

Revised 1/26

1. Show how to solve a network flow problem where it's the nodes that have capacities, not the edges. Solve this by telling how to *reduce* this problem to one where only edges have capacities. *Hint: turn each node  $v_i$  into two nodes,  $x_i$  and  $y_i$ , and install a directed edge from  $x_i$  to  $y_i$ . You can have edges of infinite capacity in a network flow problem.*

Instead of explaining your solution, illustrate it on the following undirected network:

$V = \{s, a, b, c, d, t\}$ ,  $E = \{sa, sb, ac, ad, bd, ct, dt\}$ , these edges have infinite capacity, and  $c(s) = \text{infinity}$ ,  $c(a) = 5$ ,  $c(b) = 3$ ,  $c(c) = 3$ ,  $c(d) = 5$ ,  $c(t) = \text{infinity}$ .

2. **Corrected** Consider directed graphs of the following form:

$V = \{s = c_0, a_1, b_1, c_1, a_2, b_2, c_2, \dots, a_n, b_n, c_n = t\}$ ,  $E = \{(c_i, a_{i+1}), (c_i, b_{i+1}) | 0 \leq i < n\} \cup \{(a_i, c_i), (b_i, c_i) | 1 \leq i \leq n\}$ .

- (a) Draw a picture of this graph where  $n = 3$ .
  - (b) Give an expression for the number of paths from  $s$  to  $t$ . It should be a function of  $n$ .
3. **Clarified** The main insight from the previous problem is that the number of paths from  $s$  to  $t$  can be exponential.

Give a polynomial algorithm for finding a minimum set of nodes of a network that can be taken out by a saboteur in order to cut off all paths from  $s$  to  $t$ . Your algorithm should consist of a reduction to max flow in order to get a min cut that solves the problem.

Combine the max-flow min-cut theorem with the trick from the first problem. Study the tricks of Figure 26.8 for some insights.

4. **Clarified** You want to route a set of  $k$  packets from node  $s$  to node  $t$ . You have reason to think that the security of one of the nodes of the network has been compromised, but you only know that it is not  $s$  or  $t$ .

No packet by itself will give away any secrets to an eavesdropper, but, taken together, two might. (For example, suppose that, for security reasons, each credit-card number is split between two packets.) Therefore, you want to route the packets so that no two of them go through the same node.

You need an algorithm that determines that this is impossible, or else gives a set of paths that satisfy the constraints. Describe a way to reduce the problem to max flow.

5. 26-5, page 762. Observe that since  $K$  starts out at a power of two, each time it halves it will be a power of two, hence an even number except on the last iteration, when it is 1. Each time you divide by two, there is no remainder to throw away.
  - (a) For part a, your argument should mention a bound on the sum of capacities of all the edges of the graph.

(b) I will do part b for you.

Toss out edges of capacity less than  $K$  and run DFS, starting at  $s$ . If  $s$  and  $t$  are still in the same connected component of this graph, the DFS will find a path from  $s$  to  $t$  whose capacities are least  $K$ . This will be an augmenting path of capacity  $K$ . If  $s$  and  $t$  are now in different components, there can't be an augmenting path of capacity at least  $K$ , since any augmenting path in the original graph must make use of one of the edges we have tossed, and that edge will be a bottleneck smaller than  $K$ .

We don't need to actually toss edges; we can just alter DFS so that it ignores edges of capacity less than  $K$ . From an earlier chapter of the book, we know (or can look up) that a call to DFS takes  $O(E)$  time.

(c) For part c, your argument should be based on the exit conditions of the two loops on the last iteration. What do they tell you about the residual graph? You should make your case in about two sentences.

(d) For part d, for the first iteration, this follows from part a and your initial choice of  $K$ , which is greater than  $C/2$ .

On subsequent iterations, it is no longer true that every edge has capacity less than  $2K$ . Think about what condition caused the inner loop to exit most recently, however. Combine this condition with the trick from part b to get an algorithm for finding a cut in the residual graph at this point in which each edge has capacity less than  $2K$ . Your answer should be at least as simple as my answer to part b.

Now argue in one or two sentences that capacity of this cut can't be greater than  $2K|E|$ .

(e) For part e, think about what part d says about the value of a max flow in the residual graph at the beginning of iteration  $K$ . Then explain why this means that there can be no more than  $O(E)$  augmentations in iteration  $K$ .

(f) For part f, describe how to combine parts b and e, as well as the fact that  $K$  starts out less than or equal to  $C$  and halves at each iteration.

26-3 parts a and b. To gain insight, draw the graph for an example. Let the  $c_1 = 8$ ,  $c_2 = 2$ ,  $c_3 = 5$ ,  $c_4 = 13$ ,  $c_5 = 4$ . Let  $p_1 = 7$ ,  $p_2 = 9$ ,  $p_3 = 11$ ,  $p_4 = 1$ ,  $p_5 = 2$ .

Suppose  $J_1$  requires  $A_1$  and  $A_2$ ,  $J_2$  requires  $A_1$  and  $A_3$ ,  $J_3$  requires  $A_2$ ,  $A_3$ , and  $A_4$ , and each of  $J_4$  and  $J_5$  requires  $A_4$  and  $A_5$ .

Draw the graph they suggest. Find a max flow by hand. Make sure it is a max flow by finding a min cut. How does this cut tell you an optimum solution?