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** EACH ANSWER **RIGHT AFTER ITS QUESTION/PROMPT**.
* 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 (50 points) Kruskal’s Algorithm

Consider this graph G=(V, E, w) provided as an adjacency-matrix. V = (r, s, t, u, v, w, x, y)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | r | s | t | u | v | w | x | y |
| r |  | 13 | 17 |  |  |  |  | 9 |
| s | 13 |  |  | 16 |  |  |  |  |
| t | 17 |  |  | 19 |  |  |  |  |
| u |  | 16 | 19 |  | 21 | 26 |  |  |
| v |  |  |  | 21 |  | 17 | 13 |  |
| w |  |  |  | 26 | 17 |  | 17 |  |
| x |  |  |  |  | 13 | 17 |  | 7 |
| y | 9 |  |  |  |  |  | 7 |  |

1. (5 points) Draw this graph. If needed, you can draw by hand, take a picture and insert it. Just make sure the drawing is neat and pleasant (neatness is worth 15%)

A diagram of a triangle with circles and lines

Description automatically generated

1. (45 points) Trace **Kruskal’s** algorithm and **show step by the step** the construction of the minimum spanning tree. **Draw** the MST each time you add an edge. **Highlight** the latest added edge with its weight.

We start with a set of edges A, building by adding one edge at a time. We set A to an empty set. After that we create small sets of the vertices of the graph. We will then take all edges of G andsort them in non-decreasing order. For each edge, we look at the endpoint of the edges. If they belong to different sets, we will add that edge to A and we will make a union of the two sets to which u and v belong.

Edges: non-decreasing order:

(x, y) 7

(r, y) 9

(r, s)13

(v, x) 13

(s, u) 16

(r, t) 17

(v, w) 17

(w, x) 17

(t, u) 19

(u, v) 21

(u, w) 26

A diagram of a triangle with white circles and black lines

Description automatically generated A diagram of a triangle with white circles and black lines

Description automatically generated  
The first edge in non-decreasing order is with (r, y) is the next smallest edge. We look to see if  
weight of 7 is (x, y). We will at this edge to A and r and y belong to the same set. They do not so we  
we will create a union of {x, y}. After this, look to add to A and make a union. The new set is  
the next smallest edge. {r, y, x}. After this, we look to the next smallest edge.

A diagram of a triangle with white circles and black lines

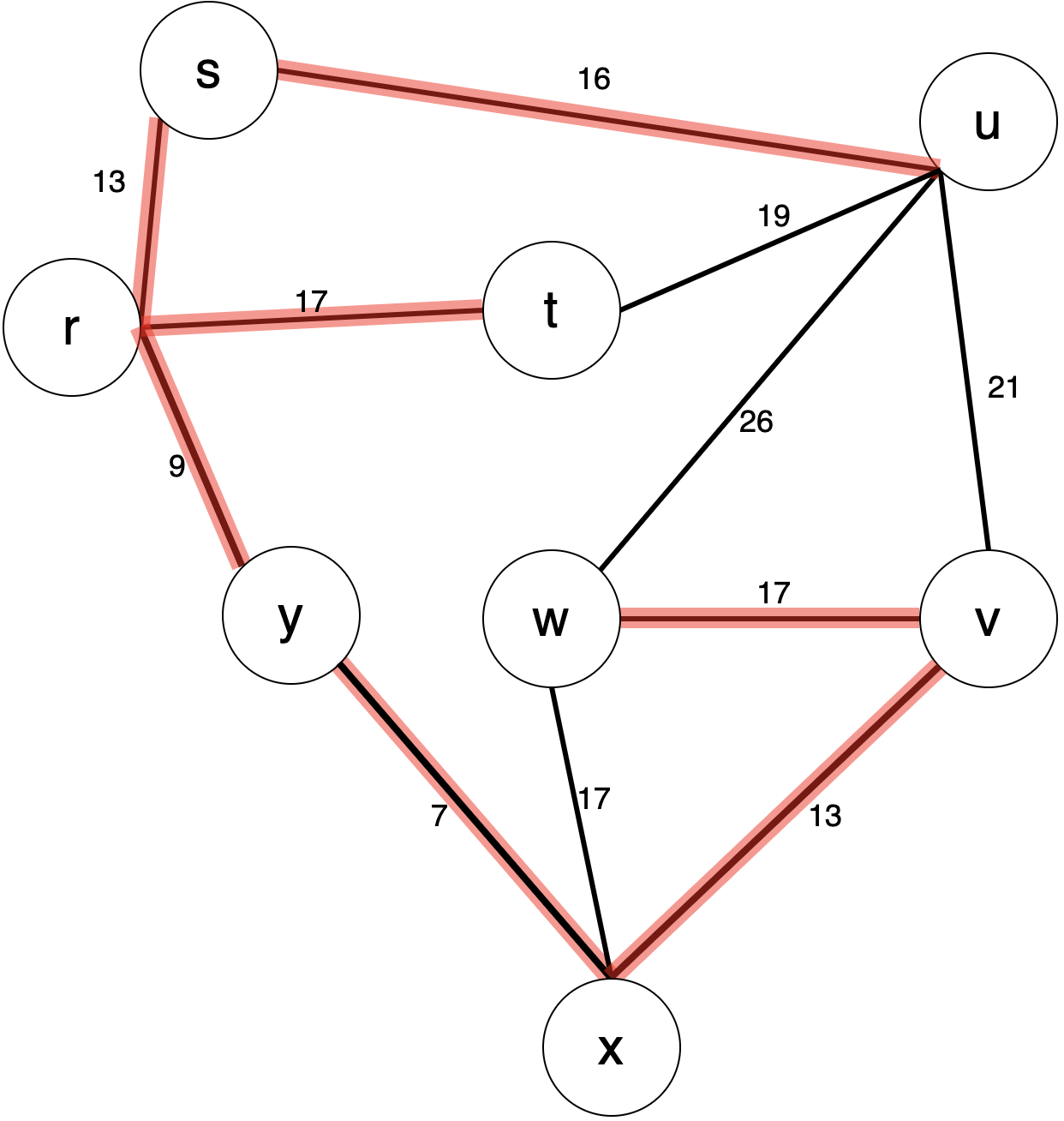
Description automatically generated A diagram of a triangle with white circles and black lines

Description automatically generated  
(r,s) is the next smallest edge. They do not belong (v, x) is the next edge. We look to see if v and x  
to the same set so we add to A and make a union. belong to the same set. They do not so we add to  
The new set is {s, r, y x}. After this, we look to A and make a union. The new set is {s, r, y, x, v}.

the next smallest edge. After this, we look to the next smallest edge.

A diagram of a triangle with white circles and black lines with Silverstone Circuit in the background

Description automatically generated A diagram of a triangle with white circles and black lines with Silverstone Circuit in the background

Description automatically generated  
(s, u) is the next edge. We look to see if s and u (r, t) is the next edge. We look to see if r and t  
belong to the same set. They do not so we add to belong to the same set. They do not so we add to A  
A and make a union. The new set is {u, s, r, y, x, v} and make a union. The new set is {r, t}. After this, we  
After this, we look to the next smallest edge. look to the next smallest edge.  
  
  
(v, w) is the next edge. We look to see if v and w (w, x) is the next edge. We look to see if w and x   
belong to the same set. They do no so we add to A belong to the same set. They do. Both belong  
and make a union. The new set is {u, s, r, y, x, v, w} to set {u, s, r, y, x, v, w} therefore we leave this  
 edge out. We then look to the next smallest edge.

(t, u) is the next edge. We look to see if t and u (u, v) is the next edge. We look to see if u and v

belong to the same set. They do. They both belong belong to the same set. They do. They both belong  
to the same set therefore we leave this edge out. to the same set therefore we leave this edge out.  
We then look to the next smallest edge. We then look to the next smallest edge.  
  
  
(u, w) is the next edge. We look to see if u and w  
belong to the same set. They do. They both belong   
to the same set therefore we leave this edge out.   
A is now empty we are left with the resulting graph.

Exercise 2 (50 points) Prim’s Algorithm

Consider this graph G=(V, E, w) provided as an adjacency-matrix. V = (r, s, t, u, v, w, x, y)

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | r | s | t | u | v | w | x | y |
| r |  | 13 | 17 |  |  |  |  | 9 |
| s | 13 |  |  | 16 |  |  |  |  |
| t | 17 |  |  | 19 |  |  |  |  |
| u |  | 16 | 19 |  | 21 | 26 |  |  |
| v |  |  |  | 21 |  | 17 | 13 |  |
| w |  |  |  | 26 | 17 |  | 17 |  |
| x |  |  |  |  | 13 | 17 |  | 7 |
| y | 9 |  |  |  |  |  | 7 |  |

1. Draw this graph (It is the same as the previous question. Copy/Paste would be just fine

A diagram of a triangle with circles and lines

Description automatically generated

1. (45 points) Trace **Prim’s** algorithm starting from Vertex and **show step by the step** the construction of the minimum spanning tree. **Draw** the MST each time you add an edge. **Highlight** the latest added edge with each weight.

We start with an empty set A and slowly add edges to it. The way this is done is to start at a specific vertex, for this we pick u. A queue is generated with ∞ in the key position for everything except vertex u, u gets a 0 since it is where we start. All vertices get a NIL for the parent position at this point. Then at each step we find the edge with the lowest weight that connects a vertex in the MST to a vertex outside of it and add that edge to the MST. We repeat this until all vertices are included in the MST. During this the queue is continually updated if the edge connecting the two vertex is less than the previous parent and key. We also keep up with all the edges added to make sure there are no cycles that get formed.  
Edges: non-decreasing order:

(x, y) 7

(r, y) 9

(r, s)13

(v, x) 13

(s, u) 16

(r, t) 17

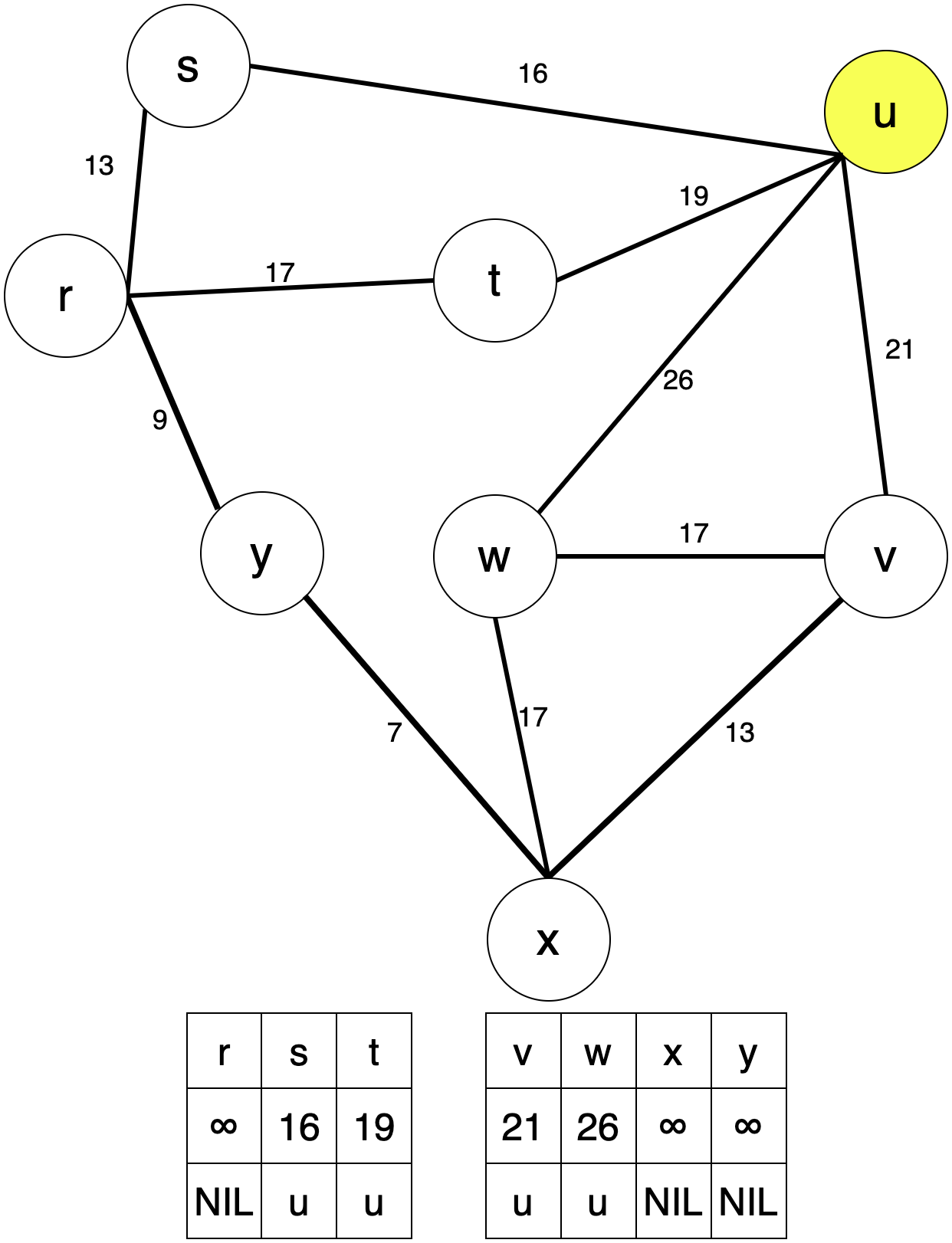
(v, w) 17

(w, x) 17

(t, u) 19

(u, v) 21

(u, w) 26

 A screenshot of a game

Description automatically generated

We start with the vertex u dequeuing it which is After observing the edges we find that the edge  
observed by the deletion of the letter and fill in the {s,u} is the one with the least weight so we add  
queue information for all of the edges that connect it to A. We then dequeue s which means we delete   
u with another vertex outside of the MST it and then add the key and parent information for s  
 to the queue if it is less than the current keys. In this  
 case that would only be the edge to r.

A screenshot of a puzzle

Description automatically generated A screenshot of a puzzle

Description automatically generated  
Since the the edge {r,s} is the least weight we add The edge with the least weight is {r,y} so this gets  
that edge to A, dequeue r, and update the parent added to A, y is dequeued and the parent and key  
and key information for vertex with edges that are information for the vertex outside of the MST that  
less weight than current. That would be the weigh less than their current values. The only one  
information for vertex t and y. applicable is x.

A screenshot of a puzzle

Description automatically generated A screenshot of a diagram

Description automatically generated  
The next lowest weight edge in the queue is {x,y} With the queue getting smaller our next edge is {x,v}  
so as before we deque x, update the parent and with a weight of 13 it is the smallest weight available.  
key information for vertex with edges weigh less We add this edge to A, queue v and update any parents  
than their current weight. For x this is both w and v. and keys in which the weight is less. There are no edges  
 that v is the parent of that weigh less than the ones in   
 the queue.

A diagram of a triangle with yellow circles and lines

Description automatically generated A screenshot of a diagram

Description automatically generated  
Since the last two edges in the queue have the The last edge in the queue is {w,v}, this edge is   
same weight we arbitrarily pick the left one which added to A and since the queue is now empty   
is {r,t}. This edge is added to A, and since there are the resulting graph is the fully formed MST  
no other vertices that are not already in the MST  
there is nothing further to update while t is selected.

1. (5 points) **Compare** the minimum spanning trees obtained by Kruskal’s and Prim’s algorithms, respectively.

The resulting minimum spanning trees obtained by Kruskal’s and Prim’s algorithms are both the same despite the different approach. Prim’s algorithm approaches it by starting from a specified vertex, Vertex u, and extends the tree by adding the cheapest outgoing edge. Whereas Kruskal’s algorithm sorts by the weight in non-decreasing order and then adds the cheapest edge that joins disjoint components. They both accomplishes the goal of providing a solution that results in the minimum weight connection between the vertices.

**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.

**Appendix**: Grading: What is an OBVIOUS and CLEAR LINK?

Here is an example to explain what an **obvious and clear link** is and how we grade your work.

Consider the following problem:

"(100 points) John travels from Auburn to Atlanta in his car at a speed of 60 mph. Leaving at 8am, at what time will John reach Atlanta".

Here are the answers of three students and their scores:

* **Student 1** answers: "9:48am". Student 1 will get 25 points.
* **Student 2**answers : "John will reach Atlanta at 9:48am". Student 2 will get 25+15 = 40 points
* **Student 3** answers: "The time t to travel a distance d at speed v is equal to d/v = d/60mph. The problem does not provide the distance d from Auburn to Atlanta. Based on GoogleMaps, the distance from Auburn to Atlanta is approximately 108 miles (**document is attached**).



Therefore, the time t = 108 miles/60mph \* 60 minutes/hour= 108 minutes. Since John left at 8am, he will then reach Atlanta at 8am + 108 minutes = 8 am + 60 minutes + 48 minutes = 9:48".

**Student 3** will get 25 + 15 + 60 = 100 points

Do you see the **direct** **link** going from the data provided in the question to the final answer, using general knowledge/formula and documents?.... Can you now solve the following problem and get 100 points?

"(100 points) Alice travels from Auburn to Atlanta in her car at a speed of 60 mph. Leaving at 8am, at what time will Alice reach Atlanta assuming that she had a flat tire that delayed her 30 minutes".