CS 260 - Assignment 13

Professor Mark W. Boady

Week 8

1 Introduction

Content by Mark Boady, Drexel University
This worksheet is worth 100 points.
You may complete this worksheet using any of the following methods.

- Print out the PDF. Complete by hand. Scan the file.
- Write directly on the PDF with editing software
- Write the Answer in MS Word, Notepad, etc. You **do not** have to rewrite the questions. Just clearly label each answer in your file.

2 Assignment 13

Question 1 : 13 points

A **heap** is a special kind of tree. The minimum value is always stored at the root of the tree. This tree is specifically designed to make finding the minimum as fast as possible.

A **heap** is normally stored in an array. The array has the following properties.

- The Root is at index 0
- The Parent of the node at index n is at index floor $((n-1)/2)$
- The Left Child of the node at index n is at $(n+1)*2-1$
- The Right Child of the node at index n is at index $(n+1)*2$

(a) (1 point) What index is the parent of the value at index 7?

(b) (1 point) What index is the parent of the value at index 8?

(c) (2 points) What index is the left child of the value at index 4?

(d) (2 points) What index is the right child of the value at index 4?

(e) (7 points) Draw the tree representation of the below array.

Index	0	1	2	3	4	5	6
Value	4	9	5	12	10	6	8

Question 2 : 6 points

When inserting a new element into the heap, we follow a two step process.

1. Insert the new element at the end of the array
2. upheap the element

The upheap algorithm is described below.

```
function UPHEAP(Heap H, Index i)
    if parent(i) < 0 then
        return
    end if
    p = H[parent(i)]
    if  p ≤ H[i] then
        return
    end if
    swap(i,parent(i))
    upheap(H,parent(i))
end function
```

The array from the previous question is given below.

A new value 2 has been inserted at the end of the array.

Index	0	1	2	3	4	5	6	7
Value	4	9	5	12	10	6	8	2

(a) (2 points) Show the array after the first swap.

(b) (2 points) Show the array after the second swap.

(c) (2 points) Show the array after the third swap.

Question 3 : 12 points

(a) (2 points) Starting with an empty array, insert 6 as the first element in the array.
Draw the array after inserting the value.

(b) (2 points) Insert 9 into the heap
Draw the array after completing all upheaps.

(c) (2 points) Insert 4 into the heap
Draw the array after completing all upheaps.

(d) (2 points) Insert 8 into the heap
Draw the array after completing all upheaps.

(e) (2 points) Insert 7 into the heap
Draw the array after completing all upheaps.

(f) (2 points) Insert 2 into the heap
Draw the array after completing all upheaps.

Question 4 : 4 points

When we want to remove the minimum element, we also complete a 2 step process.

1. Swap the minimum with the last item in the array, then delete
2. Downheap starting at the root

The downheap algorithm is described below.

```
function DOWNHEAP(Heap H, Index i)
    if children do not exist then
        return
    end if
    if  H[i] ≤ all children then
        return
    end if
    m = index of child with minimum value
    swap(i,m)
    downheap(H,m)
end function
```

The array from the previous question is given below.

We want to delete 4 from the below array

Index	0	1	2	3	4	5	6
Value	4	9	5	12	10	6	8

First, we swap with index 6 (the last index).

Index	0	1	2	3	4	5	6
Value	8	9	5	12	10	6	4

Next, we delete the last index.

Index	0	1	2	3	4	5
Value	8	9	5	12	10	6

Lastly, we downheap.

(a) (2 points) Show the array after the first downheap swap.

(b) (2 points) Show the array after the second downheap swap.

Question 5 : 12 points

(a) (2 points) Delete 2 from the below array.

Index	0	1	2	3	4	5
Value	2	7	4	9	8	6

Draw the array after deletion and completing all downheap swaps.

(b) (2 points) Delete 4 from the array from the previous part.
Draw the array after deletion and completing all downheap swaps.

(c) (2 points) Delete 6 from the array from the previous part.
Draw the array after deletion and completing all downheap swaps.

(d) (2 points) Delete 7 from the array from the previous part.
Draw the array after deletion and completing all downheap swaps.

(e) (2 points) Delete 8 from the array from the previous part.
Draw the array after deletion and completing all downheap swaps.

(f) (2 points) What is the last value left in the Array?

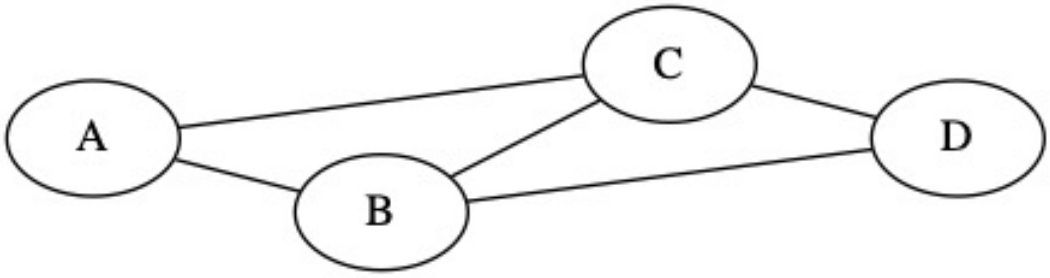
Question 6 : 9 points

A **graph** is a data structure made from nodes and edges. A **graph** is similar to a tree, but more general. It may contain cycles. All **trees** are **graphs** but all **graphs** are not **trees**

An **undirected graph** is a graph were all edges go both ways.

A graph is made of nodes (values) and edges (relationships between values).

Below is an example undirected graph.



(a) (2 points) A **node** is a value in the graph. What are the 4 nodes in this graph?

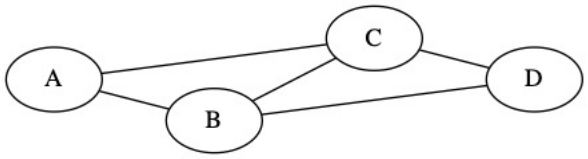
(b) (3 points) An **edge** is a relationship between two nodes in the graph. There is an edge connecting A to B. That means there is a relationship between them. We write this edge as (A,B).
What are all the edges in this graph?

(c) (2 points) A **path** is a sequence of nodes that can be used to travel between two nodes. This is written as the sequence of nodes traveled through. We can travel from B to D by the path (B,C,D) or the path (B,D).
Give 2 different paths to get from A to D.

(d) (2 points) A **cycle** is a path that starts and ends at the same point. For example, (A,B,C,A) is a cycle.
Give a cycle starting a node D.

Question 7 : 9 points

One way to represent a graph is an **Adjacency Matrix**. With an undirected graph, we put a 1 in the matrix if the edge exists and a 0 if it does not.

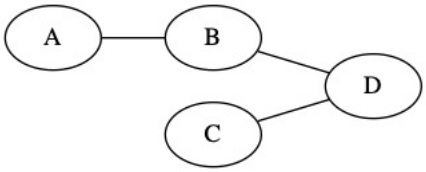


The Adjacency Matrix for this graph is

	A	B	C	D
A	1	1	1	0
B	1	1	1	1
C	1	1	1	1
D	0	1	1	1

(a) (1 point) Why does the diagonal of the matrix contain 1s?

(b) (4 points) Draw the Adjacency Matrix for the below graph.



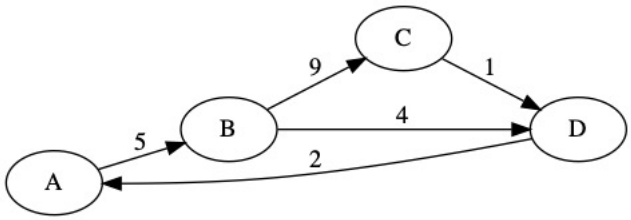
(c) (4 points) Draw the Graph for the following Adjacency Matrix.

	A	B	C	D
A	1	0	1	1
B	0	1	0	1
C	1	0	1	0
D	1	1	0	1

Question 8 : 12 points

A second type of graph is the **directed graph**. In this graph, relationships can be one-directional.

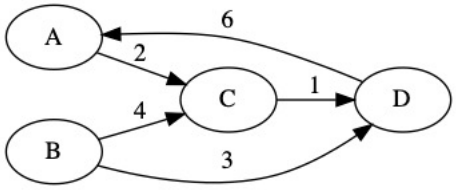
A graph can be **weighted**. The weight $w(x, y)$ is the **cost** to travel from x to y.



In a **weighted directed graph** the adjacency matrix stores the weight of the edge. We use ∞ to mean *no edge exists*. The path from a node to itself (no moving) it generally considered 0 (free).

	A	B	C	D
A	0	5	∞	∞
B	∞	0	9	4
C	∞	∞	0	1
D	2	∞	∞	0

(a) (8 points) Write the Adjacency Matrix of the below graph.



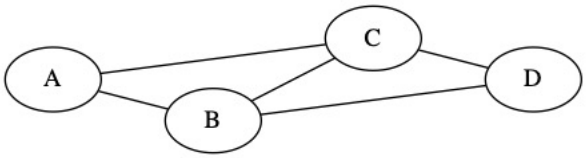
(b) (4 points) Draw the graph represented by the below matrix.

	A	B	C	D
A	0	2	∞	∞
B	∞	0	∞	6
C	1	∞	0	1
D	5	4	∞	0

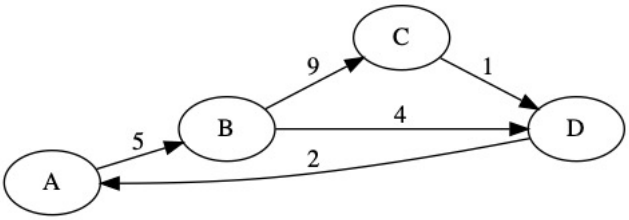
Question 9 : 10 points

The **Adjacency Matrix** has a problem. It takes up a lot of space. If we have n nodes, we need an $n * n$ matrix. The **Adjacency List** stores the edges in an Array of Linked Lists. This uses less memory, but makes it harder to find edges.

Here are two examples.

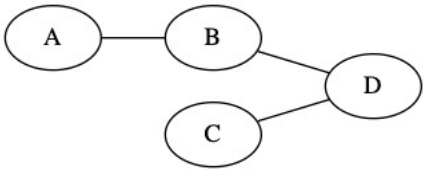


A	B,C
B	A,C,D
C	A,B,D
D	B,C

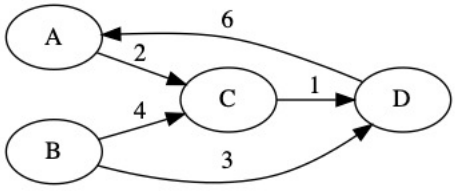


A	(B,5)
B	(C,9), (D,4)
C	(D,1)
D	(A,2)

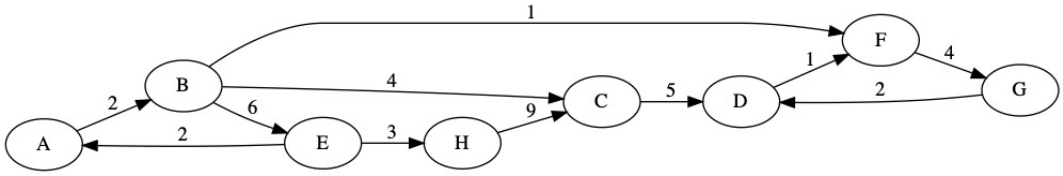
(a) (5 points) Write the Adjacency List of the following Graph.



(b) (5 points) Write the Adjacency List of the following Graph.



Question 10 : 13 points



The **shortest path** between two nodes is the **path** with the lowest weight.

For example, the shortest path from A to C has a total weight of 6 and takes the path (A,B,C).

(a) (3 points) What is the shortest path from A to F? Give the path and its weight.

(b) (4 points) What is the shortest path from A to D? Give the path and its weight.

(c) (6 points) A **Strongly Connected Component** is a set of nodes such that every node can be reached by every other node. If you pick any node in the SCC, there exists a path that starts at the node, goes in a cycle through all other nodes in the SCC, and ends at the starting point.

The graph has two SCC. What are the two subsets of nodes that makes SCC.