Questions and Exercises to work out and turn in:

Grading Guidelines:

* A right answer will get full credit when:

1. It is right (worth 25%)
2. It is right **AND** neatly presented making it easy and pleasant to read. (worth an **extra** 15%)
3. There is an **obvious and clear link** between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth an **extra** 60%).
4. Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.

You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, **personal** writing is expected.

* USE THIS FILE AS THE STARTING DOCUMENT YOU WILL TURN IN. **DO NOT DELETE ANYTHING FROM THIS FILE:** JUST **INSERT** YOUR ANSWERS.
* IF USING HAND WRITING (STRONGLY DISCOURAGED), **USE THIS FILE** BY CREATING SUFFICIENT SPACE AND WRITE IN YOUR ANSWERS.
* FAILING TO FOLLOW TURN IN DIRECTIONS /GUIDELINES WILL COST **A 30% PENALTY.**

**Objectives of this assignment**:

* to use and manipulate the concepts presented in this module
* to propose and write algorithms in pseudocode
* to analyze the time complexity of algorithms
* to analyze the space complexity of algorithms
* to learn autonomously new concepts

What you need to do:

Answer the questions and/or solve the exercises described below.

Exercise 1 (15 points)

Write pseudocode for MAKE-SET, FIND-SET, and UNION using the linked-list

representation and the weighted-union heuristic. Make sure to specify the attributes

that you assume for set objects and list objects.

Set objects have a head attribute that points to the first object in the list, which is also the representative, a tail attribute that points to the last object in the list, and a length attribute that holds the number of items in the list. List objects each have a member attribute, which is the data it is holding, a next attribute which points to the next object in the list, and a set object attribute which points back to the set object.

MAKE-SET(x)

1. set = new SetObject()
2. list = new LinkedList<ListObject>()
3. list.add(x)
4. setObject.head = x
5. setObject.tail = x
6. x.next = null
7. x.setObject = set

FIND-SET(x)

1. return x.setObject.head

UNION(x, y)

1. if x.setObject.length > y.setObject.length
2. x.setObject.tail.next = y.setObject.head
3. x.setObject.tail = y.setObject.tail
4. for each ListObject listObject in y.setObject
5. listObject.setObject = x.setObject
6. y.setObject.head = null
7. y.setObject.tail = null
8. else
9. y.setObject.tail.next = x.setObject.head
10. y.setObject.tail = x.setObject.tail
11. for each ListObject listObject in x.setObject
12. listObject.setObject = y.setObject
13. x.setObject.head = null
14. x.setObject.tail = null

Exercise 2 (15 points)

Show the data structure that results and the answers returned by the FIND-SET

operations in the following program. Use the linked-list representation with the

weighted-union heuristic.

1 for i = 1 to 16

2 MAKE-SET(xi)

3 for i = 1 to 15 by 2

4 UNION(xi, xi+1)

5 for i = 1 to 13 by 4

6 UNION(xi, xi+2)

7 UNION(x1,x5)

8 UNION(x11, x13)

9 UNION(x1; x10)

10 FIND-SET(x2)

11 FIND-SET(x9)

Exercise 3 (20 points) non recursive version of FIND-SET

Write a non recursive version of FIND-SET with path compression.

Exercise 4 (20 points)

Let an edge (p,q) that has the smallest (minimum) weight in a connected graph G=(V,E,w) where w is the weight function. Show (Prove)that the edge (p,q) belongs to some minimum spanning tree of G. (Hint: inspire yourself from the proof of Theorem 23.1. See Figure 23.3)

Exercise 4 (15 points) Kruskal’s Algorithm

Consider this graph G=(V, E, w) provided as an adjacency-matrix. V=(H, G, F, E, D, C, B, A)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | H | G | F | E | D | C | B | A |
| H |  |  |  |  |  |  |  | 4 |
| G |  |  | 8 |  |  | 5 |  |  |
| F |  | 8 |  | 6 |  | 1 |  |  |
| E |  |  | 6 |  | 3 |  |  |  |
| D |  |  |  | 3 |  | 9 |  |  |
| C |  | 5 | 1 |  | 9 |  | 7 |  |
| B |  |  |  |  |  | 7 |  | 2 |
| A | 4 |  |  |  |  |  | 2 |  |

1. Draw this graph
2. Trace Kruskal’s algorithm and show step by the step the construction of the minimum spanning tree.

Exercise 5 (15 points) Prim’s Algorithm

Consider this graph G=(V, E, w) provided as an adjacency-matrix. V=(H, G, F, E, D, C, B, A)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | H | G | F | E | D | C | B | A |
| H |  |  |  |  |  |  |  | 4 |
| G |  |  | 8 |  |  | 5 |  |  |
| F |  | 8 |  | 6 |  | 1 |  |  |
| E |  |  | 6 |  | 3 |  |  |  |
| D |  |  |  | 3 |  | 9 |  |  |
| C |  | 5 | 1 |  | 9 |  | 7 |  |
| B |  |  |  |  |  | 7 |  | 2 |
| A | 4 |  |  |  |  |  | 2 |  |

1. Draw this graph (It is the same as the previous question. Copy/Paste would be just fine).
2. Trace Prim’s algorithm starting from Vertex F and show step by the step the construction of the minimum spanning tree.
3. Compare the minimum spanning trees obtained by Kruskal’s and Prim’s algorithms, respectively.

**What you need to turn in:**

* Electronic copy of this file (including your answers) (standalone). Submit the file as a Microsoft Word or PDF file.
* Recall that answers must be well written, documented, justified, and presented to get full credit.
* How this assignment will be graded:
* A right answer will get full credit when:
* It is right (worth 25%)
* It is right AND neatly presented making it easy and pleasant to read. (worth 15%)
* There is an obvious and clear link between 1) the information provided in the exercise and in class and 2) the final answer. A clear link is built by properly writing, justifying, and documenting an answer (worth 60%).
* Calculation mistakes will be minimally penalized (2 to 5% of full credit) while errors on units will be more heavily penalized.
* You are welcome/encouraged to discuss exercises with other students or the instructor. But, ultimately, personal writing is expected.