Emily Crilley

Analysis of Algorithms - Homework 4

1. See hw4.py
2. See hw4.py
3. See hw4.py
4. Solve the Recurrence using iteration:

$T(1) = 2 \qquad T(n) = (n + 1) + \frac{1}{n}\sum_{q=1}^{n-1} T(q)$

$nT(n) = n(n+1) + \sum_{q=1}^{n-1} T(q)$

$nT(n) - (n-1)T(n-1) =$

$n(n+1) + \sum_{q=1}^{n-1} T(q) - n(n-1) - \sum_{q=0}^{n-2} T(q) =$

$n(n+1) - n(n-1) + 2T(n-1) =$

$2n + T(n-1)$

$nT(n) = 2n + (n+1) + T(n-1)$

$\frac{T(n)}{n+1} = \frac{1}{n+1} + \frac{T(n-1)}{n}$

$S(n) = \frac{T(n)}{n+1}$

$S(1) = 1$

$S(n) = \frac{1}{n+1} + S(n-1)$

$S(n) = \frac{1}{n+1} + \frac{1}{n} + S(n-2)$

$S(n) = \frac{1}{n+1} + \ldots + \frac{1}{n-(k-2)} + S(n-k)$

$k = n - 1$

$S(n) = (\frac{1}{n+1} + \ldots + \frac{1}{3}) + S(1)$

$S(n) = (H_{n+1} - 3/2) + 1$

$T(n) = (n+1)(H_{n+1} - 1)$

5.

a. It would take O(V) time complexity for the computation of the out-degree of every vertex.
It would take $O(V) + \sum_{i=1}^{n} O(E) = O(V + E)$ for the computation of in-degrees

b. The expected time complexity would be O(1) as the average time of a lookup on a hash table is O(1). This scheme's disadvantage is the worst case time of O(V). A binary search tree could be a good alternate data structure as the worst case complexity is better than that of a hash table, which is O(log(V)). The one

disadvantage of this data structure is the best case complexity is not as optimal as that of a hash table.

6.

a.

Vertex	u	r	s	t	v	w	x	y
d	0	4	3	1	5	2	1	1
π	--	s	w	u	r	t	u	u

b. Line 18 can be removed as the black nodes would just end up being dequeued within the while loop. This action makes the changing of the color redundant as the node would immediately be dequeued. The same result can be expected with the removal of line 18.

c. To start, we can assume we will end with $d=\delta(s,v)$. Because these values equal, we can change the adjacency lists without having to worry about changing it in multiple places. The algorithm itself also does not assume order, so this is not a concern. Looking at figure 22.3, we can see and use the tree in the figure, if t comes before x. However, if x comes before t and we have u before y, then (x,u) becomes an edge in the tree.

7. DFS(G):

```
V = G.V
for u in V:
   u.color = WHITE
   u.π = NULL
time = 0
S = NULL
for u in V:
   if u.color == WHITE:
      time = time + 1
      u.d = time
      u.color = GRAY
      push(S, vertex)
      while !(STACK-EMPTY(S)):
         vertex = top(S)
         flag = True
```

```
if vertex.color == GRAY:
    for w in G.Adj[vertex]:
        if w.color == WHITE:
            time = time + 1
            w.d = time
            w.color = GRAY
            push(S, w)
            flag = False
            break
if flag == True:
    time = time + 1
    vertex.f = time
    vertex.color = BLACK
    pop(S)
```