CSE531: Algorithm Analysis and Design Fall 2024

Programming Project 1

Instructor: Kelin Luo **Deadline: Oct/21/2024**

Your Name: _____ Your Student ID: _____

Problems	1	2	3	4	Total
Max. Score	10	10	10	10	40
Your Score					

Submission requirement:

- You should submit **each source file for each problem**. Name your file: Problem number_Last name_First name_YourStudents ID Number_ProjectNumber.
  Example: 1_Doe_John_55552222_P1
- Allowed programming languages: Python. Use Python version ≥ 3.4. You can run "python –version" to check the version. Please do not use any built-in functions unless they are explicitly stated in the provided code. Do not use any third-party packages other than the standard library.
- For each problem, please modify the provided file to include the algorithm (For example, 1_Doe_John_55552222_P1.py). Note that the template code is just for the student to refer to get the correct input and output format, the student can revise the template based on their own code.
- The auto similarity-check will be applied for the programming submissions.
- The P1 deadline is 21 Oct, 11:59PM EST. Late submissions (within 24 hours with 25 % penalty or 48 hours with a 50% penalty) should be submitted via Email to Instructor and Head TAs. Submissions will close 48 hours after the deadline.
- Only the most recent submission is kept on Brightspace. Note that in each submission, please submit **all** your source files. Only files with extension .py will be accepted for the submission. You should view your submission after you upload it to make sure that it is not corrupted or malformed. You are responsible for making sure your submission went through successfully.
- The grading scheme and test cases will be released when the grades for Programming Project 1 are published.

Problem 1 (10 points). You need to implement the BFS based or DFS based algorithm for the connectivity problem. An $O(n+m)$-time algorithm is suffice to pass any feasible test cases.

(a) **Input** You need to read the input from the console. In the first line of the input, we have two positive integers n and m. n is the number of vertices in the graph and m is the number of edges in the graph. The vertices are indexed from 1 to n. You can assume that $1 \leq n \leq 1000$ and $1 \leq m \leq 100000$. In the second line of the input, we have two positive integers s and t where $1 \leq s \leq n$ and $1 \leq t \leq n$. In the next m lines, each line contains 2 integers: u and v. This indicates that there is an edge $\{u, v\}$ in the graph. *The input graph is an undirected graph. The input graph may not be connected.*

(b) **Output** You need to output to the console. The output is either "yes" or "no". If the answer is "no", then there is no path connecting s to t in the given graph. otherwise, there is there is a path connecting s to t.

Below are two examples of input and output:

Example input:	Example output:
6 7 1 5 1 2 5 6 2 3 4 5 4 6 1 5 3 5	yes

Example input:	Example output:
6 6 2 4 1 2 5 6 2 3 4 5 4 6 1 3	no

Problem 2 (10 points). You need to implement the algorithm for both the interval scheduling and the interval covering problem. Please provide the solution for the interval scheduling problem in the first line of the output, and the solution for the interval covering problem in the second line of the output. An $O(n^2)$-time algorithm is sufficient to pass any feasible test cases.

(a) **Input** You need to read the input from the console. In the first line of the input, we have three positive integers n, u and v. n is the number of intervals. The intervals are indexed from 1 to n. You can assume that $1 \leq n \leq 10000$. You can assume that $1 \leq u \leq v \leq 100000$. In the next n lines, each line contains 2 integers: s_i and f_i. This indicates that there is an interval $[s_i, f_i)$. You can assume that $1 \leq s_i \leq 100000$ and $1 \leq f_i \leq 100000$.

(b) **Output** You need to output to the console. The output consists of two lines: the first line displays the maximum number of jobs for the interval scheduling problem, and the second line shows either the minimum number of jobs required to cover the interval from u to v for the interval covering problem or "no" if no feasible covering exists.

Below are two examples of input and output:

Example input:	Example output:
10 3 7 1 2 5 6 1 3 3 4 4 20 5 6 4 5 7 9 5 17 2 4	5 2

Example input:	Example output:
4 2 9 1 2 5 11 1 3 3 4	3 no

Problem 3 (10 points). You need to implement the LIFO algorithm and the FF algorithm (Furthest-in-Future algorithm) for the offline caching problem. Please provide the sum of the LIFO solution and the FF solution in the first line of the output, and the difference between the LIFO solution and the FF solution in the second line of the output. An $O(n + mk)$-time algorithm is sufficient to pass any feasible test cases.

(a) **Input** You need to read the input from the console. In the first line of the input, we have three positive integer k, n and m. k is the size of the cache, n is the number of pages, and m is the number of page requests. The pages are indexed from 1 to n. You can assume that $1 \leq k \leq 100$, $1 \leq n \leq 1000$ and $1 \leq m \leq 10000$. In the next m lines, each line contains one integer: $\rho_i \in [n]$. This indicates that there is a request ρ_i in the sequence request.

(b) **Output** You need to output to the console. The output consists of two lines: the sum of the LIFO solution and the FF solution in the first line, and the difference between the LIFO solution and the FF solution in the second line.

Example: Below are two examples of input and output:

Example input:	Example output:
3 10 10 5 1 3 1 5 6 3 2 5 1	12 0

Example input:	Example output:
3 4 6 1 3 4 1 2 4	9 1

Problem 4 (10 points). You need to implement the quicksort algorithm to sort a list of cities based on their populations in ascending order. If two cities have the same population, they should be sorted by their city index in ascending order. An $O(n \log n)$-time algorithm is sufficient to pass any feasible test cases.

Input You need to read the input from the console. The input consists of n lines of data. Each line i contains two positive integers: a_i, which is the index of the city, and p_i, representing the population of the city. You can assume that $1 \leq a_i \leq 1000$ and $1000 \leq p_i \leq 1000000$. Note that it is not need to take n as input. You can always assume the input is ending by a EOF.

Output You need to output to console. The output from line 1 to line n, you need to output the cities in ascending order of population. If two cities have the same population, they should be listed in ascending order of their city index.

Example: Below are two examples of input and output:

Example input:	Example output:
2 10000	1 10000
1 10000	2 10000
4 20000	10 10000
5 20000	4 20000
6 30000	5 20000
3 42000	6 30000
9 86000	3 42000
10 10000	7 50000
7 50000	11 67000
11 67000	9 86000
12 100000	8 93000
8 93000	12 100000

Example input:	Example output:
2 10000	1 10000
1 10000	2 10000
3 20000	4 10000
5 10000	5 10000
4 10000	3 20000