University of Windsor

COMP-8547-1-R-2021S

Advanced Computing Concepts
Summer 2021
Assignment 3

Submitted By
Anubha Sharma
Master of Applied Computing
Student Id: 110037181

Submitted To
Dr. Abedalrhman Alkhateeb

University of Windsor

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."

Anubha
110037181

Question 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.

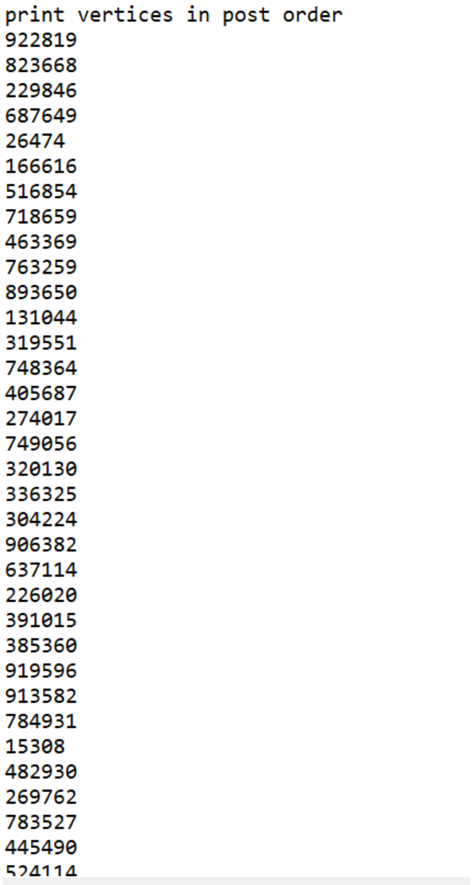
Solution 1: Please refer package solution, class QuestionOne.java and method main for the coded solution

Steps to resolve are as follows:

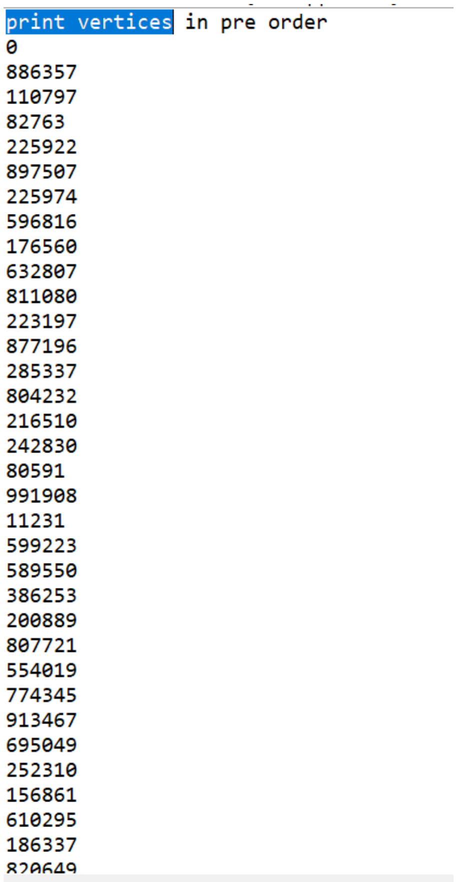
1. Read the graph using In.java and diagraph.java
2. Use DepthFirstOrder.post() to iterate through the graph in post order.
3. Use DepthFirstOrder.pre() to iterate through the graph in pre order.
4. Worst case complexity of DFS is directy proportional to the number of vertices and edges because all the vertices will be visited once in worst case. So the complexity is O(V+E).

OUTPUT

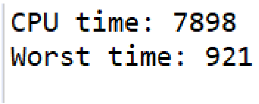
Since the output is large, hence I have attached a snippet of the output



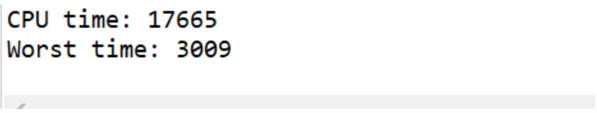
```
print vertices in post order
922819
823668
229846
687649
26474
166616
516854
718659
463369
763259
893650
131044
319551
748364
405687
274017
749056
320130
336325
304224
906382
637114
226020
391015
385360
919596
913582
784931
15308
482930
269762
783527
445490
524114
```



```
print vertices in pre order
0
886357
110797
82763
225922
897507
225974
596816
176560
632807
811080
223197
877196
285337
804232
216510
242830
80591
991908
11231
599223
589550
386253
200889
807721
554019
774345
913467
695049
252310
156861
610295
186337
820649
```



```
CPU time: 7898
Worst time: 921
```



```
CPU time: 17665
Worst time: 3009
```

Question 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.

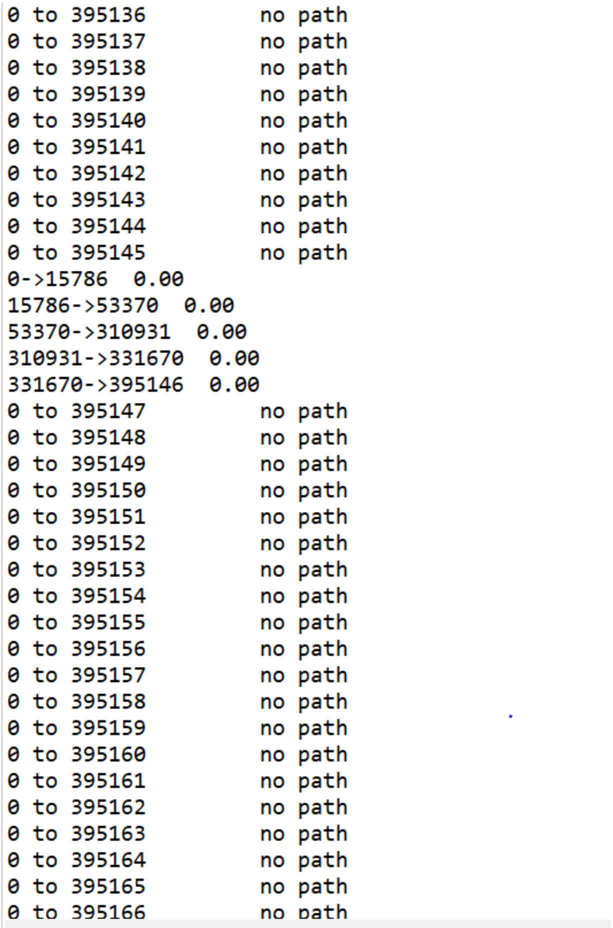
Solution 2: Please refer package solution, class QuestionTwo.java and method main for the coded solution.

Steps to resolve are as follows:
a.

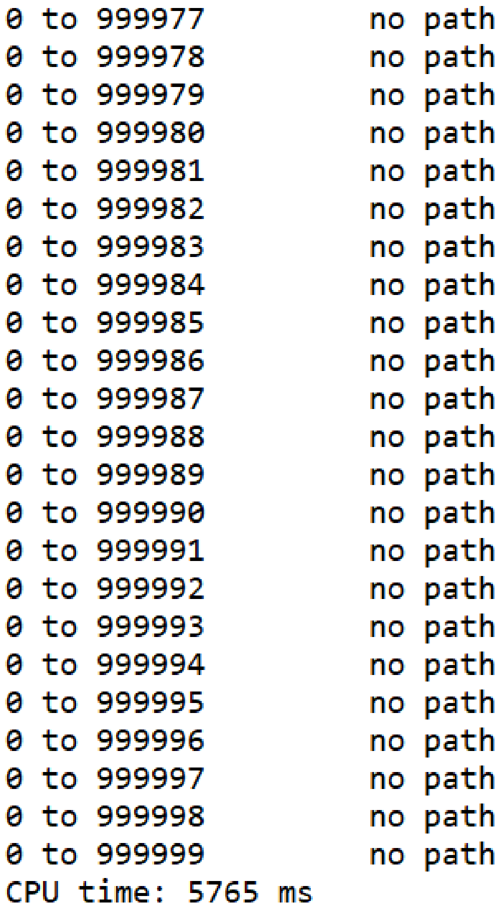
1. Using EdgeWeightedDigraph and DijkstraSP calculate the shortest distance for graph stored in largeEWG.txt
2. Calculate the CPU Time.
3. The complexity of DijkstraSP is o(E logV).

b.

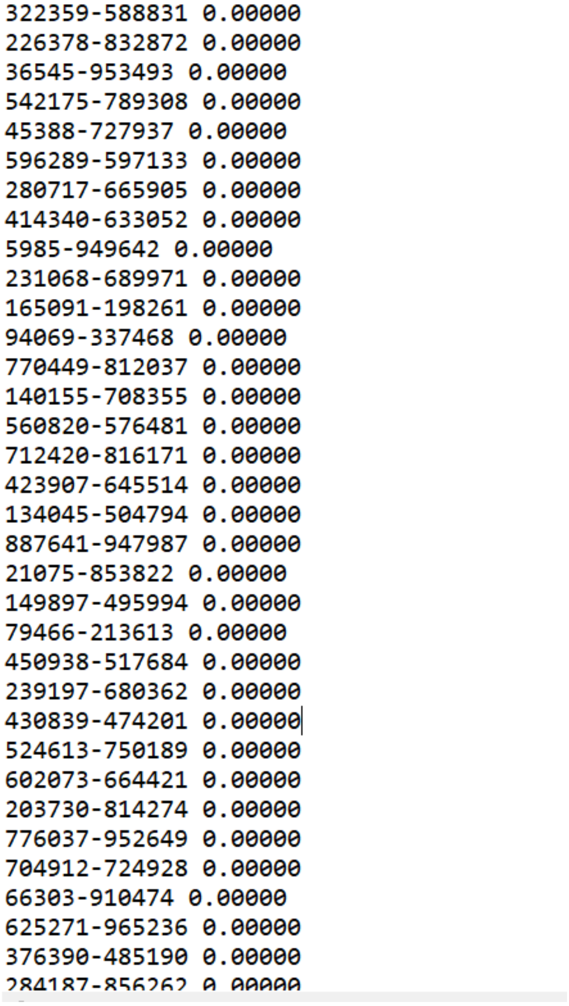
4. Using the EdgeWeightedDigraph and KruskalMST calculate the MST for the graph in largeEWG.txt
5. Calculate the CPU Time.
6. The complexity of KruskalMST is O(E log V).



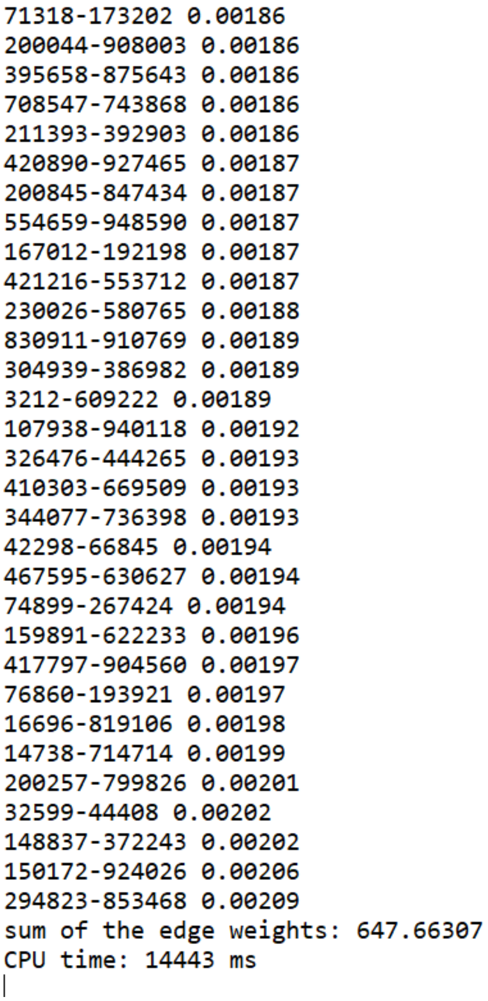
```
0 to 395136         no path
0 to 395137         no path
0 to 395138         no path
0 to 395139         no path
0 to 395140         no path
0 to 395141         no path
0 to 395142         no path
0 to 395143         no path
0 to 395144         no path
0 to 395145         no path
0->15786  0.00
15786->53370  0.00
53370->310931  0.00
310931->331670  0.00
331670->395146  0.00
0 to 395147         no path
0 to 395148         no path
0 to 395149         no path
0 to 395150         no path
0 to 395151         no path
0 to 395152         no path
0 to 395153         no path
0 to 395154         no path
0 to 395155         no path
0 to 395156         no path
0 to 395157         no path
0 to 395158         no path
0 to 395159         no path
0 to 395160         no path
0 to 395161         no path
0 to 395162         no path
0 to 395163         no path
0 to 395164         no path
0 to 395165         no path
0 to 395166         no path
```



```
0 to 999977         no path
0 to 999978         no path
0 to 999979         no path
0 to 999980         no path
0 to 999981         no path
0 to 999982         no path
0 to 999983         no path
0 to 999984         no path
0 to 999985         no path
0 to 999986         no path
0 to 999987         no path
0 to 999988         no path
0 to 999989         no path
0 to 999990         no path
0 to 999991         no path
0 to 999992         no path
0 to 999993         no path
0 to 999994         no path
0 to 999995         no path
0 to 999996         no path
0 to 999997         no path
0 to 999998         no path
0 to 999999         no path
CPU time: 5765 ms
```



```
322359-588831 0.00000
226378-832872 0.00000
36545-953493 0.00000
542175-789308 0.00000
45388-727937 0.00000
596289-597133 0.00000
280717-665905 0.00000
414340-633052 0.00000
5985-949642 0.00000
231068-689971 0.00000
165091-198261 0.00000
94069-337468 0.00000
770449-812037 0.00000
140155-708355 0.00000
560820-576481 0.00000
712420-816171 0.00000
423907-645514 0.00000
134045-504794 0.00000
887641-947987 0.00000
21075-853822 0.00000
149897-495994 0.00000
79466-213613 0.00000
450938-517684 0.00000
239197-680362 0.00000
430839-474201 0.00000
524613-750189 0.00000
602073-664421 0.00000
203730-814274 0.00000
776037-952649 0.00000
704912-724928 0.00000
66303-910474 0.00000
625271-965236 0.00000
376390-485190 0.00000
284187-856262 0.00000
```



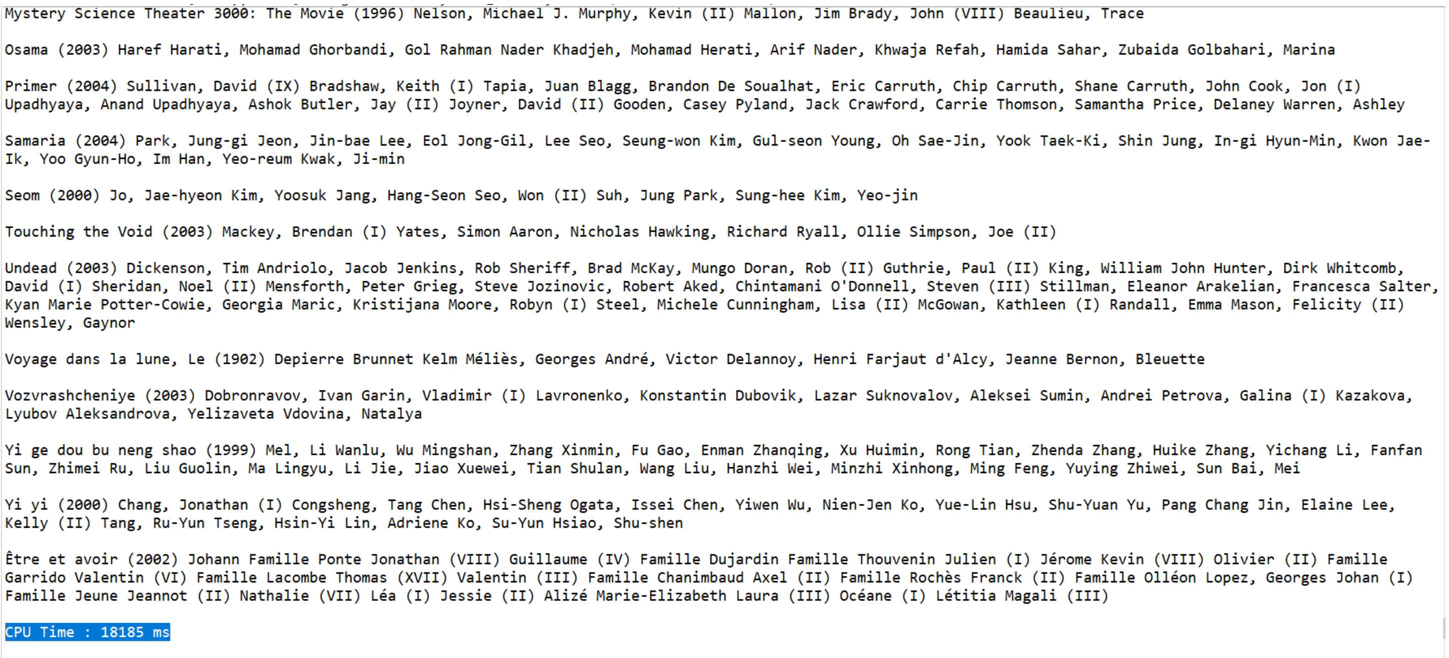
```
71318-173202 0.00186
200044-908003 0.00186
395658-875643 0.00186
708547-743868 0.00186
211393-392903 0.00186
420890-927465 0.00187
200845-847434 0.00187
554659-948590 0.00187
167012-192198 0.00187
421216-553712 0.00187
230026-580765 0.00188
830911-910769 0.00189
304939-386982 0.00189
3212-609222 0.00189
107938-940118 0.00192
326476-444265 0.00193
410303-669509 0.00193
344077-736398 0.00193
42298-66845 0.00194
467595-630627 0.00194
74899-267424 0.00194
159891-622233 0.00196
417797-904560 0.00197
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
sum of the edge weights: 647.66307
CPU time: 14443 ms
```

Question 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.

Solution 3: Please refer package solution, class QuestionThree.java and method main for the coded solution.

Steps to resolve are as follows:

1. Read the graph using SymbolGraph.java
2. Create an array of queue of the connected elements.
3. For each element of the queue, find the number of connected components using DFS.
4. Since the output is large I have attached a snippet of the output and the CPU time.
5. The worst case complexity of DFS is O(V+E).



```
Mystery Science Theater 3000: The Movie (1996) Nelson, Michael J. Murphy, Kevin (II) Mallon, Jim Brady, John (VIII) Beaulieu, Trace

Osama (2003) Haref Harati, Mohamad Ghorbandi, Gol Rahman Nader Khadjeh, Mohamad Herati, Arif Nader, Khwaja Refah, Hamida Sahar, Zubaida Golbahari, Marina

Primer (2004) Sullivan, David (IX) Bradshaw, Keith (I) Tapia, Juan Blagg, Brandon De Soualhat, Eric Carruth, Chip Carruth, Shane Carruth, John Cook, Jon (I)
Upadhyaya, Anand Upadhyaya, Ashok Butler, Jay (II) Joyner, David (II) Gooden, Casey Pyland, Jack Crawford, Carrie Thomson, Samantha Price, Delaney Warren, Ashley

Samaria (2004) Park, Jung-gi Jeon, Jin-bae Lee, Eol Jong-Gil, Lee Seo, Seung-won Kim, Gul-seon Young, Oh Sae-Jin, Yook Taek-Ki, Shin Jung, In-gi Hyun-Min, Kwon Jae-
Ik, Yoo Gyun-Ho, Im Han, Yeo-reum Kwak, Ji-min

Seom (2000) Jo, Jae-hyeon Kim, Yoosuk Jang, Hang-Seon Seo, Won (II) Suh, Jung Park, Sung-hee Kim, Yeo-jin

Touching the Void (2003) Mackey, Brendan (I) Yates, Simon Aaron, Nicholas Hawking, Richard Ryall, Ollie Simpson, Joe (II)

Undead (2003) Dickenson, Tim Andriolo, Jacob Jenkins, Rob Sheriff, Brad McKay, Mungo Doran, Rob (II) Guthrie, Paul (II) King, William John Hunter, Dirk Whitcomb,
David (I) Sheridan, Noel (II) Mensforth, Peter Grieg, Steve Jozinovic, Robert Aked, Chintamani O'Donnell, Steven (III) Stillman, Eleanor Arakelian, Francesca Salter,
Kyan Marie Potter-Cowie, Georgia Maric, Kristijana Moore, Robyn (I) Steel, Michele Cunningham, Lisa (II) McGowan, Kathleen (I) Randall, Emma Mason, Felicity (II)
Wensley, Gaynor

Voyage dans la lune, Le (1902) Depierre Brunnet Kelm Méliès, Georges André, Victor Delannoy, Henri Farjaut d'Alcy, Jeanne Bernon, Bleuette

Vozvrashcheniye (2003) Dobronravov, Ivan Garin, Vladimir (I) Lavronenko, Konstantin Dubovik, Lazar Suknovalov, Aleksei Sumin, Andrei Petrova, Galina (I) Kazakova,
Lyubov Aleksandrova, Yelizaveta Vdovina, Natalya

Yi ge dou bu neng shao (1999) Mel, Li Wanlu, Wu Mingshan, Zhang Xinmin, Fu Gao, Enman Zhanqing, Xu Huimin, Rong Tian, Zhenda Zhang, Huike Zhang, Yichang Li, Fanfan
Sun, Zhimei Ru, Liu Guolin, Ma Lingyu, Li Jie, Jiao Xuewei, Tian Shulan, Wang Liu, Hanzhi Wei, Minzhi Xinhong, Ming Feng, Yuying Zhiwei, Sun Bai, Mei

Yi yi (2000) Chang, Jonathan (I) Congsheng, Tang Chen, Hsi-Sheng Ogata, Issei Chen, Yiwen Wu, Nien-Jen Ko, Yue-Lin Hsu, Shu-Yuan Yu, Pang Chang Jin, Elaine Lee,
Kelly (II) Tang, Ru-Yun Tseng, Hsin-Yi Lin, Adriene Ko, Su-Yun Hsiao, Shu-shen

Être et avoir (2002) Johann Famille Ponte Jonathan (VIII) Guillaume (IV) Famille Dujardin Famille Thouvenin Julien (I) Jérome Kevin (VIII) Olivier (II) Famille
Garrido Valentin (VI) Famille Lacombe Thomas (XVII) Valentin (III) Famille Chanimbaud Axel (II) Famille Rochès Franck (II) Famille Olléon Lopez, Georges Johan (I)
Famille Jeune Jeannot (II) Nathalie (VII) Léa (I) Jessie (II) Alizé Marie-Elizabeth Laura (III) Océane (I) Létitia Magali (III)

CPU Time : 18185 ms
```

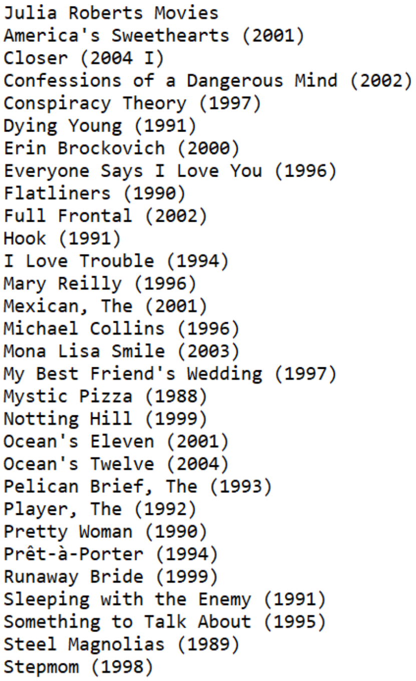
Question 4: Write a program that finds the movies starred by a particular actor. Show the movies starred by Leonardo DiCaprio. Show the movies starred by Julia Roberts, by Hugh Grant, and by both of them.

Solution 4: Please refer package solution, class QuestionFour.java and method main for the coded solution.

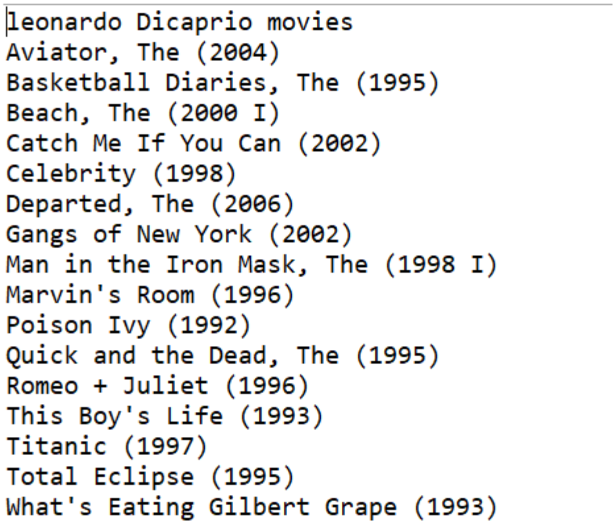
Steps to resolve are as follows:

1. For each line in the movie database, divide it according to the delimeter
2. Check each element of every line, against the name of the star.
3. If the name matches, the movie name is added to the list.

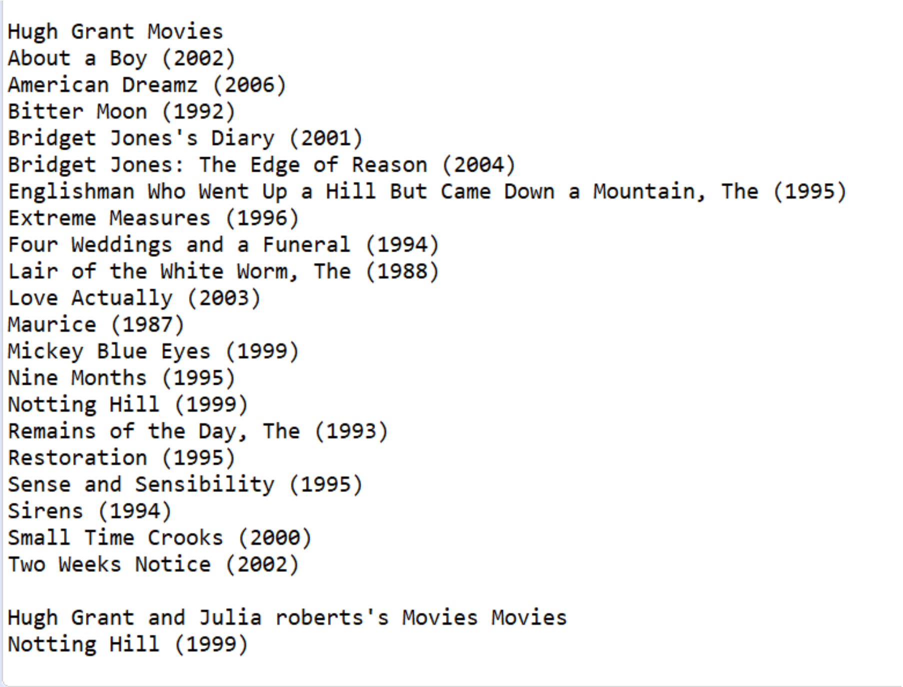
OUTPUT



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



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



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

Hugh Grant and Julia roberts's Movies Movies
Notting Hill (1999)
```

Question 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”).

Solution 4: Please refer package solution, class QuestionFive.java and method main for the coded solution.

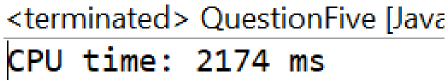
Steps to resolve are as follows:

1. Using In.java read the file Chip-seq-reads-1M.dat.
2. Create 4 arrayList and add the required lines.
3. Now each arrayList is written into file using BufferedWriter.
4. Sort each arrayList using Collections.sort().
5. Now each sorted list is again written into respective files using BufferedWriter.
6. Now, read the sorted list using “In” class provided in class.
7. Use priority queue to sort the streams and combine them into a combined sorted list.
8. Again, write it into the file with help of BufferedWriter.

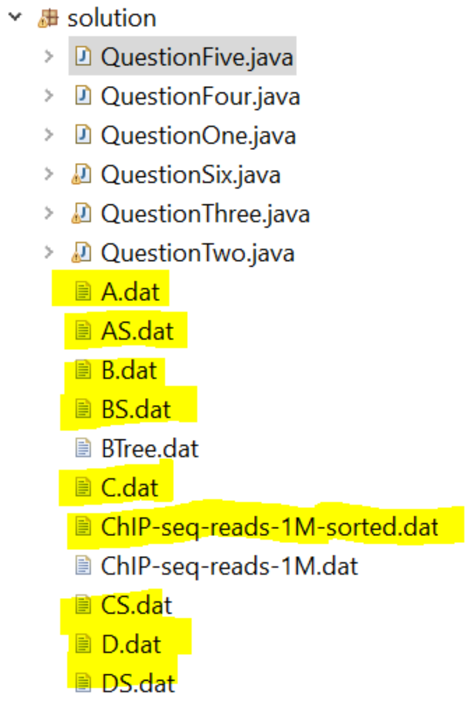
OUTPUT

Note. All the files are created under the package solution.

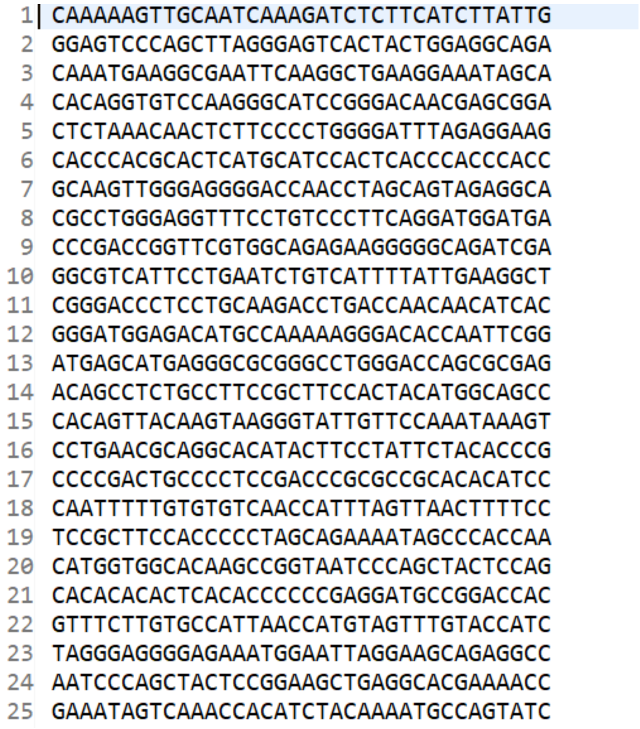
Snippets are added below



```
CPU time: 2174 ms
```

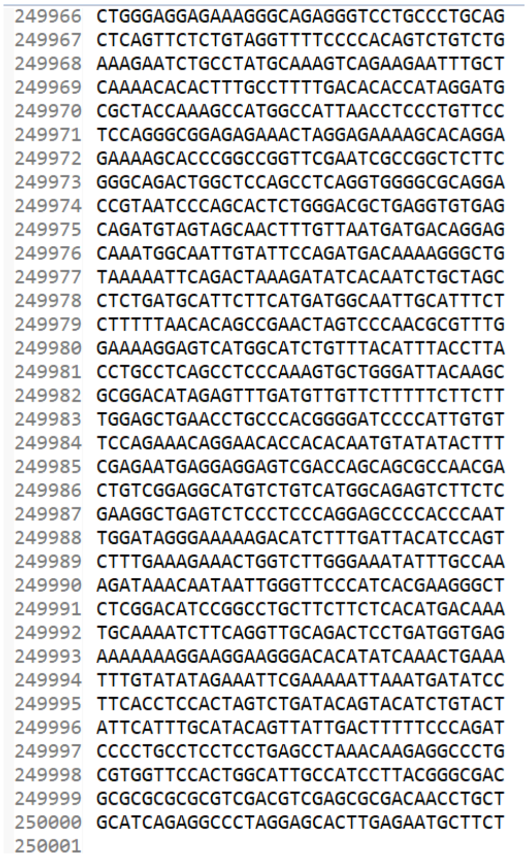


```
solution
  QuestionFive.java
  QuestionFour.java
  QuestionOne.java
  QuestionSix.java
  QuestionThree.java
  QuestionTwo.java
  A.dat
  AS.dat
  B.dat
  BS.dat
  BTree.dat
  C.dat
  ChIP-seq-reads-1M-sorted.dat
  ChIP-seq-reads-1M.dat
  CS.dat
  D.dat
  DS.dat
```



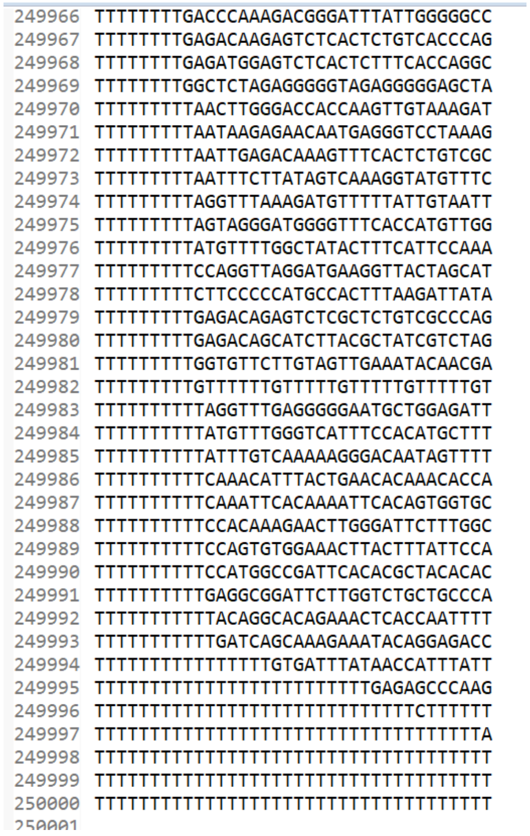
```
CAAAAAGTTGCAATCAAAGATCTCTTCATCTTATTG
GGAGTCCCAGCTTAGGGAGTCACTACTGGAGGCAGA
CAAATGAAGGCGAATTCAAGGCTGAAGGAAATAGCA
CACAGGTGTCCAAGGGCATCCGGGACAACGAGCGGA
CTCTAAACAACTCTTCCCCTGGGGATTTAGAGGAAG
CACCCACGCACTCATGCATCCACTCACCCACCCACC
GCAAGTTGGGAGGGGACCAACCTAGCAGTAGAGGCA
CGCCTGGGAGGTTTCCTGTCCCTTCAGGATGGATGA
CCCGACCGGTTCGTGGCAGAGAAGGGGGCAGATCGA
GGCGTCATTCCTGAATCTGTCATTTTATTGAAGGCT
CGGGACCCTCCTGCAAGACCTGACCAACAACATCAC
GGGATGGAGACATGCCAAAAAGGGACACCAATTCGG
ATGAGCATGAGGGCGCGGGCCTGGGACCAGCGCGAG
ACAGCCTCTGCCTTCCGCTTCCACTACATGGCAGCC
CACAGTTACAAGTAAGGGTATTGTTCCAAATAAAGT
CCTGAACGCAGGCACATACTTCCTATTCTACACCCG
CCCCGACTGCCCCTCCGACCCGCGCCGCACACATCC
CAATTTTTGTGTGTCAACCATTTAGTTAACTTTTCC
TCCGCTTCCACCCCCTAGCAGAAAATAGCCCACCAA
CATGGTGGCACAAGCCGGTAATCCCAGCTACTCCAG
CACACACACTCACACCCCCGAGGATGCCGGACCAC
GTTTCTTGTGCCATTAACCATGTAGTTTGTACCATC
TAGGGAGGGGAGAAATGGAATTAGGAAGCAGAGGCC
AATCCCAGCTACTCCGGAAGCTGAGGCACGAAAACC
GAAATAGTCAAACCACATCTACAAAATGCCAGTATC
```

A.DAT



```
CTGGGAGGAGAAAGGGCAGAGGGTCCTGCCCTGCAG
CTCAGTTCTCTGTAGGTTTTCCCCACAGTCTGTCTG
AAAGAATCTGCCTATGCAAAGTCAGAAGAATTTGCT
CAAAACACACTTTGCCTTTTGACACACCATAGGATG
CGCTACCAAAGCCATGGCCATTAACCTCCCTGTTCC
TCCAGGGCGGAGAGAAACTAGGAGAAAAGCACAGGA
GAAAAGCACCCGGCCGGTTCGAATCGCCGGCTCTTC
GGGCAGACTGGCTCCAGCCTCAGGTGGGGCGCAGGA
CCGTAATCCCAGCACTCTGGGACGCTGAGGTGTGAG
CAGATGTAGTAGCAACTTTGTTAATGATGACAGGAG
CAAATGGCAATTGTATTCCAGATGACAAAAGGGCTG
TAAAAATTCAGACTAAAGATATCACAATCTGCTAGC
CTCTGATGCATTCTTCATGATGGCAATTGCATTTCT
CTTTTTAACACAGCCGAACTAGTCCCAACGCGTTTG
GAAAAGGAGTCATGGCATCTGTTTACATTTACCTTA
CCTGCCTCAGCCTCCCAAAGTGCTGGGATTACAAGC
GCGGACATAGAGTTTGATGTTGTTCTTTTTCTTCTT
TGGAGCTGAACCTGCCCACGGGGATCCCCATTGTGT
TCCAGAAACAGGAACACCACACAATGTATATACTTT
CGAGAATGAGGAGGAGTCGACCAGCAGCGCCAACGA
CTGTCGGAGGCATGTCTGTCATGGCAGAGTCTTCTC
GAAGGCTGAGTCTCCCTCCCAGGAGCCCCACCCAAT
TGGATAGGGAAAAAGACATCTTTGATTACATCCAGT
CTTTGAAAGAAACTGGTCTTGGGAAATATTTGCCAA
AGATAAACAATAATTGGGTTCCCATCACGAAGGGCT
CTCGGACATCCGGCCTGCTTCTTCTCACATGACAAA
TGCAAAATCTTCAGGTTGCAGACTCCTGATGGTGAG
AAAAAAAGGAAGGAAGGGACACATATCAAACTGAAA
TTTGTATATAGAAATTCGAAAAATTAAATGATATCC
TTCACCTCCACTAGTCTGATACAGTACATCTGTACT
ATTCATTTGCATACAGTTATTGACTTTTTCCCAGAT
CCCCTGCCTCCTCCTGAGCCTAAACAAGAGGCCCTG
CGTGGTTCCACTGGCATTGCCATCCTTACGGGCGAC
GCGCGCGCGCGTCGACGTCGAGCGCGACAACCTGCT
GCATCAGAGGCCCTAGGAGCACTTGAGAATGCTTCT
```

B.dat



```
TTTTTTTTGACCCAAAGACGGGATTTATTGGGGGCC
TTTTTTTTGAGACAAGAGTCTCACTCTGTCACCCAG
TTTTTTTTGAGATGGAGTCTCACTCTTTCACCAGGC
TTTTTTTTGGCTCTAGAGGGGGTAGAGGGGGAGCTA
TTTTTTTTTAACTTGGGACCACCAAGTTGTAAAGAT
TTTTTTTTTAATAAGAGAACAATGAGGGTCCTAAAG
TTTTTTTTTAATTGAGACAAAGTTTCACTCTGTCGC
TTTTTTTTTAATTTCTTATAGTCAAAGGTATGTTTC
TTTTTTTTTAGGTTTAAAGATGTTTTTATTGTAATT
TTTTTTTTTAGTAGGGATGGGGTTTCACCATGTTGG
TTTTTTTTTATGTTTTGGCTATACTTTCATTCCAAA
TTTTTTTTTCCAGGTTAGGATGAAGGTTACTAGCAT
TTTTTTTTTCTTCCCCCATGCCACTTTAAGATTATA
TTTTTTTTTGAGACAGAGTCTCGCTCTGTCGCCCAG
TTTTTTTTTGAGACAGCATCTTACGCTATCGTCTAG
TTTTTTTTTGGTGTTCTTGTAGTTGAAATACAACGA
TTTTTTTTTGTTTTTTGTTTTTGTTTTTGTTTTTGT
TTTTTTTTTTAGGTTTGAGGGGGAATGCTGGAGATT
TTTTTTTTTTATGTTTGGGTCATTTCCACATGCTTT
TTTTTTTTTTATTTGTCAAAAAGGGACAATAGTTTT
TTTTTTTTTTCAAACATTTACTGAACACAAACACCA
TTTTTTTTTTCAAATTCACAAAATTCACAGTGGTGC
TTTTTTTTTTCCACAAAGAACTTGGGATTCTTTGGC
TTTTTTTTTTCCAGTGTGGAAACTTACTTTATTCCA
TTTTTTTTTTCCATGGCCGATTCACACGCTACACAC
TTTTTTTTTTGAGGCGGATTCTTGGTCTGCTGCCCA
TTTTTTTTTTTACAGGCACAGAAACTCACCAATTTT
TTTTTTTTTTTGATCAGCAAAGAAATACAGGAGACC
TTTTTTTTTTTTTTTTGTGATTTATAACCATTTATT
TTTTTTTTTTTTTTTTTTTTTTTTTGAGAGCCCAAG
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTCTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTA
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
TTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTTT
```

CS.dat

Question 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).

Solution 6: Please refer package solution, class QuestionFive.java and method main for the coded solution.

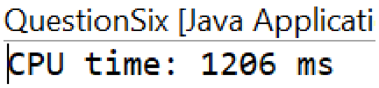
Steps to resolve are as follows:

1. Using In.java read the file Chip-seq-reads-1M.dat.
2. All the lines are inserted into Btree, provided in class.
3. Redirect the output of the console to file BTree.dat
4. Print the tree using Btree.toString();

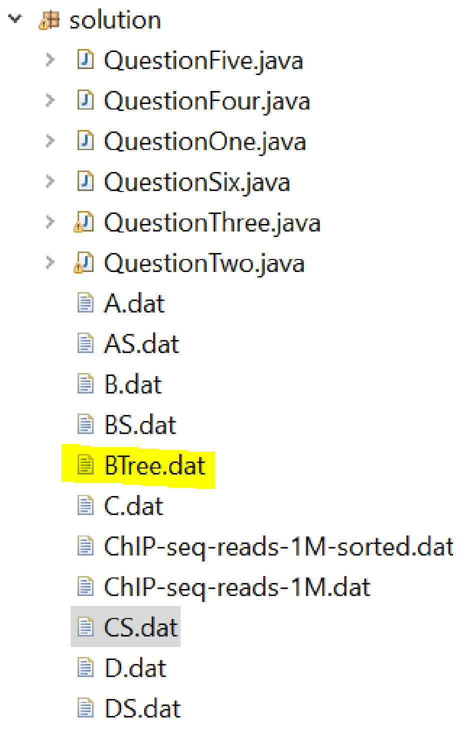
OUTPUT

Note. The file is created under the package solution.

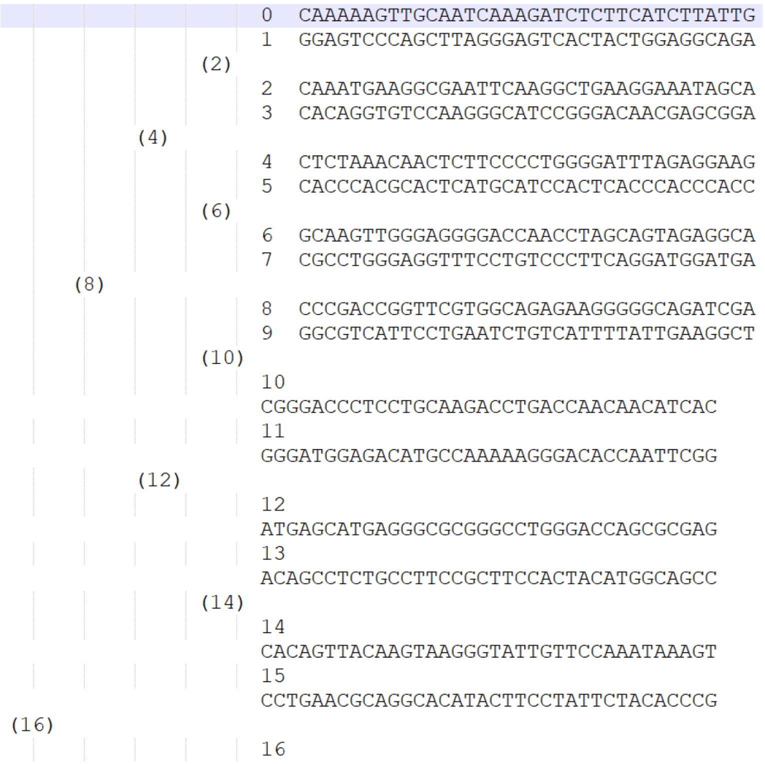
Snippet is added below



```
QuestionSix [Java Applicati
CPU time: 1206 ms
```



```
⌄ solution
   > QuestionFive.java
   > QuestionFour.java
   > QuestionOne.java
   > QuestionSix.java
   > QuestionThree.java
   > QuestionTwo.java
     A.dat
     AS.dat
     B.dat
     BS.dat
     BTree.dat
     C.dat
     ChIP-seq-reads-1M-sorted.dat
     ChIP-seq-reads-1M.dat
     CS.dat
     D.dat
     DS.dat
```



```
                0  CAAAAAGTTGCAATCAAAGATCTCTTCATCTTATTG
                1  GGAGTCCCAGCTTAGGGAGTCACTACTGGAGGCAGA
           (2)
                2  CAAATGAAGGCGAATTCAAGGCTGAAGGAAATAGCA
                3  CACAGGTGTCCAAGGGCATCCGGGACAACGAGCGGA
      (4)
                4  CTCTAAACAACTCTTCCCCTGGGGATTTAGAGGAAG
                5  CACCCACGCACTCATGCATCCACTCACCCACCCACC
           (6)
                6  GCAAGTTGGGAGGGGACCAACCTAGCAGTAGAGGCA
                7  CGCCTGGGAGGTTTCCTGTCCCTTCAGGATGGATGA
  (8)
                8  CCCGACCGGTTCGTGGCAGAGAAGGGGGCAGATCGA
                9  GGCGTCATTCCTGAATCTGTCATTTTATTGAAGGCT
           (10)
                10
                CGGGACCCTCCTGCAAGACCTGACCAACAACATCAC
                11
                GGATGGAGACATGCCAAAAAGGACACCAATTCGG
      (12)
                12
                ATGAGCATGAGGGCGCGGGCCTGGGACCAGCGCGAG
                13
                ACAGCCTCTGCCTTCCGCTTCCACTACATGGCAGCC
           (14)
                14
                CACAGTTACAAGTAAGGGTATTGTTCCAAATAAAGT
                15
                CCTGAACGCAGGCACATACTTCCTATTCTACACCCG
(16)
                16
```

Question 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.

Solution. The time taken in question 5 is 2174 ms and in question 6 is 1206 ms. The average case time complexity is O(log n) for both solutions.