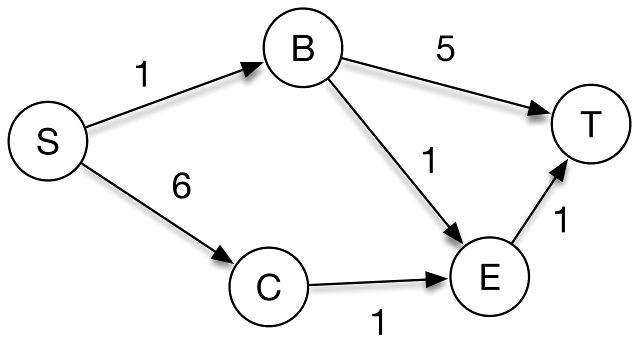
Name & UW NetID:

1. Run Dijkstra's shortest path algorithm in the following graph with vertex S as the source, and fill the table below with the results.



Vertex	Distance	Predecessor	Processed
S	0	-	✓
B	1	S	✓
C	6	S	✓
T	~~6~~ 3	~~B~~ E	✓
E	2	B	✓

2. Following is the pseudocode for Dijkstra's shortest path algorithm with binary heap. Do runtime analysis of this code. (Annotate the pseudocode with big-$\mathcal{O}$ values next to relevant statements/loops in the code.)

```
function Dijkstra(Graph G, Vertex source)                          ▷ with MPQ
    initialize distances to ∞                  → O(|V|)
    source.dist = 0
    mark all vertices unprocessed              → O(|V|)
    initialize MPQ as a min priority queue
    add source with priority 0
    while MPQ is not empty do                  → at most |V| times
        u = MPQ.getMin()                       → O(1)
        for each edge (u,v) leaving u do       → at most |E| times in all the |V| iterations of the outer while loop
            if u.dist + w(u,v) < v.dist then
                if v.dist == ∞ then
                    MPQ.insert(v, u.dist + w(u, v))                → O(log|V|)
                else
                    MPQ.decreasePriority(v, u.dist + w(u,v))
                end if
                v.dist = u.dist + w(u,v)       | O(1)
                v.predecessor = u
            end if
        end for
        mark u as processed                    O(|V|) + O((|V| + |E|) log |V|)
    end while
end function
```

Total

$O(|V| \log|V| + |E| \log|V|)$

3. Circle the strong connected components in the following graph.