WA7 - Assignment Group 16

Sravani Kirla - skirla
Matthew Heil - heilx069
Terrence Hanrahan - hanra028

1.

```
def isUnivSink(adjMatrix):
  x = 0
  y = 1
  sink = [0]*len(adjMatrix)
  # // Eliminates the n-1 vertices which has out-degree of 1 and in-degree of 0
  while (x<len(adjMatrix) and y<len(adjMatrix)):
    print("values of X: ",x," and Y: ",y)
    if adjMatrix[x][y] == 1:
      sink[x] = 1
      x = x + 1
      if x == y:
        x = x + 1
    elif adjMatrix[x][y] == 0:
      sink[y] = 1
      y = y + 1
      if x == y:
        y = y + 1
  # // get the non-eliminated vertex
  print(sink)
  if 0 in sink:
    univSink = sink.index(0)
  else:
    return False

  row = 0
  column = 0
  # // Verify if a non-eliminated vertex is a universal sink
  for i in range(len(adjMatrix)):
    print("Value of i: ",i)
    row = row + adjMatrix[univSink][i]
    column = column + adjMatrix[i][univSink]

  if row==0 and column == len(adjMatrix)-1:
    return True
  else:
    return False
```

The loops in the algorithm can only run a maximum of 2V times. The first while loop is the one with a maximum bound of 2(V-1) + 1. This is because the worst case scenario for it is that both x and y have to reach V - 1, then the next iteration ends the loop. This gives a bound of O(2(V-1)+1) which still is O(V). The second for loop only runs up to V-1 before ending. This gives a bound of O(V-1), or O(V). Adding these two bounds together gives O(2V), which still is O(V). Hence, the algorithm runs in O(V) time.

2.

WA7 - Assignment Group 16

Assume that “n” is the number of wrestlers and “r” is the rivalry pairs of wrestlers. If we use BFS algorithm version which loops through all of the vertices then we can assign “babyface” or “heel” to all of the vertices for a given graph.

Start the algorithm by assigning a “Babyface” to the start node which is d=0 will be a “babyface” and assign all of it’s adjacent vertices which is d=1 to “heels” such that we will avoid assigning the pairs to the same type of wrestlers. Then we keep moving from d=1 vertices to d=2 vertices and assign them “babyfaces” and d=4 to “heels” if we keep assigning these to the alternative depths we can finish reaching the last vertex by not assigning any of two adjacent vertices of same type.

If we calculate the runtime for this, we need to consider the breadth first search algorithm runtime as we are using BFS to visit and explore adjacent vertices we will end up having a O(V+E) run time and even if we consider assigning “babyface” or a “heel” it will be the same runtime as BFS as this is a similar step as of assigning a color to any of the vertices which takes O(E) times in the worst-case. For this scenario, as the number of vertices is “n” and number of edges is “r” and our runtime will be O(n+r) in the worst-case based on the same notation of O(n+r).

3. Discovery Time on Left, Finishing Time on Right

A → d =1, f = 12

B → d = 2, f =3

C → d = 4, f = 11

D → d = 13, f = 14

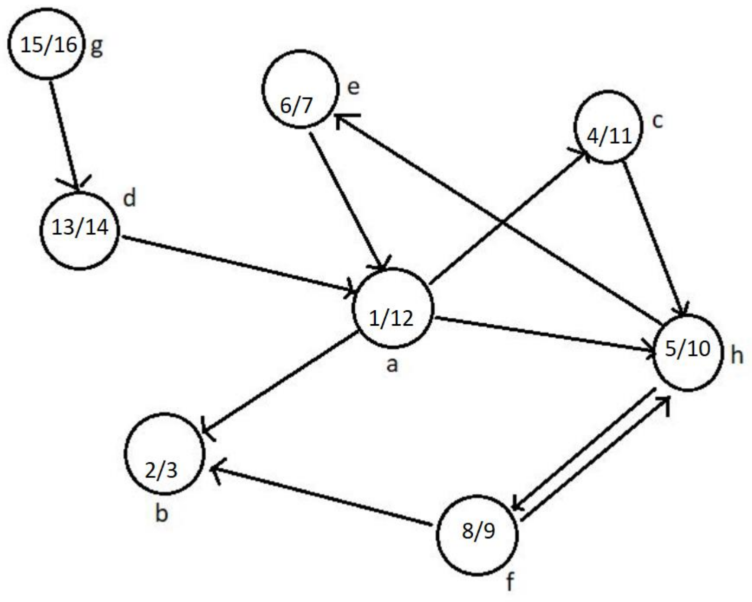
E → d = 6, f =7

F → d = 8, f = 9

G → d = 15, f = 16

H → d = 5, f = 10

WA7 - Assignment Group 16


15/16
g
6/7
e
4/11
c
13/14
d
1/12
a
5/10
h
2/3
b
8/9
f