Algorithm Design and Analysis

Assignment 04: Greedy Algorithms, Iterative Improvement

Total Points:	40
Assigned:	Tuesday, 31st October 2023
Due:	Saturday, 11th November 2023, 11:59pm.
Assignment Type:	Group assignment
Submit:	Latex for Written Work

Instructions

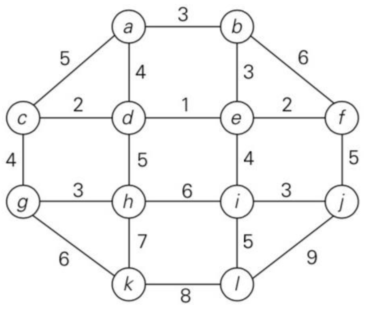
- You may complete and submit Assignment 4 in your study groups.
- Although you are working on it as a group, it is important for each person to fully participate in the discussions and understand the work done.

Preparation

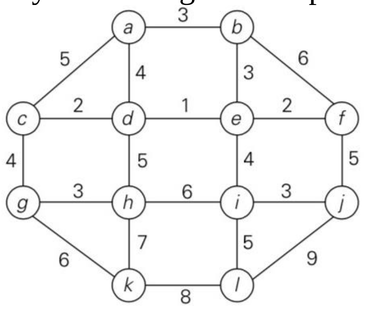
Study the relevant sections of Chapters 9 & 10 of the textbook by Levitin (on the Greedy and Iterative Improvement Paradigms respectively).

PART A: Basic Practice with Key Algorithms

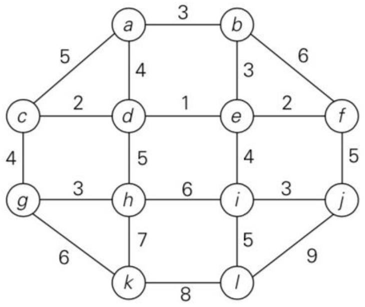
Problem 1 [3 points]: *Greedy Algorithms: Prim's Algorithm*
Execute Prim's algorithm to find the minimum spanning tree of the graph below. Break any ties among nodes alphabetically.



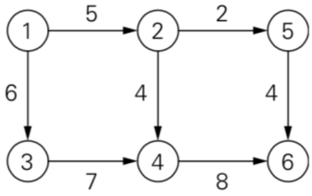
Problem 2 [3 points]: *Greedy Algorithms: Kruskal's Algorithm*
Execute Kruskal's algorithm to find the minimum spanning tree of the graph below. Break any ties among nodes alphabetically.



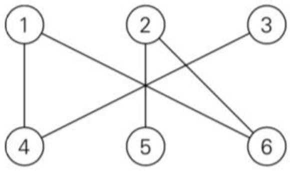
Problem 3 [3 points]: *Greedy Algorithms: Dijkstra's Algorithm*
Execute Dijkstra's algorithm on the graph below, using vertex ***a*** as the source vertex. Break any ties among nodes alphabetically.



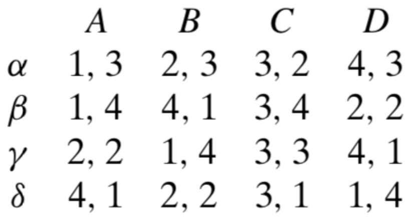
Problem 4 [3 points]: *Iterative Improvement: Maximum Flow*
Apply the shortest augmenting path method to find a maximum flow and minimum cut in the following network. Break any ties among nodes in numerical order.



Problem 5 [3 points]: *Iterative Improvement: Maximum Matching in a Bipartite Graph*
Find the maximum matching in this bipartite graph. Break any ties among nodes in numerical order.



Problem 6 [3 points]: *Iterative Improvement: Stable Marriage*
Find a stable marriage matching for the instance defined by the following ranking matrix:



	A	B	C	D
α	1, 3	2, 3	3, 2	4, 3
β	1, 4	4, 1	3, 4	2, 2
γ	2, 2	1, 4	3, 3	4, 1
δ	4, 1	2, 2	3, 1	1, 4

PART B: Understanding the Greedy & Iterative Improvement Paradigms

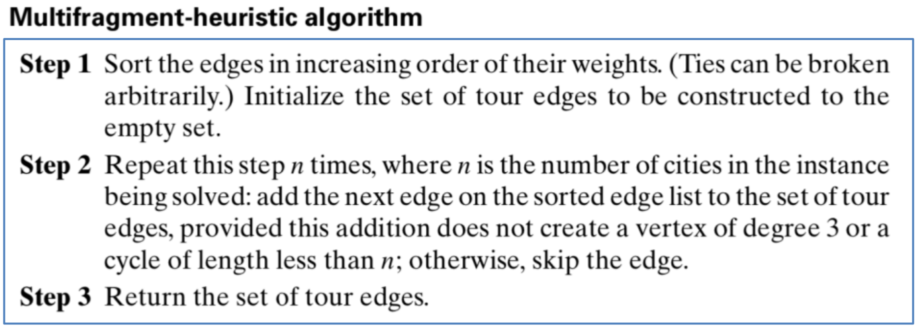
Problem 7 [6 points]: *Greedy versus Iterative Improvement*
Briefly explain the ***greedy*** paradigm and the ***iterative improvement*** paradigm, giving examples. What are the similarities and differences between them?

Problem 8 [3 points]: *Variation on the Maximum Flow Problem*
Explain how the maximum-flow problem for a network with several sources and sinks can be transformed into the same problem for a network with a single source and a single sink. Illustrate your proposed solution with an example. (That is, draw the original graph that has several sources and sinks, and then draw a new graph with only one source and sink that represents the transformed version of the original graph).

PART C: Greedy & Iterative Improvement Approximation Algorithms

As we know from our study so far, greedy algorithms and iterative improvement algorithms are both used to solve optimization problems. They each can find the optimal solution to *some* types of optimization problems, but not all. However, because of their efficiency and ease of implementation, greedy and iterative improvement approaches are often used as approximation algorithms for more difficult optimization problems, for example, the Traveling Salesman Problem.

Problem 9 [6 points]: *Greedy Approximation Algorithm for the TSP*
Background
The following algorithm, from Section 12.3 of the Levitin textbook, is an example of a greedy approximation algorithm for the travelling salesman problem:



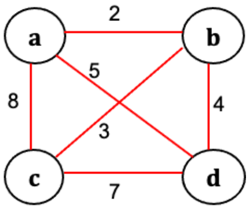
Multifragment-heuristic algorithm

Step 1 Sort the edges in increasing order of their weights. (Ties can be broken arbitrarily.) Initialize the set of tour edges to be constructed to the empty set.

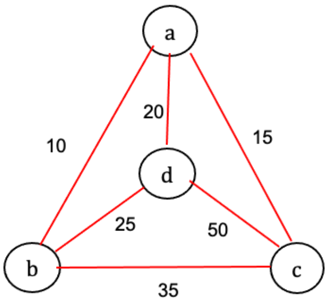
Step 2 Repeat this step n times, where n is the number of cities in the instance being solved: add the next edge on the sorted edge list to the set of tour edges, provided this addition does not create a vertex of degree 3 or a cycle of length less than n; otherwise, skip the edge.

Step 3 Return the set of tour edges.

Task
For each of the two instances of the traveling salesman problem given below, apply (i) the greedy multifragment-heuristic algorithm listed above, and (ii) the brute force approach, to solve the problem. In each case, compute the ratio between the value of the approximate solution found by the greedy approach and the optimal solution found by the brute force. This ratio is called the *accuracy ratio*.



(a)

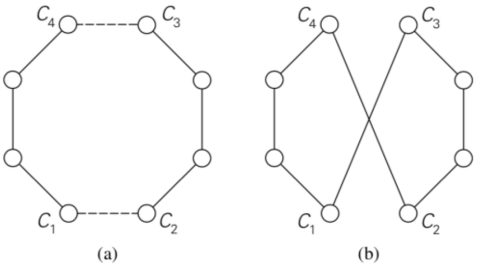


(b)

Problem 10 [2 points]: *Iterative Improvement Approximation Algorithms for the TSP*

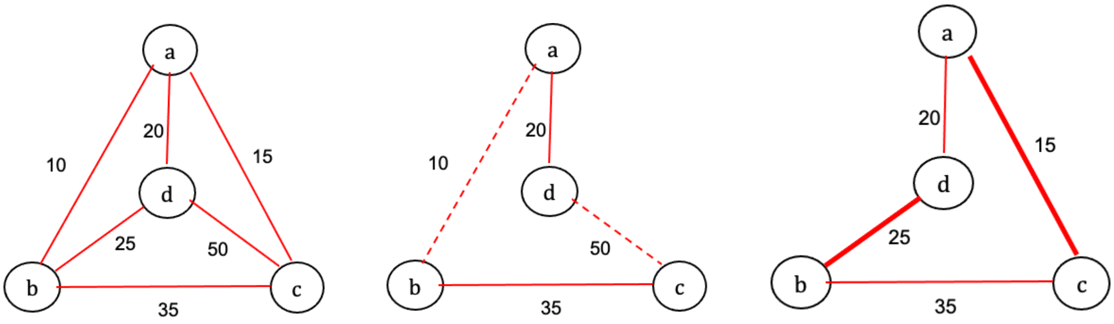
<u>*Background*</u>

There are several iterative improvement approximation algorithms, also called ***local search heuristics*** for the traveling salesman problem. A well-known one is the ***2-opt*** algorithm. The algorithm starts with some initial tour (e.g. constructed randomly or by some other heuristic). On each iteration, the 2-opt algorithm deletes a pair of nonadjacent edges in a tour and reconnects their endpoints by a different pair of edges in an attempt to find a better tour (see the figure below). This operation is called the ***2-change***. If the changes produce a shorter tour, the algorithm makes it the current tour and repeats the process. Otherwise, the current tour is returned as the algorithm's output and the algorithm stops.



2-change operation: (a) Original tour (b) New tour

The figure below illustrates a 2-change operation on a tour in a simple instance of the TSP.



Example of the 2-change operation on a specific graph.
(a) original graph for the traveling salesman problem
(b) example initial tour with total cost of 115
(c) modified tour (with cost of 95) after the 2-change operation in which the two dotted edges are deleted and the two bold edges are introduced

<u>*Task*</u>

Do some quick research and list other iterative improvement / local search heuristics for the Travelling Salesman Problem.