Question 1 (Shortest Paths, 30 points). Compute the lengths of the shortest paths from A to each vertex of the graph below:

![Graph Image]

[Note: you do not need to show your work, but doing so will be useful for assigning partial credit if you make a mistake.]

The solution is given below with the bold edges being the ones in the shortest path tree:

![Solution Graph Image]

We can compute this using Dijkstra’s Algorithm. A is the only vertex at distance 0. From there you can reach B in distance 1. E is the only vertex reachable in distance 2. Continuing on in this manner finds vertices C, D, H, J, F, I, G in that order with their appropriate distances.
**Question 2** (Breakpoint Finding, 35 points). Given an array $A$ with $n$ elements and $A[1] \neq A[n]$ you would like to find an index $i$ so that $A[i] \neq A[i+1]$. Give an algorithm to solve this problem and analyze its runtime. For full credit, your algorithm should run in time $O(\log(n))$ or better.

We proceed by divide and conquer. If we know that $A[i] \neq A[j]$, with $j > i$, then if $j = i + 1$ we are done. Otherwise, we pick a $k$ halfway between $i$ and $j$. If $A[i] \neq A[k]$, we recurse on $i$ and $k$. If $A[k] \neq A[j]$, we recurse on $k$ and $j$. Notice that at least one of these must be true since otherwise $A[i] = A[k] = A[j]$, which violates our assumption. Formally, the algorithm is as follows:

```plaintext
Breakpoint(A, i, j) \ i<j, A[i] not equal to A[j]
    If j=i+1
        Return i
    Let k = Floor((i+j)/2)
    If A[i] not equal to A[k]
        Return Breakpoint(A, i, k)
    If A[k] not equal to A[j]
        Return Breakpoint(A, k, j)
```

That this algorithm works can be proved in a straightforward manner by induction on $j-i$. In particular, we prove by induction on $m$ that if $j - i = m$ and if $A[i] \neq A[j]$, then the algorithm produces a valid response. If $j - i = 1$, then $A[i] \neq A[i+1]$, and the return of $i$ is correct. If $j = i + m$ and $A[i] \neq A[j]$, then either $A[i] \neq A[k]$ or $A[k] \neq A[j]$ and the recursive call to Breakpoint has valid inputs that differ by less than $m$ so by the inductive hypothesis will return the correct answer.

To analyze the runtime, we note that each recursive call reduces $|i - j|$ by a factor of 2. Furthermore, this algorithm requires only a constant amount of overhead. Therefore, letting $T(n)$ be the runtime when $|i - j| = n$, we have that

$$T(n) = T(n/2) + O(1),$$

so by the Master Theorem, $T(n) = O(\log(n))$. 

---

2
Question 3 (Optimal Refuelling, 35 points). Thomas is taking a trip along a highway. This stretch of highway has \( n \) rest stops labelled 1, 2, \ldots, \( n \) in order with stop 1 being his initial location and stop \( n \) being his final destination. Unfortunately, Thomas does not have enough gas to make the entire journey on one tank. At each stop \( i \), there is a fuel station that provides enough fuel to get him to any stop up to and including stop \( F(i) \) (but he cannot over-fuel or save fuel from other stops). So for example from stop 7, Thomas can drive to any of stops 8, 9, 10, \ldots, \( F(7) \) without stopping. Thomas wishes to reach his destination with as few stops as possible. Thomas proposes the following greedy algorithm to schedule his trip: after reaching stop \( i \), he drive to the end if possible, and otherwise will drive to the stop \( j \leq F(i) \) with \( F(j) \) as large as possible.

Prove that this algorithm will always produce a schedule with the minimum possible number of stops.

We proceed by an exchange argument. Suppose that Thomas’ algorithm produces the sequence of stops \( s_1, s_2, \ldots, s_k \) with \( s_1 = 1, s_k = n \) and \( s_{i+1} \leq F(s_i) \). Suppose that we have another valid path \( t_1, \ldots, t_m \). We claim that by induction on \( 1 \leq i < k \) that \( i < m \) and that \( s_1, s_2, \ldots, s_i, t_{i+1}, t_{i+2}, \ldots, t_m \) is a valid schedule.

For the base case, we note that clearly \( 1 < m \) and that since \( s_1 = t_1 = 1 \), that \( s_1, t_2, t_3, \ldots, t_m \) must be a valid schedule.

Next, suppose that \( i < k - 1 \) and that \( s_1, s_2, \ldots, s_i, t_{i+1}, t_{i+2}, \ldots, t_m \) is a valid schedule. First, we note that since \( s_{i+1} \neq n \), \( F(s_i) \) must be less than \( n \) (otherwise Thomas would have let \( s_{i+1} = n \)). Therefore \( m > i + 1 \). Next note that by the inductive hypothesis, since \( t_{i+1} \) can follow \( s_i \), we have that \( t_{i+1} \leq F(s_i) \). However, by construction \( s_{i+1} \) is the number less than \( F(s_i) \) with the largest value of \( F(x) \). Hence, \( F(s_{i+1}) \geq F(t_{i+1}) \geq t_{i+2} \). Therefore, \( s_1, s_2, \ldots, s_{i+1}, t_{i+2}, \ldots, t_m \) is a valid schedule. This completes our inductive step.

Applying this induction to \( i = k - 1 \), we conclude that \( m > k - 1 \) and thus, \( m \geq k \). Since Thomas’ schedule has as most as many stops as an arbitrary schedule, it must be optimal.