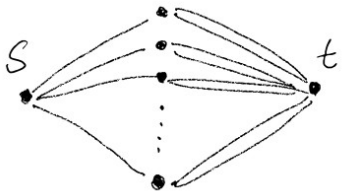
1.8

a

Consider the graph



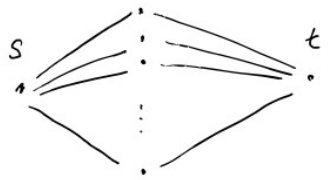
Removing any edge on the left side will increase the (s-t) min cut. Initially the left-side edges make up $\frac{1}{3}$ of the total amount of edges. At every stage if the algorithm has not contracted any wrong edge, the probability of contracting a wrong edge is than at least $\frac{1}{3}$. Thus we see that the probability of obtaining the min-cut is at most

$$\left(\frac{2}{3}\right)^{n-1}$$

Where n is the total number of edges. (This is exponentially decreasing in n).

b

Observe that any (s-t) min-cut define a unique split (2-partition) of the vertices. To see this first note that an (s-t) min-cut defines a split of the vertices defined by whether they are connected with s or not. Since it is a *cut* it must have removed any edge between the two groups of vertices. And since it is minimal, it cannot have removed any other edges. Thus it is unique. Now the number of splits of n vertices with s in the first group and t in the second is 2^{n-2}. So this is an upper bound on the number of min-cuts. Consider the graph



This achieves the upperbound of 2^{n-2} min-cuts, since any for any middle vertex you can choose either the left or the right edge to remove, resulting in a min-cut when done for all middle vertices.

2.1

We claim that for any deterministic evaluation algorithm there is a $T_{d,k}$ tree which forces the algorithm to evaluate all subnodes (and leafs), and furthermore we can choose the root node to evaluate to 0 and 1 as we please. Using DeMorgan it is easy to see that d-$\wedge$-$\vee$-trees are equivalent to d-$\downarrow$-trees so it is enough to consider these (we even allow the height to be odd). We proceed by induction on the height h of the tree. If $h = 0$ we are at leaf so we read $1 = d^0$ which we can choose to be 0 or 1. If $h = n$ its subtrees are of lower height. By induction we can choose the first $d-1$ choices of subtrees of the algorithm to evaluate - using all nodes - to 0. This is because each zero forces the algorithm to evaluate more subtrees. The last chosen subtree of the algorithm we can choose to evaluate to 0 or 1 depending on whether we want the node to evaluate to 1 resp. 0. This completes the argument.