

CS 512: Design and Analysis of Algorithms

Autumn 2020-2021

Homework # 2

Due Date: 05-10-2020

Total Marks: 20

October 2, 2020

Important

1. Typeset your answers using \LaTeX or Word. Upload a pdf file as your submission.
2. Identical answers by two students on the same problem will incur zero marks for both students for the problem.
3. Copying answers from the Internet will also be penalized by awarding zero marks.
4. A plagiarism checker will be used to detect all types of copying.
5. Include your name and roll number at the top of your answer script.

1. Consider Algorithm 1 with inputs an undirected graph $G = (V, E)$ and a vertex $s \in V$. Here $\text{MakeEmptyQueue}(Q)$ creates an empty queue called Q , $\text{InsertQueue}(Q, v)$ inserts v at the end of the queue Q and $\text{DeleteQueue}(Q)$ returns the head of the queue Q and removes it from Q .
 - (a) What does (Algorithm 1) do, *i.e.*, what value does $\text{num}[v]$ hold at the end of the algorithm? Justify your answer. (8 marks)
 - (b) What is its time complexity of the algorithm? Justify your answer. (2 marks)
2. Consider Algorithm 2 with inputs an undirected graph $G = (V, E)$, positive edge weights l_e for each edge in E and a vertex $s \in V$. Here $\text{MakeEmptyPriorityQueue}(Q)$ creates an empty min-ordered priority queue Q , $\text{IsNotEmpty}(Q)$ returns false if Q is empty and true otherwise, $\text{InsertPriorityQueue}(Q, v, c)$ inserts v into Q with key value c , $\text{DeleteMin}(Q)$ returns the minimum element of Q and deletes it from Q and $\text{DecreaseKey}(Q, v, c)$ decreases the key value of the element v to c in Q .

- (a) What does (Algorithm 2) do, *i.e.*, what value does $IsGood[v]$ hold at the end of the algorithm? Justify your answer. (8 marks)
- (b) What is its time complexity of the algorithm? Justify your answer. (2 marks)

Algorithm 1 Strange Algorithm

Input Undirected graph $G \leftarrow (V, E)$ and a vertex $s \in V$

Output For every vertex v , a positive integer $num[v]$

```
1: for each  $v \in V$  do
2:   initialize  $num[v] \leftarrow 0$ 
3: end for
4: for each  $v \in V$  do
5:   initialize  $cost[v] \leftarrow \infty$  // integer array indexed by elements of  $V$ 
6: end for
7:  $cost[s] \leftarrow 0$ 
8:  $num[s] \leftarrow 1$ 
9:  $MakeEmptyQueue(Q)$ 
10:  $InsertQueue(Q, s)$ 
11: while  $Q \neq \emptyset$  do
12:    $u \leftarrow DeleteQueue(Q)$ 
13:   for all edges  $(u, v) \in E$  do
14:     if  $cost[v] = \infty$  then
15:        $cost[v] \leftarrow cost[u] + 1$ 
16:        $num[v] \leftarrow num[u]$ 
17:        $InsertQueue(Q, v)$ 
18:     else
19:       if  $cost[v] = cost[u] + 1$  then
20:          $num[v] \leftarrow num[v] + num[u]$ 
21:       end if
22:     end if
23:   end for
24: end while
```

Algorithm 2 Weird Algorithm

Input Undirected graph $G \leftarrow (V, E)$ with edge weights $l_e > 0$ for all $e \in E$ and a vertex $s \in V$

Output For every vertex v , a Boolean value $IsGood[v]$

```
1: for each vertex  $v$  do
2:    $cost[v] \leftarrow \infty$  // integer array indexed by elements of  $V$ 
3:    $IsGood[v] \leftarrow \text{TRUE}$ 
4: end for
5:  $cost[s] \leftarrow 0$ 
6:  $MakeEmptyPriorityQueue(Q)$ 
7: for each  $v \in V$  do
8:    $InsertPriorityQueue(Q, v, cost[v])$ 
9: end for
10: while  $IsNotEmpty(Q)$  do
11:    $u \leftarrow DeleteMin(Q)$ 
12:   for each edge  $e = (u, v) \in E$ , leaving  $u$  do
13:     if  $cost[v] > cost[u] + l_e$  then
14:        $cost[v] \leftarrow cost[u] + l_e$ 
15:        $DecreaseKey(Q, v, cost[v])$ 
16:        $IsGood[v] \leftarrow IsGood[u]$ 
17:     else
18:        $IsGood[v] \leftarrow \text{FALSE}$ 
19:     end if
20:   end for
21: end while
```
