Project Assignment 1

Course: Algorithm Design and Analysis **Semester**: Spring 2024

Instructor: Shi Li **Due Date**: **2024/5/19**

Problem 1. You need to implement the algorithm that finds the bridges in an (undirected) graph. Recall that an edge e in a graph $G = (V, E)$ is a bridge iff its removal will increase the number of connected components of G.

Input:

- The input is taken from the standard input (console).
- The first line of input contains two integers n and m, where n is the number of vertices in the graph, and m is the number of edges. The vertices are indexed from 1 to n.
- The next m lines contain two integers each, representing the m edges of the graph. Each edge is given as two integers u and v with $1 \leq u < v \leq n$, denoting that there is an edge between u and v.
- It is guaranteed that the graph does not have parallel edges.

Output:

- The output is printed to the standard output (console).
- In the first line, print the number m' of bridges.
- In the following m' lines, each bridge in the graph should be printed on a separate line. Represent each bridge using the indices u and v, where $u < v$ are the two end-vertices. The m' bridges should be output in lexicographic order: they are sorted in ascending order of the first index and ties are broken in ascending order of the second index.

Example Input:	**Example Output**:	This indicates that there are two bridges: $(3, 4)$ and $(4, 5)$.
5 5 1 2 1 3 2 3 3 4 4 5	2 3 4 4 5	

Constraints:

- $1 \leq n \leq 10000$.
- $1 \leq m \leq 10^6$.
- It is expected that your program terminates in 10 seconds.

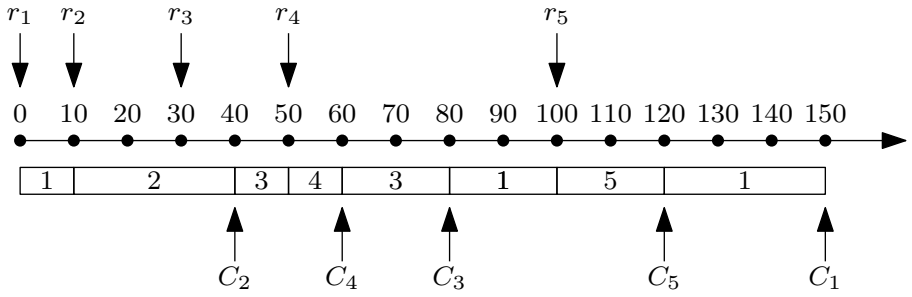
Problem 2. Given a set of n jobs indexed as $1, 2, \cdots, n$. Each job j has a release time r_j indicating when it becomes available for processing, and a processing time indicating how long it takes to complete. The machine can process jobs preemptively, meaning that it can interrupt the processing of a job to process another job and then resume the interrupted job later. There is no overhead in switching the processing of jobs. A job j is completed when it is processed for p_j units of time, and its completion time C_j is the time when it is completed. Our goal is

to decide a schedule for all the jobs so as to minimize the total completion time of all jobs, i.e., $\sum_{j=1}^{n} C_j$.

For example, consider 5 jobs with release times and processing times described in the following table:

j	1	2	3	4	5
r_j	0	10	30	50	100
p_j	60	30	30	10	20

Then, the optimum schedule for the instance is illustrated in the following figure:



The total completion time of the 5 jobs is $40 + 60 + 80 + 120 + 150 = 450$.

Input:

- The input is taken from the standard input (console).
- The first line of input contains one integer n, the number of jobs.
- The next n lines give the description of the n jobs. Each line contains two integers r and p, denoting the release time and processing time of a job.

Output:

- The output is printed to the standard output (console). It contains a single line, which is the total completion time of the optimum schedule.

Example Input:	**Example Output**:
5 0 60 10 30 30 30 50 10 100 20	450

Constraints:

- $1 \leq n \leq 10^6$.
- The release times are integers in $[0, 10^6]$ and the processing times are integers in $[1, 10^6]$.
- It is expected that your program terminates in 10 seconds.

Problem 3. Given a set of n integer-valued points in a 2-D plane, find the closest pair of points with respect to the Manhattan distance. The Manhattan distance between two points (x, y) and (x', y') is defined as $|x - x'| + |y - y'|$.

Input:

- The input is taken from the standard input (console).
- The first line of input contains an integer n, representing the number of points in the set. The points are indexed from 1 to n.
- Each of the following n lines contains two integers x and y, representing the x- and y-coordinates of a point in the plane.

Output:

- The output is printed to the standard output (console).
- The first line contains the Manhattan distance of the two closest points, and the second line contains the indices of the points, separated by a space. If there are multiple pairs achieving the minimum Manhattan distance, you can output any pair.

Example Input:	**Example Output**:	This indicates that the closest pair of points with respect to the Manhattan distance is point 3 and point 5. They have Manhattan distance 2.
7 0 0 1 2 5 5 -2 -3 4 6 7 8 9 2	2 3 5	

Constraints:

- $1 \leq n \leq 10^6$.
- The coordinates of the points are integers in $[-10^9, 10^9]$.
- It is expected that your program terminates in 10 seconds.