ECE 606, Fall 2019, Assignment 4

Zhijie Wang, Student ID number: 20856733

zhijie.wang@uwaterloo.ca

October 1, 2019

1. (a) $\Theta(s)$.

   Here we notice that DIVIDE is recursive, hence, $< 0, 0 >$ will be returned in the last iteration because the last iteration is DIVIDE$(0, y)$ and the second to last is DIVIDE$(1, y)$. Therefore, if x equals to n, Line (2) will be exactly executed $\lfloor \log_2 x \rfloor + 1$ times. Whatever y is, Line (2) will always be executed $\lfloor \log_2 x \rfloor + 1$ times. Therefore, in the worst case the input should be $< n, y >$, where y can be any natural number, and its running-time should be $\Theta(\log n)$.

   One last thing, when n is binary encoding, assume $n = 2^s$, then its running-time should be $\Theta(\log n) = \Theta(s)$, where input size is s.

   (b) *Proof.* Assume the input is $< x, y >$ of DIVIDE. Then, in the following proof, we carry out induction on x.

   Base case: $x = 0$. Then, the quotient and remainder should both be 0. In DIVIDE Line (2) will be executed and $< 0, 0 >$ will be returned, which is correct result.

   Step: our induction is that given $\lfloor \frac{x}{2} \rfloor$, then, DIVIDE$(\lfloor \frac{x}{2} \rfloor, y)$ will return correct quotient and remainder. We need now prove that given x, DIVIDE(x, y) is also correct. For Line (2), we can observe: $\lfloor \frac{x}{2} \rfloor = q_1 y + r_1$. Here we do a case analysis: (1) x is odd, and (2) x is even.

   (1) If x is odd, then, $\lfloor \frac{x}{2} \rfloor = \frac{x-1}{2}$. Therefore,

   $$\frac{x-1}{2} = q_1 y + r_1$$
   $$x - 1 = 2q_1 y + 2r_1$$
   $$x = 2q_1 y + 2r_1 + 1$$

   From Line (3) and Line (4), $q = 2q_1$, $r = 2r_1 + 1$. For Line (5), here we do another case analysis: A. $r < y$, and B. $r \geq y$.

   A. If $r < y$, then, $q = 2q_1$ and $r = 2r_1 + 1$ will be returned. Therefore, $x = 2q_1 y + 2r_1 + 1$, which is correct according to our analysis above.

   B. If $r \geq y$, then, let $q = 2q_1 + 1$ and $r = 2r_1 + 1 - y$. Therefore,

   $$\begin{aligned} x &= qy + r \\ &= (2q_1 + 1)y + 2r_1 + 1 - y \text{ which is also correct according our analysis.} \\ &= 2q_1 y + 2r_1 + 1 \end{aligned}$$

   (2) If x is even, then, $\lfloor \frac{x}{2} \rfloor = \frac{x}{2}$. Therefore,

   $$\frac{x}{2} = q_1 y + r_1$$
   $$x = 2q_1 y + 2r_1$$

   From Line (3) and Line (4), $q = 2q_1$, $r = 2r_1$. For Line (5), here we still do a case analysis: A. $r < y$, and B. $r \geq y$.

   A. If $r < y$, then, $q = 2q_1$ and $r = 2r_1$ will be returned. Therefore, $x = 2q_1 y + 2r_1$, which is correct according to our analysis above.

   B. If $r \geq y$, then, let $q = 2q_1 + 1$ and $r = 2r_1 - y$. Therefore,

   $$\begin{aligned} x &= qy + r \\ &= (2q_1 + 1)y + 2r_1 - y \text{ which is also correct according our analysis.} \\ &= 2q_1 y + 2r_1 \end{aligned}$$

   Therefore, DIVIDE is correct.

2. The pseudo-code for Q2 is shown as following. First, we use a $|V|$ size array to mark whether the vertex has been visited, initialize it with 0 then mark the input a as visited. Hence, we use a deep-first function to search the route from a to b. In the DFS, we search b in all vertexes connected to a and record the route length as *path*. If b is reached, then we firstly mark $is\_visited[b]$ as path length. If b is reached for more than one time, we use $unique = 0$ or 1 to mark whether only one shortest route between a and b. Finally, if we didn't reach b in this iteration, recursive into another one and replace a with this vertex. Therefore, when $is\_visited[b]$ is 0 after whole loop, which means there is no route between a and b. And, if $unique = 0$ means there is more than one shorter route and $unique = 1$ means there is only one shortest route.

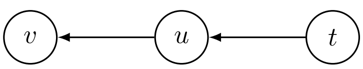
   USP(G, a, b)

   $is\_visited \leftarrow [0] * |V|$
   $is\_visited[a] \leftarrow 1$
   $path, unique \leftarrow 0$
   DFS$(G, a, b, is_visited, path, unique)$
   if $is\_visited[b] = 0$ **then**
       return 0
   else
       if $unique = 0$ **then**
           return 1
       else
           return 2
       end if
   end if

   DFS$(G, a, b, is_visited, path, unique)$

   for each $v \in E[a]$ **do**
       $path \leftarrow path + 1$
       if $v = b$ **then**
           if $is\_visited[b] = 0$ **then**
               $is\_visited[b] = path$
           else if $is\_visited[b] > path$ **then**
               $unique = 1$
           else if $is\_visited[b] = path$ **then**
               $unique = 0$
           end if
       else if $is\_visited[v] = 0$ **then**
           $is\_visited[v] = 1$
           DFS$(G, v, b, is_visited, path, unique)$
       else
           return
       end if
   end for

3. *Disprove.* Suppose we have a directed graph which have following structure, then, if we start with vertex v, then DFS will end up by forming v itself as a tree. If u is selected as next one, then, DFS will also end up by forming u itself as a tree. Therefore, DFS could end up with a vertex u that both in-degree and out-degree of us is > 0.



4. a4p4.py