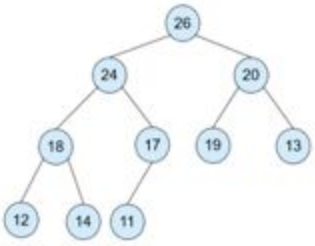
Jake Tantorski
CMPT435

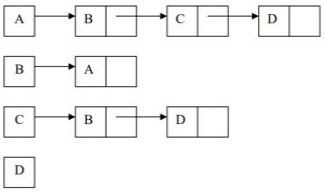
Assignment 8

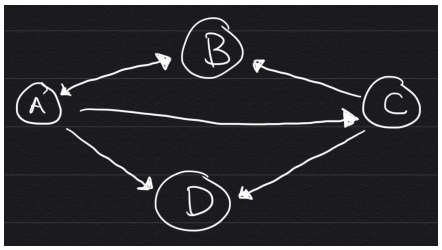
1. Max-Heap



Index	0	1	2	3	4	5	6	7	8	9
Node	26	24	20	18	17	19	13	12	14	11

2. HeapSort sorts {3, 1, 6, 5, 2, 4}
    a. Start with 3
    b. We add 1 to our heap
    c. 1 < 3, switch their places
    d. Add 6 to the heap to the right of 3 under 1
    e. Add 5 to the heap under and to the left of 3 since 5 > 3 & 5 > 1
    f. Add 2 to the heap since 2 > 1 and 2 > 3, put it in the place of 3
    g. Move 3 under 2 to the right of 5
    h. Add 4 to the heap, since 4 > 1 and 4 < 6
    i. Move 4 to where 6 was
    j. Move 6 underneath 4
3. Picture of the directed graph that has the below adjacency list representation





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

4.

I. The getAdjVert finds a vertex that is connected to the node and checks to make sure that it is not visited. From there, the dfscycle function loops through the nodes in the heap and finds nodes marked with 0 in the stack, then in the detect cycle function, it uses a boolean to see if a cycle is present.

III. The time complexity is $O(|V|)*O(|V|+|E|) = O(|V|^2+|E||V|)$ time

II. Pseudocode

```
getAdjVert(vertex)
        for ( i = 0; i < nVerts; i++)
                if (adjMat[vertex][i] == 1 && vertexcycleflag[i]!=1) {
                        return i;
        end for
        return -1;

dfscycle(vertex)
                for(i = 0; i <vertexcycleflag.length; i++)
                        vertexcycleflag[i] = -1;
                end for
                theStack.clear();
```

```
                vertexcycleflag[vertex] = 0;
                displayVertex(vertex);
                theStack.push(vertex);


                while (!theStack.isEmpty())
                        int nextTo = getAdjVert(theStack.peek());
                        if (nextTo == -1)
                                int b = theStack.pop();
                                vertexcycleflag[b] = 1;
                        end if
                        else if (vertexcycleflag[nextTo] == 0)
                                return true;
                        end else if
                        else
                                vertexcycleflag[nextTo]= 0;
                                displayVertex(nextTo);
                                theStack.push(nextTo);
                        end else
                end while
                return false;



detectcycle()
        boolean flag = false;
        for ( i = 0; i <vertexcycleflag.length; i++)
                if (dfscycle(i))
                        flag = true;
                        break;
                end if
        end for
        println(" Does the graph contain a cycle? "+ flag);
```