# 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 $\mathit{MakeEmptyQueue}(Q)$ creates an empty queue called $Q$, $\mathit{InsertQueue}(Q, v)$ inserts $v$ at the end of the queue $Q$ and $\mathit{DeleteQueue}(Q)$ returns the head of the queue $Q$ and removes it from $Q$.

   (a) What does (Algorithm 1) do, *i.e.*, what value does $num[v]$ 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 $\mathit{MakeEmptyPriorityQueue}(Q)$ creates an empty min-ordered priority queue $Q$, $\mathit{IsNotEmpty}(Q)$ returns false if $Q$ is empty and true otherwise, $\mathit{InsertPriorityQueue}(Q, v, c)$ inserts $v$ into $Q$ with key value $c$, $\mathit{DeleteMin}(Q)$ returns the minimum element of $Q$ and deletes it from $Q$ and $\mathit{DecreaseKey}(Q, v, c)$ 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 $num[v]$

**for** each $v \in V$ **do**
  initialize $num[v] \leftarrow 0$
**end for**
**for** each $v \in V$ **do**
  initialize $cost[v] \leftarrow \infty$ // integer array indexed by elements of $V$
**end for**
$cost[s] \leftarrow 0$
$num[s] \leftarrow 1$
$MakeEmptyQueue(Q)$
$InsertQueue(Q, s)$
**while** $Q \neq \emptyset$ **do**
  $u \leftarrow DeleteQueue(Q)$
  **for** all edges $(u, v) \in E$ **do**
    **if** $cost[v] = \infty$ **then**
      $cost[v] \leftarrow cost[u] + 1$
      $num[v] \leftarrow num[u]$
      $InsertQueue(Q, v)$
    **else**
      **if** $cost[v] = cost[u] + 1$ **then**
        $num[v] \leftarrow num[v] + num[u]$
      **end if**
    **end if**
  **end for**
**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 $IsGood[v]$

```
for each vertex v do
    cost[v] ← ∞ // integer array indexed by elements of V
    IsGood[v] ← TRUE
end for
cost[s] ← 0
MakeEmptyPriorityQueue(Q)
for each v ∈ V do
    InsertPriorityQueue(Q, v, cost[v])
end for
while IsNotEmpty(Q) do
    u ← DeleteMin(Q)
    for each edge e = (u, v) ∈ E, leaving u do
        if cost[v] > cost[u] + l_e then
            cost[v] ← cost[u] + l_e
            DecreaseKey(Q, v, cost[v])
            IsGood[v] ← IsGood[u]
        else
            IsGood[v] ← FALSE
        end if
    end for
end while
```