6.042/18.062J Mathematics for Computer Science October 3, 2006
Tom Leighton and Marten van Dijk

Problem Set 5 Solutions

Due: Wednesday, October 11 at 8pm

Problem 1. [[**points**] 5 points] Prove that every tree with more than one node has at least 2 leaves.

Solution. Similarly to the proof for one leaf given in lecture, let p be the longest path in the tree. Since the tree is connected, and has at least 2 nodes, p contains at least two nodes. Since the tree is acyclic, and p is the longest path, the two nodes at the beginning and end of p must be of degree 1. Therefore, there are at least two nodes of degree 1, i.e. leaves. ■

Problem 2. [[**points**] 10 points] Prove that any connected, n-node graph G with $n - 1$ edges is a tree.

Solution. To show that G is a tree, given that it is connected, we merely need to show that it is acyclic. We can prove this by contradiction. Suppose that there is a cycle in G. Then we can remove an edge from this cycle and G will still be connected. Continue this process until G no longer contains a cycle. Then we are left with a connected, n node, acyclic graph, with $< n - 1$ edges. But we proved in lecture that a connected, n node, acyclic graph (which is a tree), has exactly $n - 1$ edges, thus there is a contradiction, G must be acyclic, and is therefore a tree. ■

Problem 3. [[**points**] 10 points] Complete binary trees with N inputs and N outputs, where $N = 2^n$ for some $n \geq 0$, were described in class and the notes. In this problem we consider complete *ternary* trees with N inputs and N outputs, where $N = 3^n$. The following figure shows a ternary tree with 3 inputs and 3 outputs (i.e $N = 3$, and $n = 1$).

(a) [F pts]ind a closed-form expression for the diameter of the N-input, N-output ternary tree.

Solution. The diameter is $2 \log_3 N + 2$, or $2n + 2$. ■

(b) [P pts]rove that your expression is correct using induction. *Hint:* Your induction hypothesis should prove an expression for the length of a path from an input to the root node.

Solution. We proceed by induction on n. Let $P(n)$ be the proposition that, in a complete ternary tree with $N = 3^n$ inputs and outputs:

- the number of edges in the path from any input to the root node, or from the root node to an output is $n + 1$, and
- the diameter in the tree is $2n + 2$.

Base case ($n = 0$): A complete ternary tree with $N = 3^0 = 1$ input and 1 output consists of those two input/output nodes plus a single switch. The only path from the input to the root traverses 1 edge, which equals $n + 1 = 0 + 1$. Similarly, the only path from the root to the output traverses 1 edge. The only path from the input to the output traverses 2 edges, which equals $2n + 2 = 0 + 2$. Thus $P(0)$ is true.

Inductive step. Now assume $P(n)$ for $n \geq 0$ in order to prove $P(n + 1)$. A complete ternary tree T_0 with 3^{n+1} inputs and 3^{n+1} outputs is constructed from three complete ternary trees T_1, T_2, T_3 with 3^n inputs and 3^n outputs by connecting their root nodes to a single new root node r_0.

By the inductive hypothesis, the length of a path from an input in T_1 to its root r_1 is $n+1$. This root node is connected to the new root r_0 by two edges (r_0, r_1) and (r_1, r_0). Thus the path from an input in T_1 to r_0 traverses $(n+1)+1$ edges, as required. This same argument applies to the trees T_2 and T_3.

If an input and an output belong to T_1 then, by the inductive hypothesis, the path between them traverses $2n + 2$ edges. For an input and output in different subtrees, say T_1 and T_2, the shortest path between them consists of the path from the input to r_1, the edges (r_1, r_0) and (r_0, r_2), and the path from r_2 to the output. This path traverses $(n+1)+2+(n+1) = 2(n+1)+2$ edges, as required. Thus the maximum distance from any input to any output is $2(n + 1) + 2$. Since such a path between nodes in different subtrees will always exist, as the tree is connected, this will be the minimum diameter as well, and so the diameter must be exactly $2(n + 1) + 2$. This proves $P(n + 1)$.

By the principle of induction $P(n)$ is true for all $n \geq 0$. ■

Problem 4. [[**points**] 20 points] Let B_n denote the butterfly network with $N = 2^n$ inputs and N outputs, as defined in lecture. Show that the congestion of B_n is exactly $\sqrt{N}$ when n is even.

Hints:

- For the butterfly network, there is a unique path from each input to each output, so the congestion is the maximum number of messages passing through a vertex for any matching of inputs to outputs.
- If v is a vertex at level i of the butterfly network, there is a path from exactly 2^i input vertices to v and a path from v to exactly 2^{n-i} output vertices.

- At which level of the butterfly network must the congestion be worst? What is the congestion at the node whose binary representation is all 0s at that level of the network?

Solution. First we will show that the congestion is at most $\sqrt{N}$.

Let v be an arbitrary vertex at some level i. Let S_v be the set of inputs that can reach vertex v. Let T_v be the set of outputs that are reachable from vertex v.

By the hint, we have $|S_v| = 2^i$ and $|T_v| = 2^{n-i}$. The number of inputs in S_v that are matched with outputs in T_v is at most $\min\{2^i, 2^{n-i}\}$. To obtain an upper-bound on the congestion of the network, we need to find the maximum value of $\min\{2^i, 2^{n-i}\}$, where the maximum is taken over all i. The maximum value is achieved when 2^i and 2^{n-i} are as equal as possible. Since n is even, these two quantities are equal when $i = n/2$, hence the maximum congestion is

$$2^{n/2} = N^{1/2} = \sqrt{N}.$$

Now we need to show that the congestion achieves $\sqrt{N}$ somewhere in the network. We concluded that the congestion of $\sqrt{N}$ can be achieved only at a node at level $\frac{n}{2}$. Consider the node at that level whose binary representation is all 0s. Any packet from the input in the form $z\underbrace{0\ldots000}_{n/2\ bits}$ with destination $\underbrace{000\ldots0}_{n/2\ bits}z'$, where z and z' are any $\frac{n}{2}$-bit numbers, must pass through this node. But there are $2^{n/2} = \sqrt{N}$ of them, giving the node load $\sqrt{N}$. Therefore, we can conclude that the congestion of B_n is exactly $\sqrt{N}$ when n is even.

■

Problem 5. [[**points**] 25 points] Two students from Podunk University have a neat idea with which they intend to beat out all of the top search engines! Their new product, based on a simple web search algorithm called *Doodle*, uses the following ranking algorithm:

$$\mathit{Doodlerank}(x) = \sum_{y \to x} \mathit{Doodlerank}(y)$$

(the Doodleranks are required to be greater than or equal to 0). This is much nicer than Pagerank, since it gets rid of that silly weighting scheme!

(a) [D pts]escribe the set of possible settings of Doodlerank's for the nodes in the following graphs.

1. The directed path of length n.

   Solution. The first node in the path must get Doodlerank 0, and so, by induction on the number of nodes along the path, all nodes must get Doodlerank 0. ■

2. The directed cycle of length n.

   Solution. All nodes must get the same Doodlerank. ■

(b) [S pts]how that there are graphs in which each node can reach any other node, but for which the only way to assign weights so that the Doodlerank equations are satisfied is so that the Doodlerank weights are all zero!

Solution. An example of such a graph:

If either x_1 or x_3 has Doodlerank strictly greater than 0, then x_2 must have Doodlerank strictly greater than 0. Then x_1 and x_3 must both have Doodlerank strictly greater than 0. So, x_2, which has Doodlerank $x_1 + x_3$ has Doodlerank strictly greater than that of x_1 or x_3. But, the Doodlerank of x_1 must be equal to that that of x_2, and similarly, the Doodlerank of x_3 must be equal to that that of x_2, so we have a contradiction. ■

(c) [O pts]k, the Podunk students are finally convinced that they have to use the weighting scheme from Google – that is, the equations must satisfy

$$Doodlerank(x) = \sum_{y \to x} \frac{Doodlerank(y)}{outdegree(y)}$$

However, the Podunk students want to make their fortune by skipping the modification of the original graph, so that the sinks are not made to point to any new universal nodes as in Pagerank.

Show that their scheme has a major problem. First show that if any node y is assigned Doodlerank 0, then any node x such that x can reach y in a directed walk must also be assigned Doodlerank 0.

Solution. By induction on the length of the path. Let $P(n)$ be the predicate that any node that can reach a 0 Doodlerank node in $\leq n$ steps must have 0 Doodlerank. For the base case, $n = 0$, the node has 0 Doodlerank and we are done. Assume $P(n)$ is true, let's try to prove $P(n+1)$. Let x be any node that can reach the 0 Doodlerank node in i steps where $i \leq n+1$. Let d be the Doodlerank of x. If $i \leq n$, then we know that d must be 0 by the induction hypothesis. If $i = n+1$, then let y be the first node on the shortest path from x to the 0 Doodlerank node. Since y can reach the 0 Doodlerank node in n steps, we know, by the induction hypothesis, that its Doodlerank must be 0. On the other hand, x contributes $d/outdegree(x) > 0$ to the Doodlerank of y. The only way this could be the case is if $d = 0$. Thus, $P(n+1)$ is true and we have shown that any node which can reach a 0 Doodlerank node must be assigned a Doodlerank of 0. ■

(d) [I pts]n the next set of problem parts we are going to show that any sink must be assigned Doodlerank 0. To do this, we are going to remember the view of the Doodlerank equations as describing votes by nodes for each of their neighbors.

1. First show that any set of Doodleranks that satisfy the equations must satisfy "what goes in must come out" – that is, the Doodlerank of a node x must equal the sum of his votes for (or Doodlerank contribution to) each of his neighbors.

   Solution. $Doodlerank(x) = \frac{Doodlerank(x)}{outdegree(x)} \cdot outdegree(x)$ ■

2. Next show the same for any set of nodes S – that is, show that the sum of the Doodleranks of the nodes in S is equal to the weighted sum of the votes of the nodes in S applied towards nodes in V.

   Solution. $\sum_{x \in S} Doodlerank(x) = \sum_{x \in S} \frac{Doodlerank(x)}{outdegree(x)} \cdot outdegree(x)$
   $= \sum_{\{x \to y\} | x \in S, y \in V\}} \frac{Doodlerank(x)}{outdegree(x)}$ ■

3. Finally, show that any sink must be assigned Doodlerank 0. *Hint: Let S be the set of sinks, and $T = V - S$ the set of nonsinks. Write $\sum_{x \in T} Doodlerank(x)$ in terms of the weighted sum of the votes of the nodes in the graph for nodes in T and then in terms of the weighted sum of the votes in the whole graph. Use this to show that the weighted sum of the votes for nodes in S must be 0.*

   Solution. On one hand, we have that

$$\begin{aligned}\sum_{x \in T} Doodlerank(x) &= \sum_{x \in T} \sum_{\{y \to x | y \in V\}} \frac{Doodlerank(y)}{outdegree(y)} \\ &= \sum_{\{y \to x | x \in T, y \in V\}} \frac{Doodlerank(y)}{outdegree(y)}\end{aligned}$$

   where the first equality is just by the Doodlerank equations.
   On the other hand, we have that

$$\begin{aligned}\sum_{x \in T} Doodlerank(x) &= \sum_{\{x \to y | x \in T, y \in V\}} \frac{Doodlerank(x)}{outdegree(x)} \\ &= \sum_{\{x \to y | x \in V, y \in V\}} \frac{Doodlerank(x)}{outdegree(x)} \\ &= \sum_{\{y \to x | y \in V, x \in V\}} \frac{Doodlerank(y)}{outdegree(y)}\end{aligned}$$

where the second equality follows since $\{x \to y | x \in T, y \in V\}$ describes the set of all edges in the graph, as there are no edges directed out of S. The third equality follows by a simple change of variables.

So, putting these two together,

$$\sum_{\{y \to x | y \in V, x \in V\}} \frac{Doodlerank(y)}{outdegree(y)} = \sum_{\{y \to x | x \in T, y \in V\}} \frac{Doodlerank(y)}{outdegree(y)}$$

But,

$$\sum_{\{y \to x | y \in V, x \in V\}} \frac{Doodlerank(y)}{outdegree(y)} = \sum_{\{y \to x | x \in T, y \in V\}} \frac{Doodlerank(y)}{outdegree(y)} + \sum_{\{y \to x | x \in S, y \in V\}} \frac{Doodlerank(y)}{outdegree(y)}$$

so it must be the case that

$$\sum_{\{y \to x | x \in S, y \in V\}} \frac{Doodlerank(y)}{outdegree(y)} = 0$$

Since all the Doodleranks are positive, this means that for each sink x, any node y that has an edge to x must have Doodlerank 0. But then, by the Doodlerank equations, the Doodlerank of x must also be 0.

■

4. Finally, conclude that any node which can reach a sink must also be assigned a Doodlerank of 0. So it's not too likely that Podunk U. is going to be hitting up these students for contributions anytime soon!

   Solution. We have shown that sinks must be assigned Doodlerank 0, and that any node which can reach a Doodlerank 0 node must also be assigned Doodlerank 0. Note that we just did a proof by induction in which the base case was harder than the inductive step! ■

Problem 6. [[**points**] 20 points]

In "Die Hard: The Afterlife", the ghosts of Bruce and Sam have been sent by the evil Simon on another mission to save midtown Manhattan. They have been told that there is a bomb on a street corner that lies in Midtown Manhattan, which Simon defines as extending from 41st Street to 59th Street and from 3rd Avenue to 9th Avenue. Additionally, the code that they need to defuse the bomb is on another street corner. Simon, in a good mood, also tosses them two carrots:

- He will have a helicopter initially lower them to the street corner where the bomb is.
- He promises that the code is placed only on a corner of a numbered street and a numbered avenue, so they don't have to search Broadway.

The map of midtown Manhattan is an example of an $N \times M$ (undirected) grid. In particular, midtown Manhattan is a 19×7 grid.

Bruce and Sam need to check all $19 \cdot 7 = 133$ street corners for the code. Once they are at a corner, they don't need any additional time to verify if the code is there. Once they find the code and return to the bomb, they can disarm it in 2 minutes (even, or especially, as the timer ticks down to 0). Also, they can run one block (in any of the four directions) in exactly 1 minute. They are given 135 minutes total in which to find the code and disarm the bomb, which means that they need to return to the bomb, code in hand, in 133 minutes.

Sam realizes that the map of NYC is actually a graph, and that they need to use a cool new 6.042 concept: A *Hamiltonian cycle* is a path that visits each vertex in a graph exactly once and ends at its starting point (so it is a cycle). A graph is *Hamiltonian* if it has a Hamiltonian cycle.

Hamiltonian graphs are really useful because you can visit each node and return to the starting point by taking only n steps, where n is the number of nodes – if a graph is not Hamiltonian, you would need at least $n + 1$ steps to visit each of the n nodes and return to the starting point.

In general, we don't know how to efficiently determine whether a general graph is Hamiltonian or not. However, Sam is very excited because he thinks that he can show that Midtown Manhattan is Hamiltonian. If it is, Bruce and Sam can save the day! Will they make it?

(a) [S pts]how that they cannot do it – that is, more generally, show that if both N and M are odd, then the $N \times M$ grid is *not* Hamiltonian. *Hint: First show that any $N \times M$ 2-dimensional undirected grid is bipartite.*

Solution. As per the hint, let us first show that any 2-dimensional grid is bipartite. For this, let us exhibit a coloring with 2 colors $\{0, 1\}$. Indexing the vertices of the grid by their (x, y)-coordinates, color vertex (i, j) with color $rem(i + j, 2)$. It is easy to see that this is a valid 2-coloring.

Suppose the graph is Hamiltonian. Now, since N, M are both odd, there are an odd number of vertices in the graph. Thus the hamiltonian cycle in this graph is an odd cycle. However, since the graph is bipartite, this graph has no odd cycles. This is a contradiction: thus our supposition that the graph was Hamiltonian is wrong, and we are done. ■

(b) [S pts]uppose Simon defined Midtown in the more standard way as extending from 40th Street to 59th Street and from 3rd Avenue to 9th Avenue (that is suppose Midtown Manhattan was a 20×7 grid), and gave them another 7 minutes,

1. Show that if either N or M is even, then the $N \times M$ grid is Hamiltonian. *Hint: assuming N is even (without loss of generality) use induction on N.*

   Solution. Suppose N is even (wlog).

   Assume the grid is laid out on the plane occupying the integer points between $(0, 0)$ and $(N - 1, M - 1)$.

We will give the hamiltonian path explicitly by specifying the k'th vertex visited for each k from 0 to NM. Let $q = \lfloor \frac{k}{M-1} \rfloor$ and let $r = rem(k, M-1)$.

On step k:

- if $k \leq N(M-1)$ and q is even, then visit vertex $(q, r+1)$.
- if $k \leq N(M-1)$ and q is odd, then visit vertex $(q, M-1-r)$.
- if $k > N(M-1)$, then visit vertex $(0, N-(k-N(M-1)))$.

Checking that it is a Hamiltonian cycle is routine.

Drawing the above path on an actual grid is very enlightening.

■

2. Explain why your proof breaks down when N and M are odd.

   Solution. The odd/even conditions on q determine require N to be even for the above sequence of vertices to actually give a cycle. ■

3. Would they survive? Does it depend on where the bomb is placed?

   Solution. Come on, of course! No it doesn't depend on where the bomb is placed. ■

Problem 7. [[**points**] 10 points] Our friends at Podunk University, after their failure with *Doodle*, have instead decided to find fame and fortune by solving a different problem. They now claim that they have discovered a *greedy algorithm* that solves the problem of finding minimum-weight perfect matchings in (undirected) graphs where a perfect matching exists. The greedy algorithm is as follows:

1. Add to the matching the minimum-weight edge in the graph which is not incident to the same node as any of the edges already in the matching.

2. Continue until there are $|V|/2$ edges in the matching, where $|V|$ is the number of nodes in the graph.

Show that this algorithm doesn't work by finding a counterexample, i.e. a graph where there is a perfect matching, but this algorithm fails to find the minimum-weight perfect matching.

Solution. One possible counterexample:

pset5-counterexample.jpg

In this graph, the algorithm will select the edge $\{x_1, x_2\}$ first, and then be stuck with the edge $\{x_3, x_4\}$. The minimum-weight perfect matching is, of course, $\{x_1, x_4\}$ and $\{x_2, x_3\}$.

■