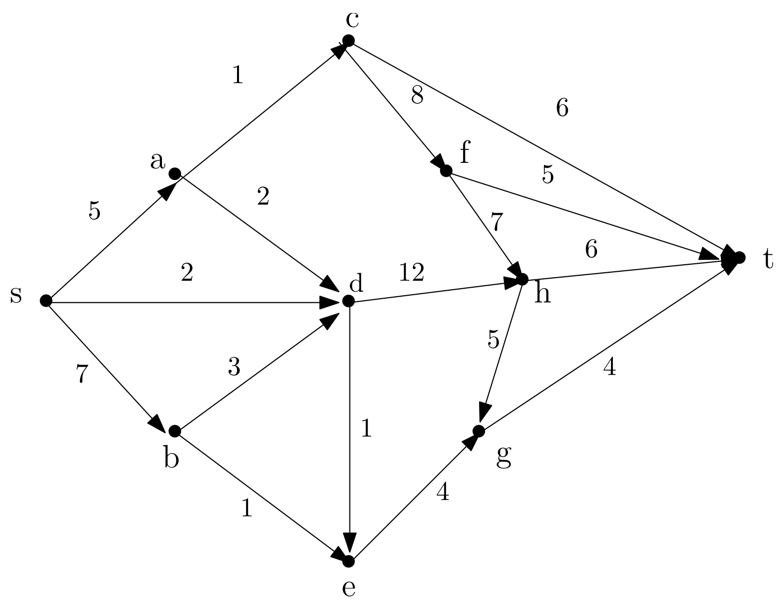
COMP 3711H – Fall 2014

Tutorial 10 – Revised Nov 20, 2014

1. For the graph below find a max-flow from s to t.
   Prove that it is a max-flow (by showing that it is equal to the capacity of some s-t cut)



2. Let $G = (V, E)$ be a directed graph with designated source s and sink t. Two s-t paths are *edge-disjoint* if they do not share any edges. A set of k paths is edge-disjoint if all pairs of paths in the set are edge-disjoint.

   Having a large number of edge-disjoint paths is a good first indication of network robustness (to work around edge failures). The larger the set of k-disjoint paths, the more robust is to failures.

   Show how to use the Max-Flow algorithm to
   (a) efficiently find the largest number k such that G contains a set of k edge-disjoint paths and
   (b) find the corrresponding set of k edge-disjoint paths

3. In class we learned how to solve the max-flow problem on directed graph G that has given capacities on edges.

   Now suppose that instead of being given capacities $c(e)$ for each *edge* $e \in E$ we are given capacities $c(v) \geq 0$ on *vertex* $v \in V - \{s, t\}$.

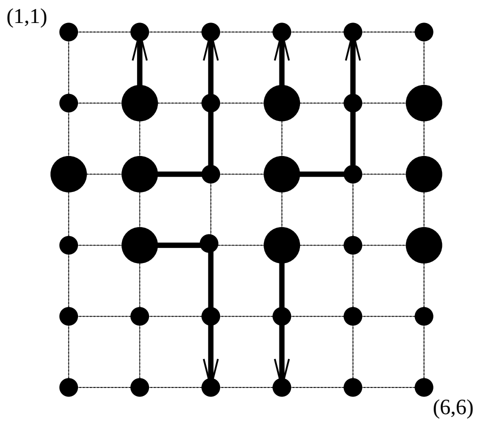
   More specifically, arbitrarily large amounts of flow are allowed on any *edge* but
   (a) the total flow *into* any vertex $v \neq s, t$ is at most $c(v)$ and
   (b) all the flow that enters vertex $v \neq s, t$ must leave v.

   The problem now is to find the largest amount of flow that can be sent from s to t satisfying these capacity constraints.

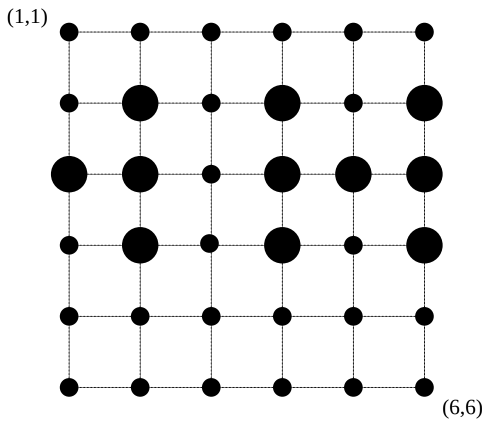
   For every directed graph $G = (V, E)$ with vertex capacities, show how to create a new graph G' with edge capacities that models the vertex capacity problem on G such that solving the max flow problem on G' yields a max flow (satisfying vertex capacities) on G.

4. **A non-obvious use of max flow**

   A $n \times n$ grid is an undirected graph with n rows and columns of vertices, as shown below. Imagine that a person is standing at every *marked vertex* (shown as a big black circle), and that the edges represent very narrow hallways. The *escape problem* is to determine whether it is possible to find a safe set of non-touching escape routes, that allows every person to escape separately. In other words, no two people should pass through the same vertex.



A grid with a valid set of escape routes



A grid that has no valid set of escape routes

   The problem is to design an efficient algorithm to solve the escape problem for any n and for any given set of m marked vertices. "An escape exists" means there exists a set of m vertex-disjoint paths (which means no two paths share any vertex) leading each of the people out to some point on the perimeter.

   Hint: Use the transformation from the previous problem to change the vertex capacity problem (with capacity one for each vertex) into an edge capacity problem. this will give you a max-flow problem with many sources and many sinks. Recall how we dealt with that in class.

5. Consider the problem of neatly formatting a paragraph in a word processor *without* right alignment. The input text is a sequence of n words of lengths $\ell_1, \ell_2, \ldots, \ell_n$, measured in characters. We want to format the paragraph neatly on a number of lines that hold a maximum of M characters each. Our criterion of "neatness" is as follows. If a given line contains words i through j and we leave exactly one space between words, the number of extra space characters at the end of the line is $M - j + i - \sum_{k=i}^{j} \ell_k$. We wish to minimize the sum, over all lines except the last, of the cubes of the numbers of extra space characters at the end of lines. Give a dynamic-programming algorithm to print a paragraph of n words neatly on a printer. Analyze the running time and space requirements of your algorithm.