Solution Set 7

Due: In class on Wednesday, April 7. Starred problems are optional.

Problem 7-1. Show that if d is an exact power of 2, then the 2^d-node cube-connected-cycles network is a subgraph of the 2^d-node hypercube.

Solution: Consider a cube-connected-cycles network consisting of $d2^d$ nodes, i.e. one which is equivalent to a d-dimensional hypercube with a d-node cycle at each hypercube node. We give to each node of the cube-connected-cycle a $d + \lg d$-bit label, and then show that we can embed this network in a $d + \lg d$-dimensional hypercube without changing the node labels.

Each label will be made up of two components, one of d bits and one of $\lg d$ bits. The d high bits come from the position of the node's containing cycle in the d-dimensional hypercube. We give each cycle a d-bit label in the ordinary manner for a hypercube (such that traversing any edge flips exactly one bit), and use that label as the high b bits for every node in that cycle.

For the $\lg d$ low bits, we construct a lgd-bit cyclic gray code over the d nodes around each cycle, giving corresponding nodes in different cycles the same label. We then append this to the high bits to obtain a label for each node. Since any two nodes in different cycles differ in their labels' high bits, and any two nodes in the same cycle differ in their labels' low bits, these labels are unique.

Now consider the edges in the cube-connected-cycles network. There are two kinds of edges: the ones that make up the cycles, and the ones connecting corresponding nodes of adjacent cycles in the hybercube. We used a gray cycle within each cycle, so we know that exactly one low bit will flip across each cycle edge. Since every node in the same cycle has the same high bits, this means that crossing a cycle edge flips exactly one bit in the labels.

Similarly, we labelled the corresponding node in each cycle with the same low bits, so crossing the hypercube edges does not change the low bits. Since the high bits were constructed from the hypercube structure, we know crossing a hypercube edge will flip one high bit, so again cossing the edge flips exactly one it in the whole label.

Since crossing any edge flips exactly one bit in the $d \lg d$ bit label, each of those edges is also an edge of a $d \lg d$-bit hypercube. Thus, if we map each node in the $d2^d$-node cube-connected-cycle network to the node in the d-dimensional hypercube with the same label, we will have a complete embedding of the cube-connected-cycle into the hypercube.

Problem 7-2. Show that the bisection width of an N-node butterfly network is $\Theta(N/\lg N)$. (*Hint:* Use the technique of embedding a complete graph in the network.)

Solution: As the hint suggests, we start by embedding a complete graph in the network. That is, we find a path from every node to every other node in the network. We then count the maximum number of such paths crossing each edge, and use that to determine the minimum number of edges needing to be cut to bisect the network.

To start with, we assume that the network wraps around, i.e. that the rightmost column consists of the same nodes as the leftmost column. Note that in the butterfly network, there are only two distinct kinds of edges: horizontal ones and diagonal ones. The network can be redrawn so that any edge of one kind can be put in the place of any other edge of the same kind, so we can consider all edges of the same kind to be "equivalent" for the purposes of symmetry arguments. Note that we can even exchange all diagonal edges with all horizontal edges, so we have symmetry between all edges as well. As usual, we let the network have $N = n \lg n$ nodes.

We now choose the paths to use for the embedded complete graph. We know that any node in any column can reach any node in the same column only by going from left to right and wrapping around once. For the path, then, we go all the way from the start node to the to the node in the proper row in the same column, and then from that column to the right to the destination node. If the source node is in column i and the destination is in column j, then this path crosses a total of $\lg n$ edges of both kinds, plus $j - i$ more horizontal edges.

Summed over all N^2 paths, this gives $N^2 \lg n$ edges evenly distributed over both kinds of edges. By the symmetry argument, then, since there are $2N$ total edges in the network, there are $\frac{1}{2}N \lg n$ paths crossing each edge in the main left-to-right traversal. Similarly, the average number of additional horizonatal edges on each path is $\frac{1}{2}\lg n$, so there are a total of $\frac{1}{2}N^2 \lg n$ additional horizontal edges, for a total of $\frac{1}{2}N \lg n$ additional paths crossing each horizontal edge (by symmetry).

This means that at most $N \lg n$ paths cross any edge in the network. We know that a bisection of a N-node complete graph must break $N^2/2$ edges (those from one half to the other half), so the bisection of the butterfly network must cut edges containing a total of at least $N^2/2$ paths. This means at least $\frac{N^2}{2}/N \lg n = N/2 \lg n = \Omega(N/\lg N)$ edges must be cut for a bisection.

Additionally, we can upper-bound the bisection width to $O(N/\lg N)$ with an example: cut horizontally across the middle of the network, passing through the $n = N/\lg n = \Theta(N/\lg N)$ edges of the major cycle and no other edges. This shoes the bisection width is exactly $\Theta(N/\lg N)$.

(The argument still holds if we do not allow wraparound, since we can get the same effect with either $\lg n$ edges per path going from the right side to the left when necessar. This adds $O(N \lg n)$ paths per edge, so we still have $\Theta(N \lg n)$ paths per edge.)

Problem 7-3. Prove that an n-input Beneš network can simulate any n-node, degree-d network off-line in $O(d \lg n)$ time. (*Hint:* Show that any graph of degree-d can be edge-colored with $d+1$ colors.)

Solution:

Lemma 1*Any graph of degree d can be edge-colored with $2d-1$ colors.*

Proof. Arbitrarily choose an edge, and arbitrarily choose a color for it with does not conflict with any previously colored adjacent edges. Repeat until every edge has been colored. There will always be a valid choice because every node has degree d, and every edge touches 2 nodes, so every edges is adjacent to at most $2d-2$ other edges. Thus, regardless of previous choices, at most $2d-2$ colors woulc cause a conflict, so at least one color is acceptable.

Now, given any n-node, degree-d network, we edge-color it using $2d-1$ colors. Choose one color, say, red. By the definition of an edge-coloring, each node has at most one incident red edge. So in the subgraph consisting of only red edges, each node needs to communicate with at most one other node. This is just a subpermutation, and we saw in class that a n-input Benes network can route any permutation (and thus any subpermutation) of its n inputs offline in $O(\lg n)$ time.

We repeat this process for all $2d-1$ colors of edge, doing the subpermutation corresponding to each color one at a time, and taking $O(d \lg n)$ time total. After all this has been done, every edge in the original network has been simulated, so one step of the entire network has been simulated. Thus we have simulated the original network in $O(d \lg n)$ time.

Problem 7-4. * Prove that an N-node linear array can be embedded with dilation 3 into any N-node connected graph. **Solution:** Given any N-node connected graph, we construct a spanning tree on that graph. We then number

the nodes with a "mixedorder"-traversal of that tree, as follows:

Algorithm 1*Begin by labelling the root and traverse downwards, visiting all of a node's children before returning to that node's parent. Each time we go down the tree to first visit a node of even depth, label that node. Each time we go up the tree from a node of odd depth after labelling all its children, label that node. (Label nodes with ascending integers.)*

We claim that a N-node linear array can be embedded in the tree (and thus in the original graph) by placing the first node at the node labelled 1, the second node at the node labelled 2, and so on, with communication occuring in both directions along the edges of the spanning tree, but with only one message needing to cross in each direction at any time. This will have dilation at most 3, i.e. it never takes more than 3 steps to get from one node to the next node in the sequence.

This is equivalent to saying that, during the traversal which labelled the nodes, at no time were three consecutive edges crossed without labelling any nodes. We consider all the cases of three consecutive edge crossings, and show that the destination node of at at least one edge crossing is labelled. (The edge we leave with the first crossing may or may not be labelled as well.)

Up, Up, Up: After leaving the first node, goes up from two consecutive levels after labelling all children. One of those is at odd depth and thus will be labelled.

Up, Up, Down: The first up and the last down reach the same depth. If that depth is odd, a node will be labelled when the second up leaves it; if it is even, a node will be labelled when the down reaches it.

Up, Down, Up: Down can only be followed by up when a leaf was reached, and leaves are labelled as they are reached.

Up, Down, Down: The two consecutive downs reach two consecutive depths. One of those depths must be even, and thus the node at that depth will be labelled as it is reached.

Down, Up, Up: Again, down followed by up indicates a leaf.

Down, Up, Down: Another leaf.

Down, Down, Up: Another leaf.

Down, Down, Down: At least one of the consecutive downs results in a label.