third maximum frequency. So, it must go in this order. From biggest frequency to the smallest one. Because there is no need to give an effort to reach the nodes at the bottom of the tree because of its high frequency. By putting them to top means, we access the other nodes by accessing them. That way total access time can be minimized.
So, in order to do that, first we should sort our words by their frequency from biggest to smallest. BST trees can be used for sorting by using RNL.

Here is the in-order traversal binary tree which has a better total access time:

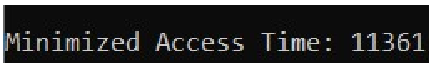
41, 74, 35, 93, 33, 70, 28, 100, 27, 65, 26, 92, 22, 64, 19, 205, 16, 62, 15, 89, 14, 61, 13, 99, 10, 60, 7, 88, 6, 56, 4, 300, 3, 54, 2, 83, 1, 51, 97, 50, 77, 49, 201, 46, 76, 44, 94, 43, 75, 42



```
******BT with Minimize the Total Access Time******
41, 74, 35, 93, 33, 70, 28, 100, 27, 65, 26, 92, 22, 64, 19, 205, 16, 62, 15, 89, 14, 61, 13, 99, 10, 60, 7, 88, 6, 56,
4, 300, 3, 54, 2, 83, 1, 51, 97, 50, 77, 49, 201, 46, 76, 44, 94, 43, 75, 42,
```

There are also words as lnr on console output.

d) Total access time of the binary tree in part (c) is 11361.



```
Minimized Access Time: 11361
```

e) The tree in part (a) has a bigger height. And height of the binary tree in part (c) is smaller. If a tree's height grows, access time also increases. So, from here we can see that;
Part (a) → 18995
Part (c) → 11361

Part (c) is much better and faster than the other one. Because tree's topology is better. Every node of every level except the last level has two children and placed by their access time. The most accessible key is on top. And all other keys placed by this rule. This why it is better.