CS3310 Final Project (300 Points) Spring 2020

Instructor: Tannaz R.Damavandi

Due Date: Friday – 05/08/2020 at 11:59 pm.

Task #1 – Sudoku Puzzle Solver

(100 Points)

Problem Definition:

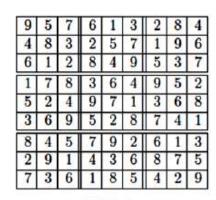
Sudoku is a number-placement problem in which a partially filled 9x9 grid is filled with digits from 1 to 9. The 9x9 grid itself consists of 9 sub-girds of size 3x3. The objective of this puzzle is to assign single digits from 1 to 9 in empty cells of this grid such that every row, every column and every sub-gird contains exactly one instance of the numbers between 1 to 9.

- Use backtracking technique to solve a given Sudoku Puzzle.
- Input: Your program must read from an *input.txt* file which contains a partially filled 9x9 gird of numbers. (To represent empty cells use 0 or -1).
- Output: Print the solved puzzle in format of a 9x9 grid.

Use the following Sudoku puzzle and its solution as your reference.

						2		
	8				7		9	
6		2				5		
	7			6				
			9		1			
				2			4	
		5		П		6		3
	9		4				7	
		6						

(a) Sudoku Puzzle



(b) Solution

- Use the Best-First search with Branch-and-Bound algorithm to solve 0/1 knapsack problem.
- Input: Positive integer *W* (knapsack capacity) and an *input.txt* file that contains the weights and profits of *n* objects. (The first line of this file should contain profits and the second line should include respective weights of n objects; use colon or space to separate values.)
- Output:
 - Visited node, profit, weight, bound, maximum profit and updated Priority queue at each step.
 - An integer maximum profit that is the sum of the profits of an optimal set and objects of that optimal set.

Note: you can use the example explained in the class as your reference.

Task #3 – Traveling Salesman Problem

(100 Points)

Problem Definition:

The goal of traveling salesman problem (TSP) is to find the shortest path in a directed graph that starts at a given vertex, visits each vertex exactly once, and ends back at the starting vertex.

Use the Best-First search with Branch-and-Bound algorithm to solve TSP.

- Input: An *nxn* adjacency matrix that shows the distances between n cities (vertices).
 - Your program must read from an input.txt file (use space to separate values.)
- Output: Optimal tour and its length.

What to Submit?

- 1. Java or Python source codes for each task (Please comment your code properly)
- 2. A detailed report on explaining the solutions and algorithms for each task.
- 4. Readme.txt (Please describe how to run your code).
- 5. Input.txt files of all tasks
- 6. For codes implemented in java, make sure to include .jar file.
- 7. Please zip all the documents as *yourname_FinalProject.zip* and submit it via the link provided on blackboard before or on its due date/time

IMPORTANT NOTE: LATE SUBMISSIONS WILL NOT BE ACCEPTED AND WILL RECEIVE ZERO.

Discussion among students is encouraged, but I expect each student to hand in original work.