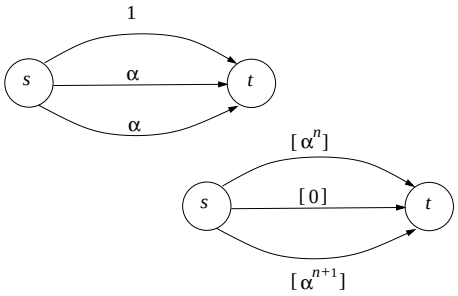
MA346m Quiz 05, due noon, Friday 15/12/17

Answer 2 questions, as usual

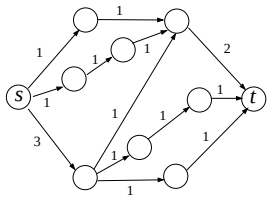
(1). The Ford-Fulkerson algorithm may run forever. The figure below illustrates the idea. Admittedly it is on a multigraph; but it's easy to convert to a flow network in the usual sense. The figures in square brackets are the residual capacities, though sometimes in a forward and sometimes in a backward direction. Of course, α must satisfy a certain quadratic equation.

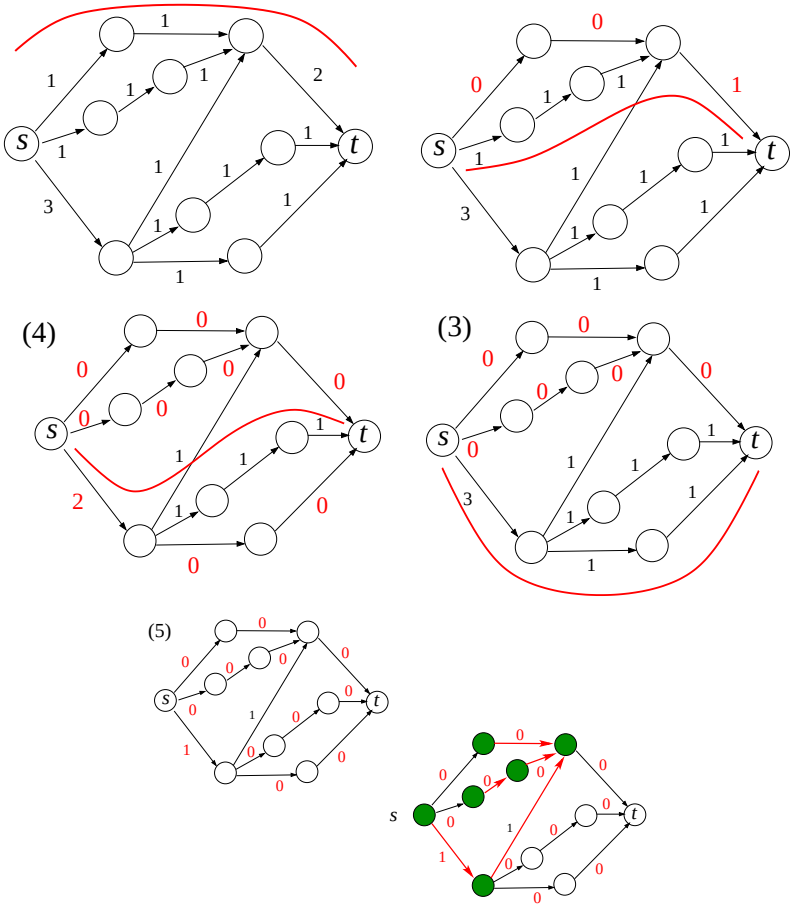


Calculate α and exhibit an infinite sequence of augmentations.

Answer. $\alpha > 0$ satisfies $\alpha^2 = 1 - \alpha$ so $\alpha = (-1 + \sqrt{5})/2$. Take the initial (zero) flow, with residual capacities $1, \alpha, \alpha$. Raise the flow as much as possible along the middle edge, leaving residual capacities $1, 0, \alpha$. Now raise the third edge and the first by α reducing the middle by α (back edge), getting residual capacities $\alpha^2, \alpha, 0$. In general with residual capacities $\alpha^{n+1}, 0, \alpha^n$ (in some order), the first edge can be saturated, flow raised on the third and lowered on the middle edge: $0, \alpha^{n+1}, \alpha^n - \alpha^{n+1} = \alpha^{n+2}$. Thus the general pattern $\alpha^{n+1}, 0, \alpha^n$ can be continued indefinitely. ∎

(2). Calculate a maximal flow, on the network shown below, by several augmentations. Your answer should include a minimum-capacity cutset.

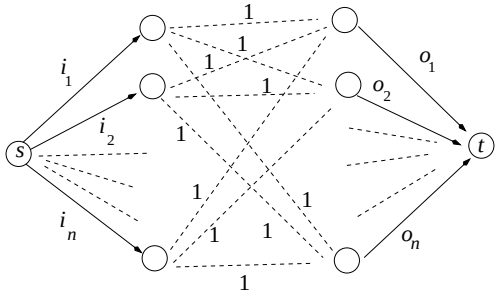




(3). There is a curious application of network flows. Given a list $(i_1, o_1), \ldots, (i_n, o_n)$ of pairs of nonnegative integers, to construct a digraph G whose j-th vertex has indegree i_j and outdegree o_j, if it exists, and to show that no such digraph exists, otherwise.

Describe how to convert the 'degree problem' to a network flow problem.

Answer Create a network with $2n + 2$ vertices at 4 levels: s at level 0, then n at level 1, n at level 2, and t at level 3. Connect s to all vertices at level 1, with capacity i_j connecting s to the j-th vertex. Connect all vertices at level 2 to t, with capacities o_j. Connect all vertices at level 1 to all at level 2, with capacities 1. Construct a maximal flow on this network...



(4). There is a bijection between binary trees on n nodes with balanced sets of $2n$ parentheses, allowing one alternative ways to count both. It is also the number of ways the numbers $1 \dots n$ can be output by pushing them on a stack with intermittent popping.

For each n, the number of trees/balanced sets/restricted permutations is the n-th Catalan number

$$\frac{1}{n+1}\binom{2n}{n}$$

Derive this estimate for any one of the three structures mentioned above.

This can be calculated in at least three ways: by using the general Binomial Theorem on a generating function, by Lagrange inversion on same, or by a trick mentioned in 'introduction to probability theory...' by Feller.

To save time, here is a derivation of the generating function. Let b_n be the number of binary trees on n nodes and let $B(x) = \sum_n b_n x^n$. We take it that $b_0 = 1$.

A tree with $n+1$ nodes separates into left and right subtrees plus the root. If the left and right subtrees have i and j respectively then $i + j = n$. Therefore

$$b_{n+1} = \sum_{i+j=n} b_i b_j$$

$$b_{n+1}x^{n+1} = x \sum_{i+j=n} b_i x^i b_j x^j$$

$$B(x) - 1 = xB^2(x).$$

$$xB^2 - B + 1 = 0$$

$$B = \frac{1 \pm \sqrt{1-4x}}{2}$$

zero at 0

$$\frac{1-\sqrt{1-4x}}{2x}$$

$$b_0 = 0$$

$$-\frac{1}{2}\sum \binom{\frac{1}{2}}{n+1}(-4)^{n+1}x^n$$

$$b_n = -\frac{1}{2}\left(\frac{(1/2)(1/2-1)\cdots(1/2-(n+1)+1)(-2)^{n+1}2^{n+1}}{(n+1)!}\right)$$

$$\frac{2^n(2n-1)\times(2n-3)\cdots 3\times 1}{(n+1)!} =$$

$$\frac{(2n)!}{n!(n+1)!} = \frac{1}{n+1}\binom{2n}{n}.$$

(5). True or false: Tarjan's SCC algorithm emits sccs according to the postorder rank of their roots. (Explain your answer.)

Answer. True. Let v and w be two SCC roots. Suppose first that neither is a descendant of the other, and, say, that v is processed first. Then v is output before dfs(w) begins, and w

gets higher postorder. If on the other hand v is a descendant of w, then v is processed before w and v precedes w in postorder. ∎