2801ICT Computing Algorithms – Assignment (30%)

Due Week 11-12: 31 May 2023 11:59 PM

This assignment must be done individually. The submission is done via L@G.

Students should submit multiple files:

1. A report and a source code file for problem 1
2. A report and a source code file for problem 2

If you have trouble uploading multiple files, you can upload them all in a zip file and submit.

Problem 1 (Dynamic Programming) – 20%

Problem Description

Samsung and *Apple* are two tech giants and the biggest rivals of each other. Recently, *Samsung* has lost a lawsuit against *Apple* and the court asked *Samsung* to pay one billion dollars to *Apple*. *Samsung* has no option other than paying the money, but the *Samsung CEO* has come up with an innovative idea to take revenge on *Apple*. He decided to pay the money in coins. There are unlimited coins in *Samsung*'s vault and the values of coins are all primes e.g. 2, 3, 5, 7, … including 1 and a *Gold coin*. The *Gold coin* is equivalent to the amount to be paid.

Assume that you are the principal software engineer at Samsung. One morning, the *CEO* called you and described his plan to pay the money to *Apple* in coins. He wants you to write a program that can calculate the total number of ways a given amount can be paid using a specified number of coins. For example, $5 can be paid in 6 different ways: 5 (Gold coin), 2+3, 1+1+3, 1+2+2, 1+1+1+2 and 1+1+1+1+1 using all possible coins. However, $8 can be paid in two ways 2+3+3 and 5+2+1 if three coins are used only.

Input format

Input.txt is a file that contains inputs in every line. An input may have a single integer which represents the money you have to pay using all possible combinations of coins. An Input may have two integers where the first integer represents the amount to be paid and the second integer represents the number of coins you should use to pay the amount. Additionally, if an input contains 3 integers, the first integer is the amount to be paid, the second and the third integers represent the range of number of coins that can be used. For example, if the second number is 5 and the third number is 8, then your program should display the total number of ways the amount can be paid using 5, 6, 7 and 8 coins.

Output format

Output.txt contains corresponding outputs to every line of *Input.txt*. The output is an integer number that shows the total number of ways the money can be paid.

Sample Input	Sample Output
5	6

8 3 8 2 5	2 10

Explanation of Sample Output:

6---> 5, 2+3=5, 1+1+3=5, 1+2+2=5, 1+1+1+2=5, 1+1+1+1+1=5

2---> 5+2+1=8, 3+3+2=8

10---> 1+7=8, 5+3=8, 3+3+2=8, 5+2+1=8, 2+2+2+2=8, 5+1+1+1=8, 3+2+2+1=8, 3+3+1+1=8, 1+1+2+2+2=8, 1+1+1+2+3=8

Sample Results for C++ Implementation

Input	Output	CPU Time (Secs)	Input	Output	CPU Time (Secs)
5	6	0.000000	20 10 15	57	0.000000
6 2 5	7	0.000000	100 5 10	14839	0.000000
6 1 6	9	0.000000	100 8 25	278083	0.006000
8 3	2	0.000000	300 12	4307252	2.155000
8 2 5	10	0.000000	300 10 15	32100326	22.968000

Sample Results for Python Implementation

Input	Output	CPU Time (Secs)	Input	Output	CPU Time (Secs)
5	6	0.000523	20 10 15	57	0.007611
6 2 5	7	0.001361	100 5 10	14839	0.912590
6 1 6	9	0.001741	100 8 25	278083	39.795810
8 3	2	0.000537	300 12	4307252	372.670860
8 2 5	10	0.001271	300 10 15	32100326	3992.311160

Notes

1. You must design your own algorithm and write the program code submitted. The program must be able to be run from the command line passing the path to the *Input.txt* file as an argument. The naming convention for your program file is: <student_number>_Pay_in_Coins.py (or <student_number>_Pay_in_Coins.cpp or <student_number>_Pay_in_Coins.java)(without the <>) and try and keep the program within a single file.
2. You must produce a report describing your algorithm (preferably including the pseudocode), correctness of your algorithm, the testcases of the problem (sample *input and output* are provided) and performance analysis.

Marking

This problem is worth 20% of your final grade. The problem will be marked out of 100 and marks will be allocated as follows:

1. (**Algorithmic Design**) Overview (describe the problem in your own words), Algorithm Description (the main idea behind your solution), and preferably Algorithm Pseudo-Code**: 50 marks**
2. (**Implementation**) Quality of code (e.g., appropriate naming conventions, code comments, class structure, method structure, appropriate use of data structures etc.): **30 marks**
3. (**Testcases and Algorithm Analysis**) Your own testcases (input and output) of the problem and performance analysis: **20 marks.**

Please note that all submitted assignments will be analysed by a plagiarism detector including the submitted source code of the program. If any form of plagiarism is found, it will be dealt with in accordance with university policy: https://www.griffith.edu.au/academic-integrity/academic-misconduct.

Problem 2 (Graph Algorithms) – 10%

Background

Melanie is a second-year computer science student currently studying at Griffith School of ICT. One day, Melanie's father came to her and told her that he wants to travel from Southport to Brisbane CBD and asks Melanie to recommend exactly K-shortest loopless paths for some reason. Luckily, Melanie is undertaking the "2801ICT: Computing Algorithms" course. Though, Melanie knows both Dijkstra's algorithm and Bellman Ford's algorithm very well as these two algorithms have been taught already in the class, she is not exactly sure how to solve this K-**shortest loopless paths** problem for the Queensland road network graph. This is an interesting problem and it could be solved by the following requirement. Melanie's father is happy if and only if the first recommended path is the **shortest path** between Southport to Brisbane CBD and the **rest of the $K-1$ paths are approximated shortest paths** as these paths are his backup paths.

Problem Description

The shortest path problem is about finding a path between **2** vertices in a graph such that the total sum of the edges weights is minimum. This problem could be solved easily using (BFS) if all edge weights were **1**, but here weights can take any value. There are TWO popular algorithms in the literature for computing the shortest paths from the source vertex to all other vertices in a graph: (i) Dijkstra's Algorithm; and (ii) Bellman Ford's Algorithm. The Bellman Ford's algorithm exploits the idea that a shortest path contains at most $n-1$ edges, and a shortest path cannot have a cycle. The Bellman Ford's algorithm works for graphs with negative weight cycle(s). The time complexity of Dijkstra's Algorithm with min-priority queue is $O(V + E \log V)$. On the other hand, the time complexity of Bellman Ford algorithm is relatively high $O(V.E)$, in case $E = V^2$, $O(V^3)$. Both Dijkstra's and Bellman Ford's algorithms unable to solve the this K-shortest loopless paths (KSP) problem. There exist many works in the literature to solve KSP problem. Some useful references are given at the end of this document.

In this problem, you are given 2 integers (N, M), N is the number of vertices, M is the number of edges. You will also be given a_i, b_i, w_i where a_i and b_i represents an edge from a vertex a_i to a vertex b_i and w_i respresents the weight of that edge. Finally, you will be given two vertices s and d, where s denotes the source vertex (Southport) and b denotes the destination vertex (Brisbane CBD) and the value of K.

For each input file, print the length of the K-shortest loopless paths separated by commas and these paths must include the **shortest path between s and b** and the **rest of the $K-1$ paths are approximated shortest paths**.

Sample Input: Consider the following directed graph as an example.

6 9

C D 3

C E 2

D F 4

E D 1

E F 2

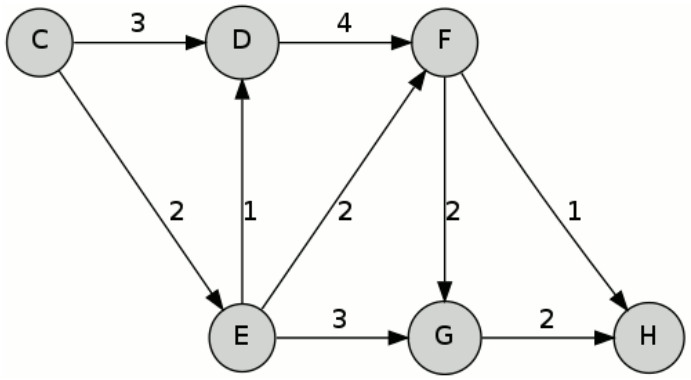
E G 3

F G 2

F H 1

G H 2

C H 3



Sample Output: 5, 7, 8

The 3 shortest paths from source **C** to destination **H** are as follows:

P_1 = C→E→F→H, Cost (P_1) = 5

P_2 = C→E→G→H, Cost (P_2) = 7

P_3 = C→D→F→H, Cost (P_3) = 8

Sample Results for Java Implementation (Yen Algorithm)

Input (k=?)	Output	CPU Time (millisecs)
2	1038.57, 1042.38	1286
3	1038.57, 1042.38, 1043.02	2007
5	1038.57, 1042.38, 1043.02, 1043.29, 1044.05	7767
7	1038.57, 1042.38, 1043.02, 1043.29, 1044.05, 1044.12, 1044.63	10897
10	1038.57, 1042.38, 1043.02, 1043.29, 1044.05, 1044.12, 1044.63, 1044.90, 1045.73, 1045.97	20345

*sample input file (finalInput.txt is uploaded at L@G, you can change the k-value in the last line)

Notes

1. You must design your own algorithm and write the program code submitted. The program must be able to be run from the command line passing the path to the input.txt file as an argument.
2. The naming convention for your program file is: <student_number>_ k_shortest_paths.py (or <student_number>_ k_shortest_paths.cpp or <student_number>_ k_shortest_paths.java) (without the <>) and try and keep the program within a single file.

Marking

This porblem is worth 10% of your final grade. The problem will be marked out of 100 and marks will be allocated as follows:

1. (**Algorithmic Design**) Overview (describe the problem in your own words), Algorithm

Description (the main idea behind your solution), and preferably Algorithm Pseudo-Code**: 50 marks**

2. (**Implementation**) Quality of code (e.g. appropriate naming conventions, code comments, class structure, method structure, appropriate use of data structures etc.)**: 30 marks**
3. (**Testcase and Algorithm Analysis**) Your testcases of the problem (sample testcase was provided) and performance (time-complexity) analysis: **20 marks**

Useful Articles for KSP problems

1. Yen, J.Y., 1971. Finding the k shortest loopless paths in a network. *management Science*, 17(11), pp.712-716.
2. Eppstein, D., 1998. Finding the k shortest paths. *SIAM Journal on computing*, 28(2), pp.652-673.
3. Martins, E.D.Q.V., Pascoal, M.M.B. and Dos Santos, J.L.E., 1998. The k shortest paths problem.
4. Liu, H., Jin, C., Yang, B. and Zhou, A., 2018. Finding top-k shortest paths with diversity. *IEEE Transactions on Knowledge and Data Engineering*, 30(3), pp.488-502.