Analysis of Algorithms
Homework 4

Thomas Schollenberger (tss2344)

Question: 4

The expected case time complexity for the select algorithm is characterized by the following recurrence
$T(1) = 2$

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

Use techniques, including iteration, to solve the recurrence exactly.

Proof of 4: We know that:

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

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

From that definition, we can assert that:

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

Thus:

$$\begin{aligned}
T(n+1) - T(n)| &= (n+2) - (n+1) + \left(\frac{1}{n+1}\sum_{q=1}^{n} T(q) - \frac{1}{n}\sum_{q=1}^{n-1} T(q)\right) \\
| &= (n+2) - (n+1) + \left(\frac{1}{n+1}\sum_{q=1}^{n-1} T(q) - \frac{1}{n}\sum_{q=1}^{n-1} T(q)\right) + \frac{1}{n+1}T(n) \\
| &= 1 + \left(\frac{1}{n+1}\sum_{q=1}^{n-1} T(q) - \frac{1}{n}\sum_{q=1}^{n-1} T(q)\right) + \frac{1}{n+1}T(n) \\
| &= 1 + \left(\frac{1}{n+1} - \frac{1}{n}\right)\sum_{q=1}^{n-1} T(q) + \frac{1}{n+1}T(n) \\
| &= 1 - \frac{1}{n(n+1)}\sum_{q=1}^{n-1} T(q) + \frac{1}{n+1}T(n)
\end{aligned}$$

We can now substitute from the definition:

$$\begin{aligned}
T(n+1) - T(n)| &= 1 - \frac{1}{n(n+1)}\left(n(T(n) - (n+1)\right) + \frac{1}{n+1}T(n) \\
| &= 1 - \frac{T(n) - (n+1)}{(n+1)} + \frac{T(n)}{n+1} \\
| &= 1 - \left(\frac{T(n)}{n+1} - 1\right) + \frac{T(n)}{n+1} \\
| &= 1 + \frac{-T(n)}{n+1} + 1 + \frac{T(n)}{n+1} \\
| &= 2
\end{aligned}$$

Which means we end up with, after subtracting $-T(n)$:

$$\begin{aligned}
T(n+1)| &= T(n) + 2 \\
| &= 2n
\end{aligned}$$

Proving that our select algorithm runs in $O(n)$ time

😄

Question: 5

a. CLRS 20.1-1
b. CLRS 20.1-8

Answer to 5a: Out-degree: $\theta(V + E)$, In-degree: $\theta(VE)$ 😄

Answer to 5b: The expected look up time to see if an edge is in the graph is $O(\frac{n}{m})$, where n is the number of edges in the graph and m is the size of the hash table. This is because, due to the uniform hashing algorithm used in the table, the edges would all hash to the same value and then be chained to their respective index. This would make the hash table function almost identically to a linked list.
A better solution would be a matrix. If we grouped our matrix by adjacent edges, this would have $O(1)$ look up time. The matrix would take up more space, but the speed-space tradeoff would be worth it. 😄

Question: 6

CLRS 20.2-2

Answer to 6:

$$d_r = 4d_s = 3d_{t/x/y} = 1d_u = 0d_v = 5d_w = 2$$

😄