

“I confirm that I will keep the content of this assignment confidential. I confirm that I have not received any unauthorized assistance in preparing for or writing this assignment. I acknowledge that a mark of 0 may be assigned for copied work.” + Arun Reddy Nalla + 110088379

TASK 1

1. **Consider the graph stored in largeDG.txt (download it from Resources). Run DFS on that graph and show the vertices of the graph in pre-order and post-order. Compute the CPU time and report the worst-case complexity of DFS.**

i had created a new class TASK1.java for the above task.

I had used the ‘mediumDG.txt’ because I’m unable to run largeDG.txt.

I had used DepthFirstOrder method given in DepthFirstOrder.java from source code to run dfs on the medium.txt and also to obtain pre order and post order

And also calculated the time taken in milliseconds

output:

v	pre	post
0	0	38
1	39	41
2	33	27
3	42	48
4	47	45
5	22	8
6	6	22
7	37	37
8	29	32
9	19	2
10	49	49
11	20	1
12	35	30
13	26	13
14	41	39
15	3	33
16	18	3
17	48	44
18	27	15
19	32	25
20	44	42
21	21	0
22	5	23
23	25	10
24	46	46
25	34	26
26	36	29
27	45	47
28	7	21
29	28	20
30	14	11
31	11	16
32	12	14
33	31	28
34	1	35
35	16	5
36	43	43
37	15	9
38	24	6
39	9	18

40	2	34
41	38	36
42	10	17
43	23	7
44	8	19
45	40	40
46	17	4
47	13	12
48	30	31
49	4	24

```
Pre order - 0 34 40 15 49 22 6 28 44 39 42 31 32 47 30 37 35 46 16 9 11 21 5 43 38 23 13 18 29 8 48 33 19 2 25 12 26 7 41 1 45 14 3 36 20 27 24 4 17 10
The total CPU Time For PreOrder = 3ms

Post order - 21 11 9 16 46 35 38 43 5 37 23 30 47 13 32 18 31 42 39 44 29 28 6 22 49 19 25 2 33 26 12 48 8 15 40 34 41 7 0 14 45 1 20 36 17 4 24 27 3 10
The total CPU Time For PostOrder = 0ms
```

According to the output,
pre-order traversal takes longer than post-order traversal.

Depth first search's worst-case performance is as follows: $O(V+E)$
where V denotes the number of vertices and E denotes the number of edges

2. Consider the graph stored in largeEWG.txt (download it from Resources).
 - a. Write a program that finds the shortest path for all pairs of nodes (you choose the algorithm). Calculate the CPU time and report the complexity of the algorithm you chose.
 - b. Write a program that finds the MST (you choose the algorithm). Calculate the CPU time and compare it with the complexity of the algorithm you chose.

I had created Task2A.java for finding shortest path for all pair of nodes.

I had chosen Dijkstra Algorithm from DijkstraSP.java for this question

I had calculated shortest path and CPU time in milliseconds using DijkstraSP.java

Implementing Dijkstra Algorithm

```
From 0 to 0, Distance = 0.00
From 0 to 15786, Distance = 0.00
From 0 to 53370, Distance = 0.00
From 0 to 54517, Distance = 0.00
From 0 to 128999, Distance = 0.00
From 0 to 157440, Distance = 0.00
From 0 to 200079, Distance = 0.00
From 0 to 212733, Distance = 0.00
From 0 to 306618, Distance = 0.00
From 0 to 310931, Distance = 0.00
From 0 to 312057, Distance = 0.00
From 0 to 328956, Distance = 0.00
From 0 to 331670, Distance = 0.01
From 0 to 339430, Distance = 0.01
From 0 to 359324, Distance = 0.01
From 0 to 380102, Distance = 0.01
From 0 to 395146, Distance = 0.01
From 0 to 402714, Distance = 0.01
From 0 to 408379, Distance = 0.00
From 0 to 414512, Distance = 0.01
From 0 to 431744, Distance = 0.00
From 0 to 440226, Distance = 0.01
From 0 to 454730, Distance = 0.00
From 0 to 460714, Distance = 0.01
From 0 to 460790, Distance = 0.00
From 0 to 465923, Distance = 0.01
From 0 to 474885, Distance = 0.00
From 0 to 476791, Distance = 0.00
From 0 to 481182, Distance = 0.00
From 0 to 481534, Distance = 0.01
From 0 to 491631, Distance = 0.01
From 0 to 495005, Distance = 0.01
From 0 to 511813, Distance = 0.01
From 0 to 521551, Distance = 0.01
From 0 to 534225, Distance = 0.01
From 0 to 540833, Distance = 0.00
From 0 to 557988, Distance = 0.01
From 0 to 559815, Distance = 0.01
From 0 to 566128, Distance = 0.01
From 0 to 568284, Distance = 0.01
```

From 0 to 568807, Distance = 0.01
From 0 to 569465, Distance = 0.01
From 0 to 577617, Distance = 0.01
From 0 to 578709, Distance = 0.01
From 0 to 588007, Distance = 0.00
From 0 to 594358, Distance = 0.00
From 0 to 596329, Distance = 0.01
From 0 to 618450, Distance = 0.00
From 0 to 626648, Distance = 0.01
From 0 to 636889, Distance = 0.01
From 0 to 639148, Distance = 0.01
From 0 to 646447, Distance = 0.01
From 0 to 656781, Distance = 0.00
From 0 to 662396, Distance = 0.01
From 0 to 687911, Distance = 0.00
From 0 to 689769, Distance = 0.01
From 0 to 695790, Distance = 0.00
From 0 to 696678, Distance = 0.00
From 0 to 701374, Distance = 0.00
From 0 to 708003, Distance = 0.00
From 0 to 712706, Distance = 0.01
From 0 to 713461, Distance = 0.00
From 0 to 713712, Distance = 0.01
From 0 to 713944, Distance = 0.01
From 0 to 721804, Distance = 0.00
From 0 to 727281, Distance = 0.01
From 0 to 729709, Distance = 0.00
From 0 to 730118, Distance = 0.01
From 0 to 733403, Distance = 0.01
From 0 to 740055, Distance = 0.01
From 0 to 741044, Distance = 0.01
From 0 to 749867, Distance = 0.00
From 0 to 752483, Distance = 0.00
From 0 to 755966, Distance = 0.01
From 0 to 757322, Distance = 0.01
From 0 to 760029, Distance = 0.00
From 0 to 760815, Distance = 0.00
From 0 to 761112, Distance = 0.01
From 0 to 775588, Distance = 0.00
From 0 to 780059, Distance = 0.01
From 0 to 784709, Distance = 0.00
From 0 to 785187, Distance = 0.01
From 0 to 786475, Distance = 0.01
From 0 to 787584, Distance = 0.01
From 0 to 788891, Distance = 0.01
From 0 to 789796, Distance = 0.01
From 0 to 793394, Distance = 0.01
From 0 to 795878, Distance = 0.01
From 0 to 808159, Distance = 0.00
From 0 to 814262, Distance = 0.01
From 0 to 820985, Distance = 0.01
From 0 to 822796, Distance = 0.00
From 0 to 826279, Distance = 0.01
From 0 to 831605, Distance = 0.01
From 0 to 832346, Distance = 0.01
From 0 to 833756, Distance = 0.01
From 0 to 836993, Distance = 0.00
From 0 to 840164, Distance = 0.00
From 0 to 844148, Distance = 0.01
From 0 to 845001, Distance = 0.01
From 0 to 853383, Distance = 0.01
From 0 to 862314, Distance = 0.01
From 0 to 864194, Distance = 0.01
From 0 to 864526, Distance = 0.01
From 0 to 874680, Distance = 0.01
From 0 to 876346, Distance = 0.01
From 0 to 879889, Distance = 0.01
From 0 to 885692, Distance = 0.01
From 0 to 890258, Distance = 0.01
From 0 to 900196, Distance = 0.01
From 0 to 900655, Distance = 0.00
From 0 to 902442, Distance = 0.01
From 0 to 904610, Distance = 0.01
From 0 to 916585, Distance = 0.01
From 0 to 917224, Distance = 0.00
From 0 to 931111, Distance = 0.02
From 0 to 932942, Distance = 0.00
From 0 to 935462, Distance = 0.00
From 0 to 935862, Distance = 0.00
From 0 to 939609, Distance = 0.01
From 0 to 944881, Distance = 0.01
From 0 to 945400, Distance = 0.01

```
From 0 to 947445, Distance = 0.00
From 0 to 948039, Distance = 0.02
From 0 to 952370, Distance = 0.01
From 0 to 953125, Distance = 0.00
From 0 to 954042, Distance = 0.01
From 0 to 955608, Distance = 0.01
From 0 to 966067, Distance = 0.02
From 0 to 968628, Distance = 0.01
From 0 to 969240, Distance = 0.00
From 0 to 970361, Distance = 0.01
From 0 to 971961, Distance = 0.01
From 0 to 972236, Distance = 0.01
From 0 to 981357, Distance = 0.01
From 0 to 982015, Distance = 0.01
From 0 to 990620, Distance = 0.00
From 0 to 996673, Distance = 0.00
From 0 to 998885, Distance = 0.01

Total CPU Time taken by Dijkstra Algorithm = 156 milliseconds
```

The CPU time taken by Dijkstra Algorithm is 156 milliseconds

From chapter 6 in Resources

The complexity of the Dijkstra Algorithm, according to theory, is $O(|E| + |V| \log |V|)$.

TASK 2B

For this Task, I have created Task2B.java class

I had chosen Kruskal Algorithm for this question

I had used KruskalMST.java to find MST and also calculated CPU time in milliseconds

I had taken largeEWG.txt file for this task

```
223350-600765 0.00142
193206-965603 0.00142
112730-737747 0.00142
194742-256890 0.00142
21617-464187 0.00142
508499-832729 0.00142
165000-461895 0.00142
190020-926140 0.00142
318692-768040 0.00142
692857-859392 0.00142
188673-932921 0.00142
551973-969416 0.00142
244718-960406 0.00142
751749-879537 0.00142
227674-891689 0.00142
595666-980401 0.00142
23965-976072 0.00142
93736-523201 0.00142
```

```
76860-193921 0.00197
16696-819106 0.00198
14738-714714 0.00199
200257-799826 0.00201
32599-44408 0.00202
148837-372243 0.00202
150172-924026 0.00206
294823-853468 0.00209
Finding MST of large weighted Graph
```

```
CPU Time taken to compute the graph with Kruskal Algorithm is 3493ms
```

```
CPU Time taken to compute the graph with Kruskal Algorithm is 3493 milliseconds
```

From chapter 6

Worst-case running time: ▪ Using a binary heap: $O(|E| \log |V|)$

TASK 3

Consider the movie database stored in movie.txt, and SymbolGraph.java. Write a program that uses DFS to find all connected components. Use CC.java as a template. Show the CPU time and report the worst-case complexity of DFS.

I had created Task3.java class for this task.

I had SymbolGraph.java for this program

Below is the output after executing the program using movie.txt

```
Done reading movies.txt
33 components
Total CPU Time = 837ms
```

If the graphs are explicit and traversed once,

the worst-case complexity is $O(|V| + |E|)$,

where V represents the number of vertices and E represents the number of edges.

Task 4

Write a program that finds the movies starred by a particular actor. Show the movies starred by Leonardo DiCaprio. Show the movies starred by Julia Roberts, by Hugh Grant, and by both of them.

I had created Task4.java class for this task.

I had written the program which movies starred by each actor from movies.txt file
And also movies that which are starred by both Julia Roberts and Hugh Grant.

```
|done reading movies.txt
```

```
Searching movies for the actor : DiCaprio, Leonardo
```

```
What's Eating Gilbert Grape (1993)  
Total Eclipse (1995)  
Titanic (1997)  
This Boy's Life (1993)  
Romeo + Juliet (1996)  
Quick and the Dead, The (1995)  
Poison Ivy (1992)  
Marvin's Room (1996)  
Man in the Iron Mask, The (1998 I)  
Gangs of New York (2002)  
Departed, The (2006)  
Celebrity (1998)  
Catch Me If You Can (2002)  
Beach, The (2000 I)  
Basketball Diaries, The (1995)  
Aviator, The (2004)
```

Searching movies for the actor : Roberts, Julia (I)

Stepmom (1998)
Steel Magnolias (1989)
Something to Talk About (1995)
Sleeping with the Enemy (1991)
Runaway Bride (1999)
Pratt & Porter (1994)
Pretty Woman (1990)
Player, The (1992)
Pelican Brief, The (1993)
Ocean's Twelve (2004)
Ocean's Eleven (2001)
Notting Hill (1999)
Mystic Pizza (1988)
My Best Friend's Wedding (1997)
Mona Lisa Smile (2003)
Michael Collins (1996)
Mexican, The (2001)
Mary Reilly (1996)
I Love Trouble (1994)
Hook (1991)
Full Frontal (2002)
Flatliners (1990)
Everyone Says I Love You (1996)
Erin Brockovich (2000)
Dying Young (1991)
Conspiracy Theory (1997)
Confessions of a Dangerous Mind (2002)
Closer (2004 I)
America's Sweethearts (2001)

Searching movies for the actor : Grant, Hugh (I)

Two Weeks Notice (2002)
Small Time Crooks (2000)
Sirens (1994)
Sense and Sensibility (1995)
Restoration (1995)
Remains of the Day, The (1993)
Notting Hill (1999)
Nine Months (1995)
Mickey Blue Eyes (1999)
Maurice (1987)
Love Actually (2003)
Lair of the White Worm, The (1988)
Four Weddings and a Funeral (1994)
Extreme Measures (1996)
Englishman Who Went Up a Hill But Came Down a Mountain, The (1995)
Bridget Jones: The Edge of Reason (2004)
Bridget Jones's Diary (2001)
Bitter Moon (1992)
American Dreamz (2006)
About a Boy (2002)

Searching common movies for both actors

Roberts, Julia (I) And Grant, Hugh (I) is Notting Hill (1999)

Roberts, Julia (I) And Grant, Hugh (I) is Notting Hill (1999)

TASK 5

Consider the one million Chip-seq reads given in the files called “Chip-seq-reads-1M.dat”. Write a program that partitions the list of reads into 4 sublists. Save each sublist in a separate file (called A.dat, B.dat, C.dat, and D.dat). Sort each sublist and store it in a file (AS.dat, BS.dat, CS.dat, DS.dat). Take the 4 sorted sublists from the files and merge them in to a sorted list. Store the sorted list in a file (called “Chip-seq-reads-1M-sorted.dat”).

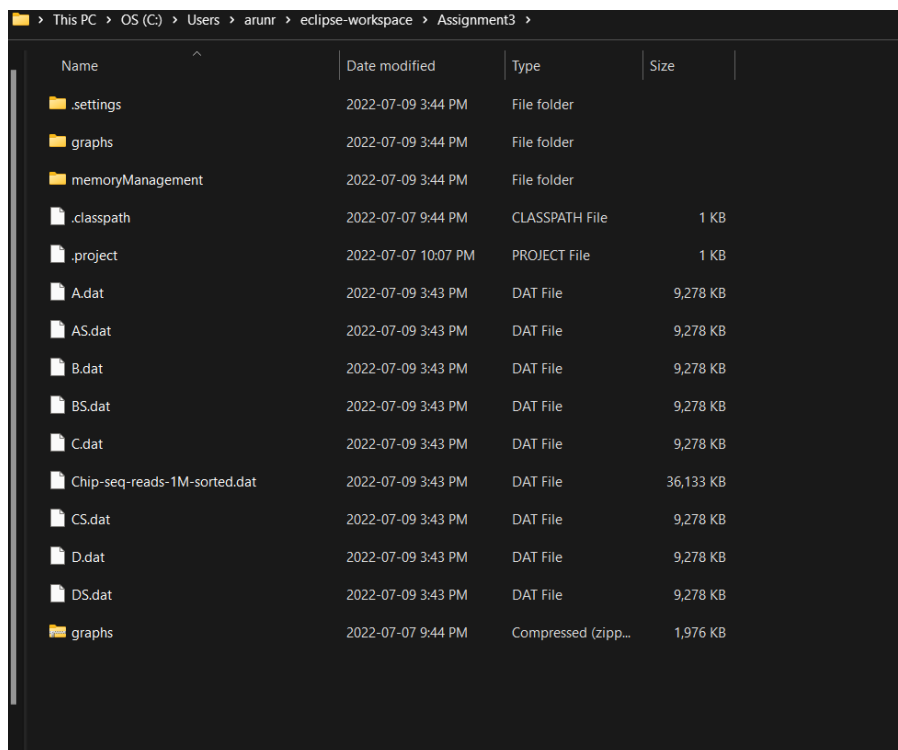
I had used Task5.java for this task

The program reads the 1 million-record from file Chip-seq-reads-1M.dat and writes the data into four sub-files A.dat, B.dat, C.dat, and D.dat, then sorts the files using Arrays sort and writes the sorted data in AS.dat, BS.dat, CS.dat, and DS.dat.

After sorting through all of the data from four files these are merged and saved into single file Chip-seq-reads-1M-sorted.dat

```
<terminated> Task5 (2) [Java Application] C:\Users\arunr\p2\pool\plugins\org.eclipse.
```

```
Total Time:3107
```



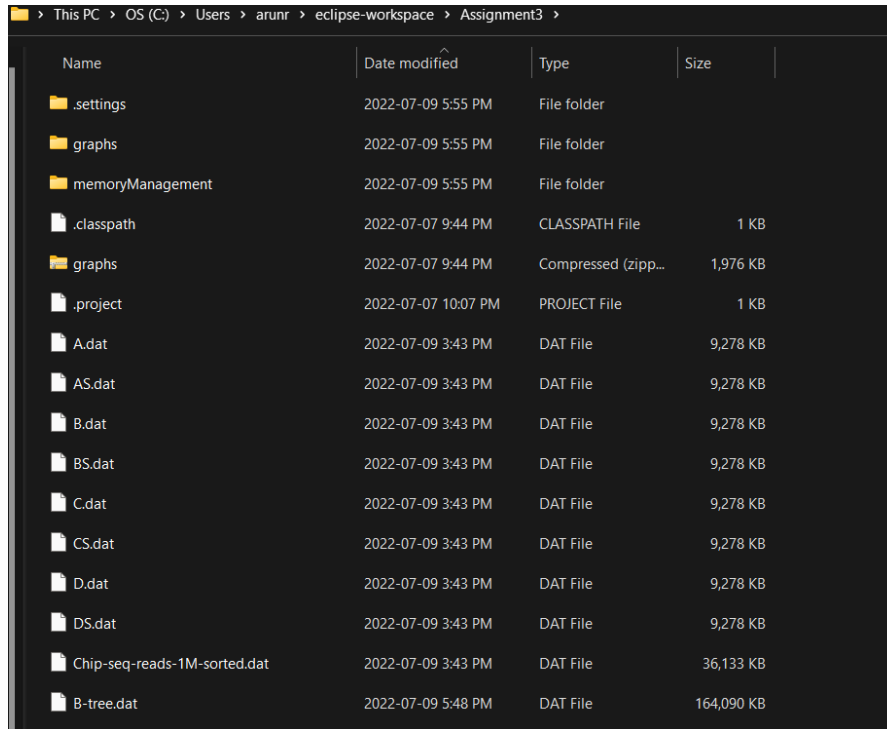
Name	Date modified	Type	Size
.settings	2022-07-09 3:44 PM	File folder	
graphs	2022-07-09 3:44 PM	File folder	
memoryManagement	2022-07-09 3:44 PM	File folder	
.classpath	2022-07-07 9:44 PM	CLASSPATH File	1 KB
.project	2022-07-07 10:07 PM	PROJECT File	1 KB
A.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
AS.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
B.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
BS.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
C.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
Chip-seq-reads-1M-sorted.dat	2022-07-09 3:43 PM	DAT File	36,133 KB
CS.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
D.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
DS.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
OS.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
graphs	2022-07-07 9:44 PM	Compressed (zipp...	1,976 KB

TASK 6

Create a B-tree and insert all the reads from the original list (Chip-seq-reads-1M.dat) as they appear in the file. List the B-tree in in-order traversal and save the output all keys in a file (called B-tree.dat).

I had created Task6.java for this program

I had read all the 1M records from Chip-seq-reads-1M.dat and listed B-tree in in-order traversal and saved the output all keys in a B-tree.dat file.



Name	Date modified	Type	Size
.settings	2022-07-09 5:55 PM	File folder	
graphs	2022-07-09 5:55 PM	File folder	
memoryManagement	2022-07-09 5:55 PM	File folder	
.classpath	2022-07-07 9:44 PM	CLASSPATH File	1 KB
graphs	2022-07-07 9:44 PM	Compressed (zipp...	1,976 KB
.project	2022-07-07 10:07 PM	PROJECT File	1 KB
A.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
AS.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
B.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
BS.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
C.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
CS.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
D.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
DS.dat	2022-07-09 3:43 PM	DAT File	9,278 KB
Chip-seq-reads-1M-sorted.dat	2022-07-09 3:43 PM	DAT File	36,133 KB
B-tree.dat	2022-07-09 5:48 PM	DAT File	164,090 KB

```
<terminated> Task6 [Java Application] C:\Users\arunr\.p2\pool\plugins\org.eclipse.justj.openjdk  
Total Time for B-Tree is:7565
```

TASK 7

Record total CPU times for #5 and #6. Comment on the obtained CPU times and compare them with the corresponding complexities as discussed in class.

For Task 5 CPU time : 3107 milliseconds

time complexity is $O((n/B)\log(n/B)\log d)$

Where B is the size of the block

For Task 6 CPU time : 7565 milliseconds

From chapter 7

B-trees Complexity:

- Searches, insertions and removals

$O(\log_B n)$ I/O complexity

Uses $O(n/B)$ blocks