Homework 12 Programming

(Maximum 25 points)
Due 11:59 pm Wednesday May 5, 2021

Submit your Java codes in one file via Blackboard. Refer to the grading guidelines posted on Blackboard to understand how the submitted exercises will be graded.

1. (25) [Maximum flow: Ford-Fulkerson programming] Write a Java program that implements the Ford-Fulkerson maximum flow algorithm pseudocode discussed in class and revised as shown below, and run your program against the flow network shown below. Note the following specifications in writing the program.

- The flow network G can be hard-coded in your program, but the residual graph G_f should be constructed by your program.
- Make an arbitrary choice of an augmenting path at each iteration – depth-first or breadth-first search is fine.
- Your program code should maintain the structure of the algorithm pseudocode and should include the Max-Flow, constructResidualGraph, findAugmentingPath, Augment, bottleneck, and updateResidualGraph as separate functions.
- The input/output signature of each function may vary depending on your implementation.
- The flow network G and the residual graph G_f can be represented in any data structure (e.g., adjacency list, adjacency matrix, edge list).
- At the beginning of the execution, your program should output which graph data structure has been used, e.g., "An adjacency is listed to represent a graph."
- During the execution, your program should output the augmenting path and the augmented flow amount each time augmented flow is added to the flow network. An example of the output format is "s → 1 → 3 → t" for an augmenting path and '8' for an augmented flow amount.
- At the end of the execution, your program should output the resulting maximum flow value, e.g., "Maximum flow value = 28."

It is okay to look up Ford-Fulkerson Java code in another source (e.g., book, Internet) as long as you cite the source clearly.

```
Augment(f, P) {
   b ← bottleneck(P)
   foreach e ∈ P {
      if (e ∈ E) f(e) ← f(e) + b
      else       f(e) ← f(e) - b
   }
   return f
}
```

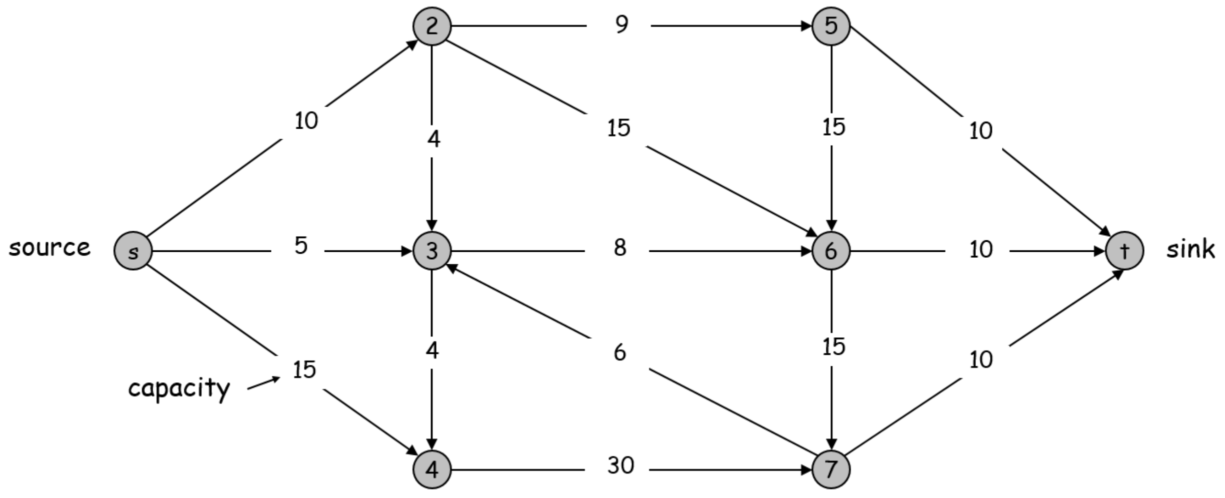
f belongs to G, P to G_f

e is a forward edge in G_f

e is a backward edge in G_f

```
Max-Flow(G, s, t) {
   foreach e ∈ E  f(e) ← 0
   Gf ← constructResidualGraph(G)

   P = findAugmentingPath(Gf, s, t)
   while (P != null) {
      f' ← Augment(f, P)
      updateResidualGraph(Gf, f, f')
      P = findAugmentingPath(Gf, s, t)
   }
   return f
}
```



Submit the source codes via Blackboard. Your program should work correctly. In addition, program codes should be neatly organized and well commented.

Last modified: April 22, 2021