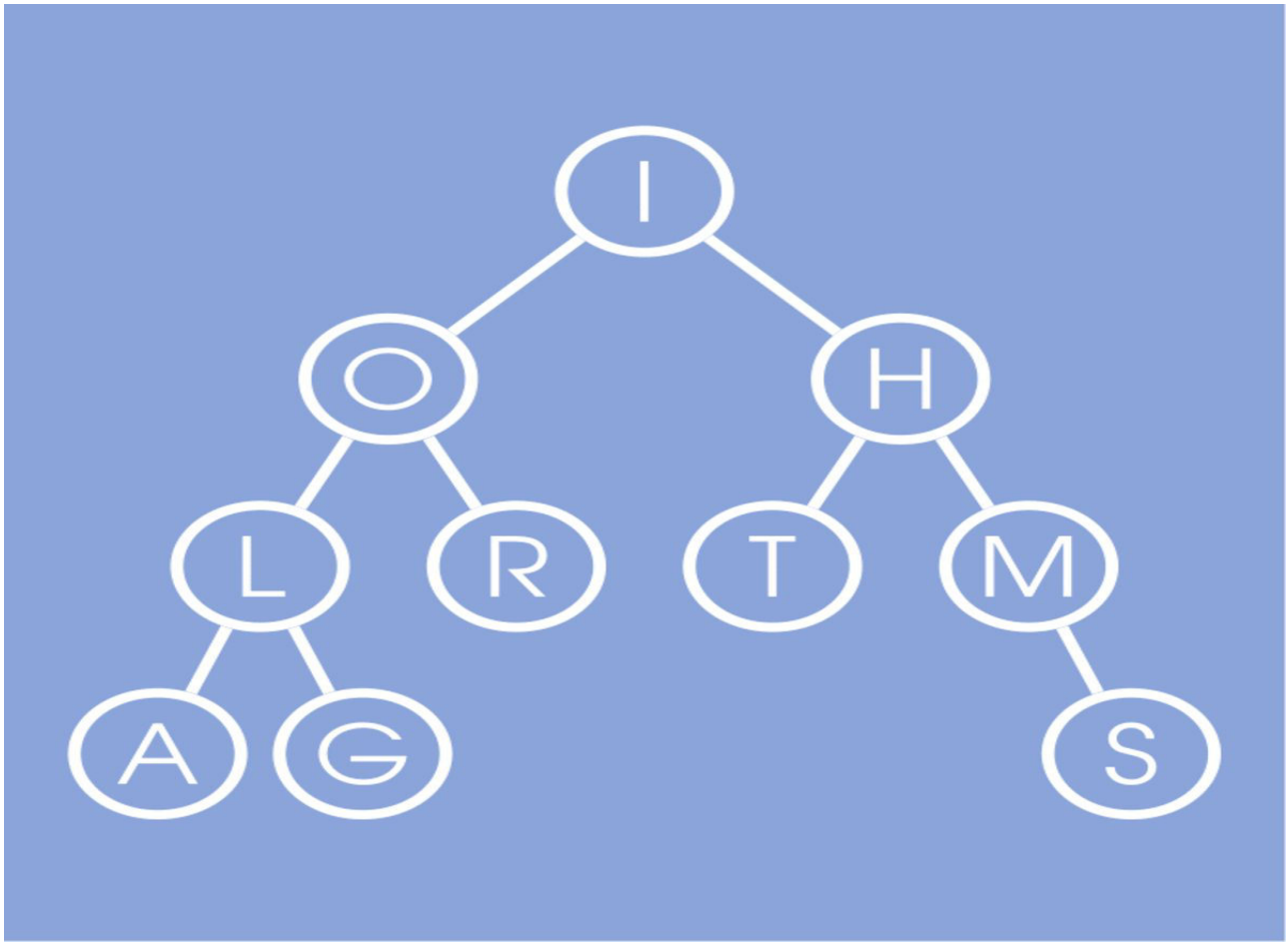
Experimentation

Assignment 1 : Taxi & For-Hire Vehicle Trip Dataset, Information Retrieval using Binary Search Trees



Syed Ahammad Newaz Saif

08/09/2019

Student Number : 684933

Bachelor of Science (Computing and Information Systems)

University of Melbourne

INTRODUCTION

In this assignment, the students were told to implement a binary search tree using linked list data structure for handling duplicates with the data handled here were of a particular data structure that had records (18 constant column fields) of a New York based Taxi and Limousine Commission trip record data .

DATASETS

There are primarily 4 basic comma separated values file :

a. 2018sample1_no_headers.csv
b. 2018sample2_no_headers.csv
c. 2018sample3_no_headers.csv
d. 2018_headers_only.csv
e. Rest are the set of many input test csv files

There were many key text files generated from this primary first 4 csv files.

There were many linux based and script techniques using 'shuf' , 'head', 'cat' and many other flag based commands that helped regenerated the test keys text files (both random and non random) and test input text files (both random and non random ones).

There was a basic shell script file provided by head tutor Grady Fitzpatrick that had 10 edge cases with first 5 test cases set for Stage 1 and the last 5 were for Stage 2.

STAGE 1 WITH TIME COMPLEXITY ANALYSIS

In this case, a binary search tree with a dictionary was implemented with basic insertion, search operations. We tried to read, write the files onto a trips structure that was inserted with "PUdatetime" as a key with the rest of the fields acting as values. The total number of nodes was equal to the total number of rows with the cost of insertion and cost of searching being O(log_2 N), where N is the number of nodes.

Basic command for Stage 1 : – ./ dict1 datafile outputfile < keyfile

STAGE 2 WITH TIME COMPLEXITY ANALYSIS

In this case, we do an in-order traversal on the binary search tree with a dictionary implemented that already had the basic insertion, search operations. We tried to read, write the files onto a trips structure that was inserted with “PUdatetime” as a key with the rest of the fields acting as values and the set of “PUlocationID”s were checked in order in the tree. The majority of the in-order traversal went to the right hand of the subtree and the left subtree was left empty as the one below suggests as my program makes nodes and inserts in the tree. Thereby, the cost of inorder traversal is O(N). where N is the number of nodes.

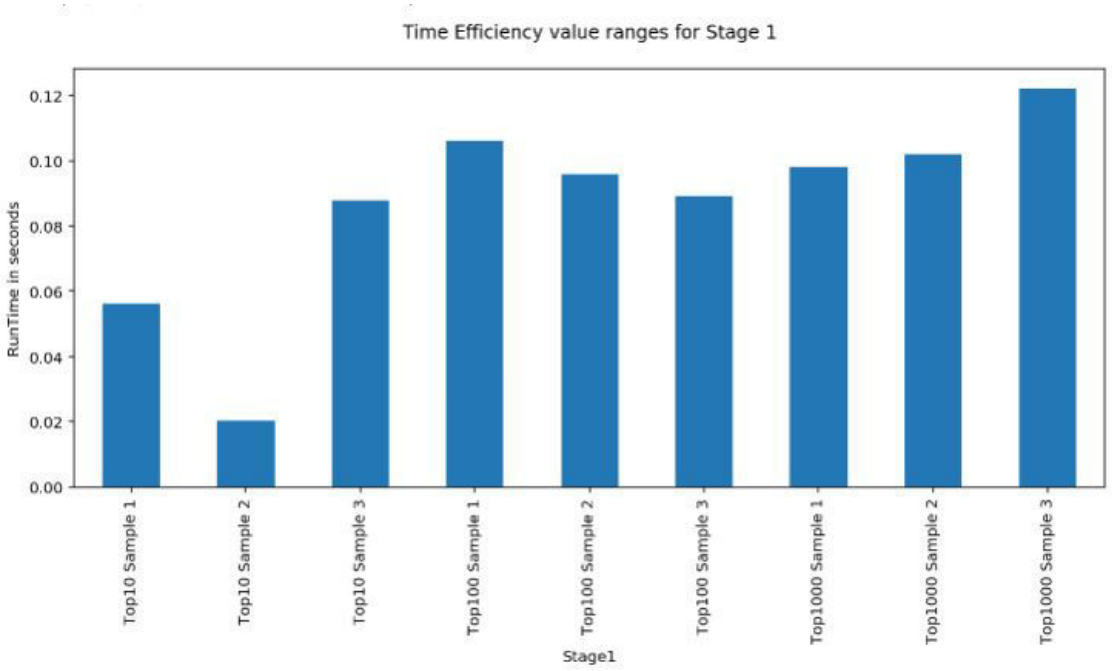
```
dict2: bstllist.c:135: printInorder: Assertion `bstree->left' failed.
./basic_ass1_tests.sh: line 55:   523 Aborted                 timeout $TIMEOUT_LENGTH ./$1 "$INPUT_DATA_FILE" "$RESULTS_FILE" > "$STDOUT_FILE" < "$ST
DIN_DATA_FILE"
```

Basic command for Stage 2 : – ./ dict2 datafile outputfile < idsfile

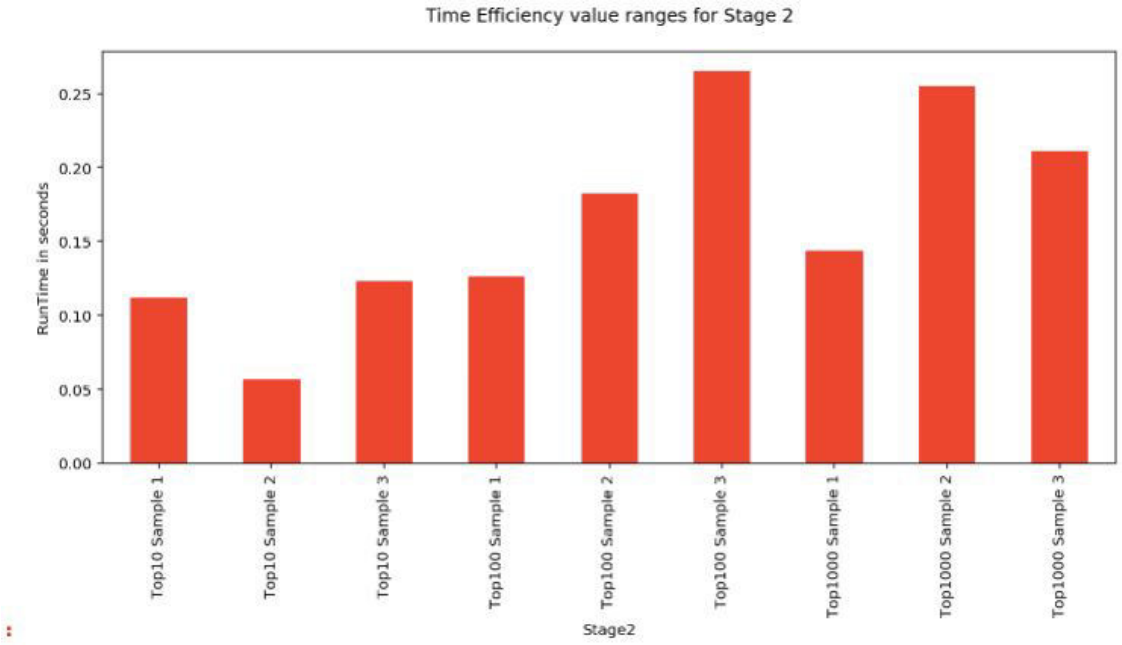
DISCUSSION AND EXPERIMENTATION

In the terminal, we tried to experiment by shuffling the 4 basic data sets (along with its accompanying keyfiles and idfiles) and found some conclusive runtimes for various datasets mainly top10, top100 and top1000 datasets for each of the 3 files (excluding headers file). A detailed bar chart of Stage 1 runtime execution, Stage 2 runtime execution and the contrast of the two runtimes are given below.

The Stage 1 runtime goes well with the theory and the runtime seems to be O(logN) as we have predicted.

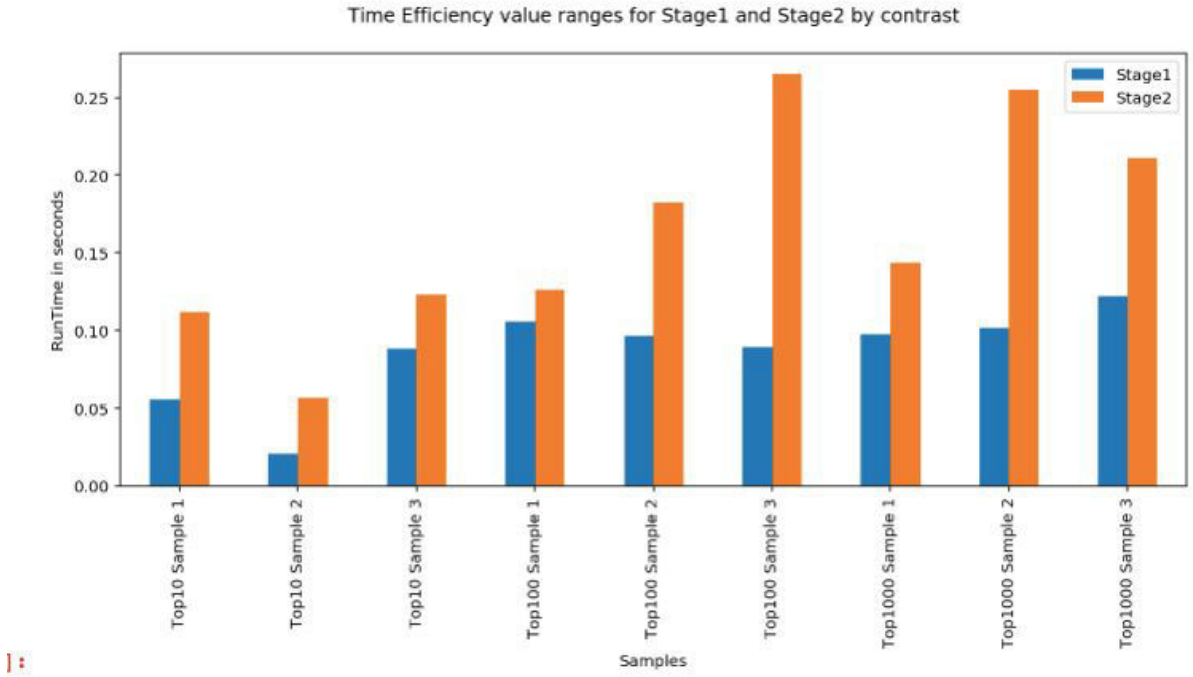


The Stage 2 In-order Traversal goes with the theory that lays out the reasoning for runtime bar plots that indicates that a O(N) of runtime complexity meaning inorder traversal takes place on the left subtree.



The final bar plot of the difference in bar plot that contrast in peaks strongly shows the difference in runtime peaks and match with the analysis already. The bar plot of the two

stages are contrasted with blue color suggesting Stage 1's O(logN) and orange color clearly signifying Stage 2's O(N).



]:

I have attached the Python Jupyter Notebook used for analysis along with this

2019-09-08-011731

September 8, 2019

[5]:
```
import pandas as pd;
import numpy as np
import matplotlib.pyplot as plt
```

[10]:
```
data = {'Stage1' : [ 'Top10 Sample 1', 'Top10 Sample 2', 'Top10 Sample 3',␣
↪'Top100 Sample 1', 'Top100 Sample 2', 'Top100 Sample 3', 'Top1000 Sample 1',␣
↪'Top1000 Sample 2', 'Top1000 Sample 3'], 'RunTime in seconds': [0.056, 0.
↪0203, 0.088,0.106,0.096,0.089,0.098,0.102,0.122] }
```

[11]:
```
df = pd.DataFrame(data)
```

[12]:
```
df
```

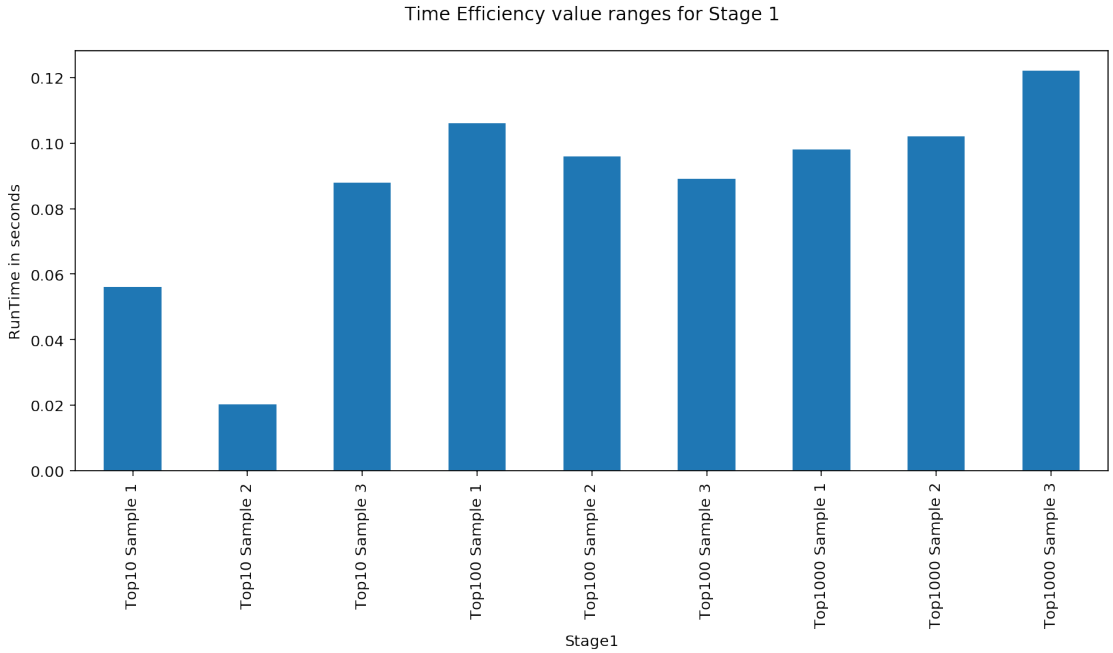
[12]:

	Stage1	RunTime in seconds
0	Top10 Sample 1	0.0560
1	Top10 Sample 2	0.0203
2	Top10 Sample 3	0.0880
3	Top100 Sample 1	0.1060
4	Top100 Sample 2	0.0960
5	Top100 Sample 3	0.0890
6	Top1000 Sample 1	0.0980
7	Top1000 Sample 2	0.1020
8	Top1000 Sample 3	0.1220

[13]:
```
ax = df.plot.bar('Stage1','RunTime in seconds',figsize=(12,5),legend = False,␣
↪title = 'Time Efficiency value ranges for Stage 1\n')
ax.set_ylabel('RunTime in seconds')
```

[13]:
```
Text(0,0.5,'RunTime in seconds')
```

[13]:



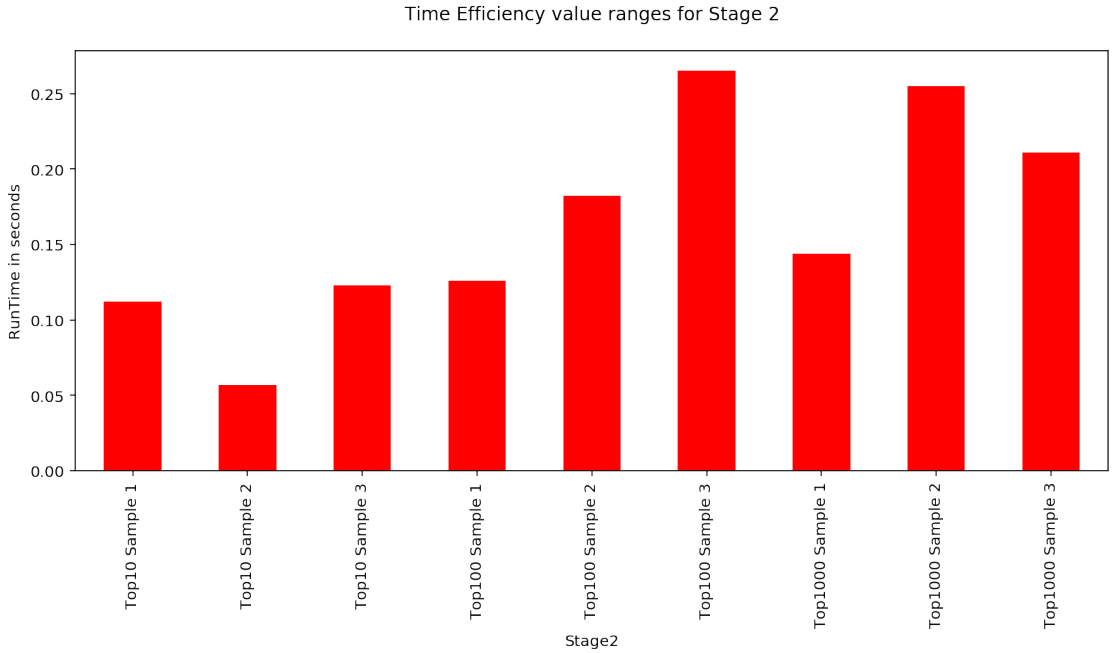
[14]:
```
data2 = {'Stage2' : [ 'Top10 Sample 1', 'Top10 Sample 2', 'Top10 Sample 3',␣
↪'Top100 Sample 1', 'Top100 Sample 2', 'Top100 Sample 3', 'Top1000 Sample 1',␣
↪'Top1000 Sample 2', 'Top1000 Sample 3'], 'RunTime in seconds': [ 0.112, 0.
↪057, 0.123,0.126,0.182,0.265,0.144,0.255,0.211] }
```

[15]:
```
df2 = pd.DataFrame(data2)
```

[16]:
```
ax2 = df2.plot.bar('Stage2','RunTime in seconds',figsize=(12,5),legend = False,␣
↪title = 'Time Efficiency value ranges for Stage 2\n',color = 'red')
ax2.set_ylabel('RunTime in seconds')
```

[16]:
```
Text(0,0.5,'RunTime in seconds')
```

[16]:



[17]:
```
final_df = pd.DataFrame({'Samples' : [ 'Top10 Sample 1', 'Top10 Sample 2',␣
↪'Top10 Sample 3', 'Top100 Sample 1', 'Top100 Sample 2', 'Top100 Sample 3',␣
↪'Top1000 Sample 1', 'Top1000 Sample 2', 'Top1000 Sample 3'], 'RunTime in␣
↪seconds Stage1': [0.056, 0.0203, 0.088,0.106,0.096,0.089,0.098,0.102,0.122],␣
↪'RunTime in seconds Stage2': [0.112, 0.057, 0.123,0.126,0.182,0.265,0.144,0.
↪255,0.211]})
```

[18]:
```
final_df
```

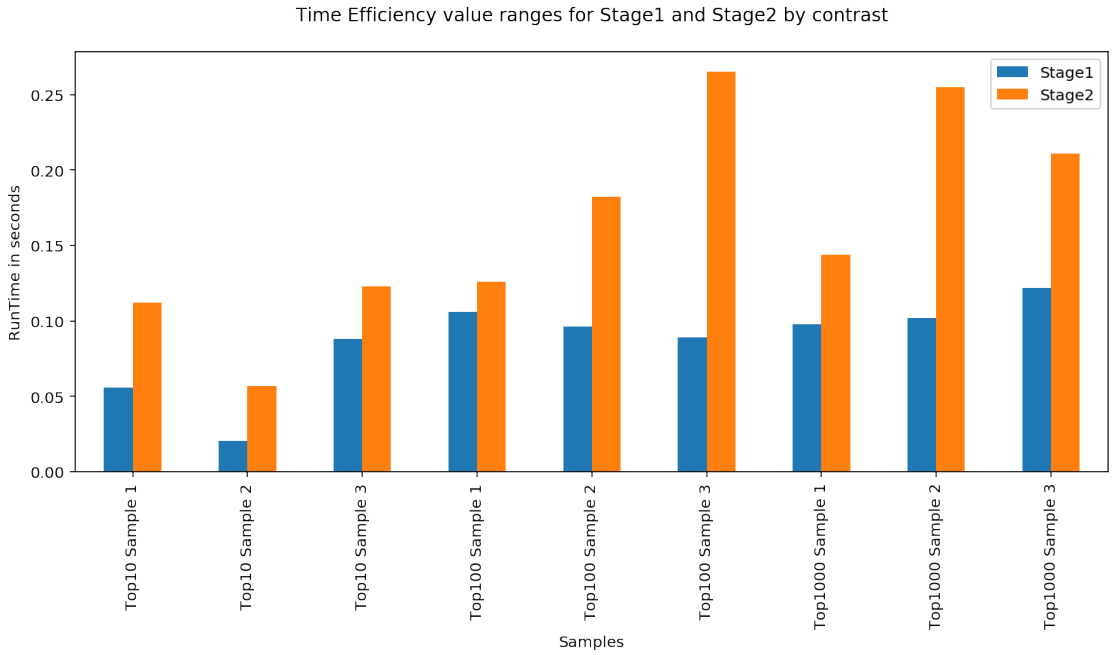
[18]:

	Samples	RunTime in seconds Stage1	RunTime in seconds Stage2
0	Top10 Sample 1	0.0560	0.112
1	Top10 Sample 2	0.0203	0.057
2	Top10 Sample 3	0.0880	0.123
3	Top100 Sample 1	0.1060	0.126
4	Top100 Sample 2	0.0960	0.182
5	Top100 Sample 3	0.0890	0.265
6	Top1000 Sample 1	0.0980	0.144
7	Top1000 Sample 2	0.1020	0.255
8	Top1000 Sample 3	0.1220	0.211

[19]:
```
ax2 = final_df.plot.bar(x ='Samples',y = ['RunTime in seconds Stage1','RunTime␣
↪in seconds Stage2'],figsize=(12,5),legend = False, title = 'Time Efficiency␣
↪value ranges for Stage1 and Stage2 by contrast\n')
ax2.legend(["Stage1", "Stage2"]);
ax2.set_ylabel('RunTime in seconds')
```

[19]: Text(0,0.5,'RunTime in seconds')

[19]:



[0]: