Problem Session 6

Network Flow

Note: I did not write the forth question which is about dynamic programming that I have solved in the class. Instead, I have added another question about network flows as forth question.

1. Figure shows a flow network on which an *s-t* flow has been computed. The capacity of each edge appears as a label next to the edge, and the numbers in boxes give the amount of flow sent on each edge. (Edges without boxed numbers have no flow being sent on them.)

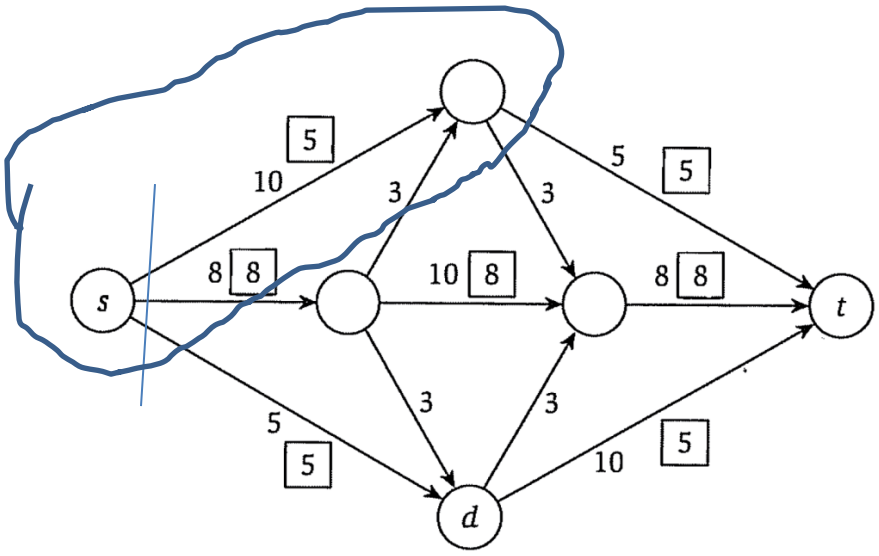
(a) What is the value of depicted flow? Is this a maximum *(s,t)* flow in this graph?
Value:18. No

(b) Find a minimum s-t cut in the flow network and its capacity.
Min cut def. The capacity of a cut (A, B) is:

$$cap(A, B) = \sum_{e \text{ out of } A} c(e)$$

Circle is min cut: 5+8+3+5 = 21

Max-flow min-cut theorem. [Ford-Fulkerson 1956] The value of the max flow is equal to the value of the min cut. For both questions prove your answers by applying Ford-Fulkerson algorithm. Draw residual graph and find max flow = min capacity value.



2. Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.
Let G be an arbitrary flow network, with a source s, a sink t, and a positive integer capacity c_e on every edge e. If f is a maximum s-t flow in G, then f saturates every edge out of s with flow (i.e., for all edges e out f s, we have f(e) = c_e).

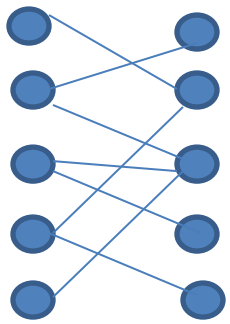
False. Consider a graph with nodes s,v,w,t, edges (s,v), (v,w), (w,t), capacities of (s,v) and (w, t) is 2, capacity of (v,w) is 1. Then the maximum flow has value 1, and does not saturate the edge out of s.

3. Decide whether you think the following statement is true or false. If it is true, give a short explanation. If it is false, give a counterexample.
Let G be an arbitrary flow network, with a source s, a sink t, and a positive integer capacity c_e on every edge e; and let (A, B) be a minimum s-t cut with respect to these capacities $\{c_e : e \in E\}$. Now suppose we add 1 to every capacity; then (A, *B) is still a minimum s-t cut with respect to these new capacities $\{1+c_e\}$.*

False. Consider a graph with nodes s,v1,v2,v3,w,t,edges (s,vi) and (vi,w) for each i, and an edge (w,t). There is a capacity of 4 on the egde (w,t), and a capacity of 1 on all other edges. Then setting A = {s} and B = V- A gives a minimum cut, with capacity 3. But if we add one to every edge then this cut has capacity 6, more than the capacity of 5 on the cut with B = {t} and A = V-B.

4. The following graph shows a number of job applicants (A) and companies (B). If there is an edge between a person and a company, it means that the person has applied to the company and the company is interested in the person. A person can be employed at only one company and a company can hire only one person. We want to have as many people placed in jobs as possible. What is the maximum number of people that can be placed at a job?

a. Show the method and the pseudocode of the algorithm you used.
b. Show the execution of your algorithm.
c. What is the complexity of your algorithm in terms of the number of people N_A and the number of companies N_B.



Dynamic Programming

Box Stacking

You are given a set of n types of rectangular 3-D boxes, where the i^th box has height h(i), width w(i) and depth d(i) (all real numbers). You want to create a stack of boxes which is as tall as possible, but you can only stack a box on top of another box if the dimensions of the 2-D base of the lower box are each strictly larger than those of the 2-D base of the higher box. Of course, you can rotate a box so that any side functions as its base. It is also allowable to use multiple instances of the same type of box.

Suggest an algorithm for this problem and apply it to the following input.

wdh
1 1 1
1 2 3
4 5 6

Generate all possible rotations for each box, let's say N to the total number
Sort them according to descending order of 2D base area, store them a struct array called block
Create an array called heights size of N
For each box type (i=0 to N)

```
flag = false;
max = 0;
For j = 0 to i
        If block[j].width > block[i].width && block[j].depth > block[i].depth
                flag = true;
                If heights[max] <= heights[j]
                        max= j;
if flag ==true
        heights[i] = heights[max] + block[i].height
else
        heights[i] = block[i].height
```

Print max value of heights array

Ex: Possible blocks: 111,123,132,231,456,465,564
Heights: 4,5,10,11,12,14,12

So maximum height is 14 (123,231,456,564).