

Problem Set 5 Solutions

Due: Monday, October 7

Problem 1. [20 points]

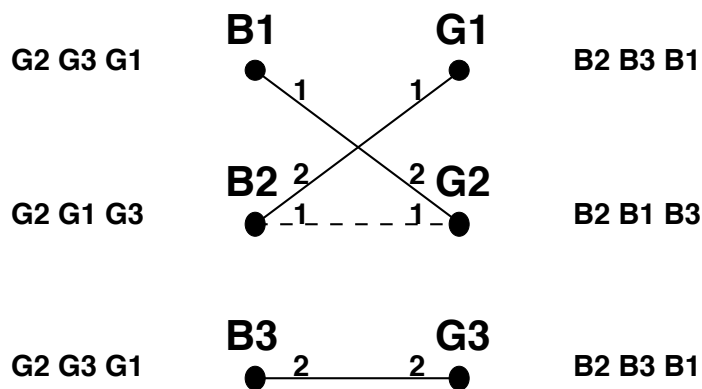
In lecture we discussed the notion of pairing up boys and girls by finding a *minimum weight matching* on the bipartite graph, G , where the weight of an edge $b-g$ is the sum of the rank of the g on b 's list plus the rank of b on g 's list. The minimum weight matching of G is the matching that produces the lowest sum of weights on the edges of G .

Ex. If boy b_1 has a ranking of girls g_1, g_2, g_3, g_4 , and girl g_1 has a ranking of boys b_2, b_3, b_4, b_1 , then the weight of the edge b_1-g_1 will be $1 + 4 = 5$, since g_1 is ranked first and b_1 is ranked fourth.

(a) [10 pts] Prove that the minimum weight matching is not always a stable matching by providing a counterexample.

(Hint: There is a counterexample with 3 boys and 3 girls.)

Solution. Consider the counterexample shown below. The minimum weight matching gives a sum of weights $3 + 3 + 4 = 10$. However, there is a rogue couple between B2 and G2 (shown in the dotted line), who prefer each other to their mates.



(b) [10 pts] The minimum weight matching minimizes the sum over *all* possible matchings. Instead, consider a greedy algorithm that recursively matches the minimum weight edge over all unmatched nodes. Prove that this matching is also not always a stable matching by providing a counterexample.

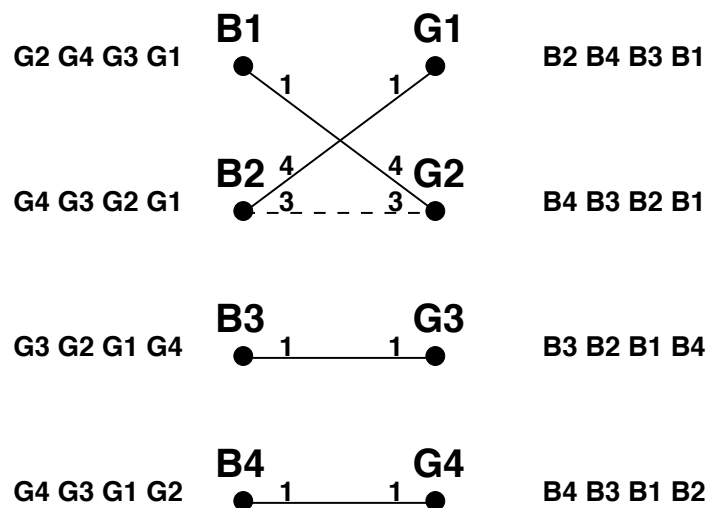
Ex. If all boys have the same ranking of girls g_1, g_2, g_3, g_4 , and all girls have the same ranking of boys b_1, b_2, b_3, b_4 , then b_1-g_1 will be matched first (weight= 2), b_2-g_2 will be matched second (weight= 4), b_3-g_3 will be matched third (weight= 6), and b_4-g_4 will be matched last (weight= 8).

Note two things:

- Suppose that, in the case of a tie for minimum weight edge, the algorithm matches all of the tied edges. If this creates a conflict (for example, if g_1-b_1 and g_1-b_2 both have weight= 3), then the algorithm just matches one of the pairs randomly (for example, g_1-b_1). Choose a counterexample with no such conflict.
- The algorithm does *not* recalculate the weights after each recursion, so the weight of an edge does not change, even as higher ranked preferences become unavailable. There is also a counterexample to stability for such an algorithm that does reweight the edges, but it's much more difficult to find!

(Hint: There is a counterexample with 4 boys and 4 girls.)

Solution. Consider the counterexample shown below. The first round of the algorithm removes the edges B3-G3 and B4-G4, which both have weights of 2. The second round pairs up B1-G2 and B2-G1, which each have weight of 5, less than the weight 8 of B1-G1 and weight 6 of B2-G2. However, there is a rogue couple between B2 and G2 (shown in the dotted line), who prefer each other to their mates.



■

Problem 2. [15 points]

(a) [5 pts] Suppose that G is a simple, connected graph on n nodes. Show that G has exactly $n - 1$ edges *iff* G is a tree.

Solution. To show the biimplication, it is necessary to show both that if G is a tree then it has $n - 1$ edges *and also* if G has $n - 1$ edges then it is a tree. The first part was proved in lecture by induction on the number of nodes. We prove the second part by contradiction.

To show that a connected graph G with $n - 1$ edges is a tree, it is sufficient to show it is acyclic. Assume to the contrary that there is a cycle in G . We can remove an edge from any cycle preserving connectivity, so remove edges from G until it no longer contains a cycle, forming a connected acyclic graph G' . G' is by definition a tree, but has fewer than $n - 1$ edges since we removed at least one edge from G . This contradicts the proof that all trees on n nodes have exactly $n - 1$ edges. G therefore is acyclic and thus a tree, completing the proof of the biimplication. ■

(b) [10 pts] Prove by induction that any connected graph has a spanning tree.

Solution. The proof is by induction on the number of edges. Let $P(k)$ be the predicate that if G is connected with $k \geq n - 1$ edges, then G has a spanning tree.

Base Case: $k = n - 1$, part (a) demonstrates that G is a tree and thus a spanning tree of itself.

Inductive Step: Assume $P(k)$. If G is a connected graph with $k + 1 > n - 1$ edges, then it must not be a tree by part (a). Thus, it must have a cycle. Removing an edge from that cycle creates a connected graph G' with k edges, which has a spanning tree over the nodes by our inductive hypothesis. This spanning tree is also a spanning tree over G , thus $P(k + 1)$ holds.

By induction, a connected graph G with k edges has a spanning tree for all $k \geq n - 1$. ■

Problem 3. [20 points] An n -node graph is said to be *tangled* if there is an edge leaving every set of $\lceil \frac{n}{3} \rceil$ or fewer vertices. As a special case, the graph consisting of a single node is considered tangled. (Recall that the notation $\lceil x \rceil$ refers to the smallest integer greater than or equal to x .)

(a) [7 pts] Find the error in the proof of the following claim.

Claim. Every non-empty, tangled graph is connected.

Proof. The proof is by strong induction on the number of vertices in the graph. Let $P(n)$ be the proposition that if an n -node graph is tangled, then it is connected. In the base case, $P(1)$ is true because the graph consisting of a single node is defined to be tangled and is trivially connected.

In the inductive step, for $n \geq 1$ assume $P(1), \dots, P(n)$ to prove $P(n+1)$. That is, we want to prove that if an $(n+1)$ -node graph is tangled, then it is connected. Let G be a tangled, $(n+1)$ -node graph. Arbitrarily partition G into two pieces so that the first piece contains exactly $\lceil \frac{n}{3} \rceil$ vertices, and the second piece contains all remaining vertices. Note that since $n \geq 1$, the graph G has at least two vertices, and so both pieces contain at least one vertex. By induction, each of these two pieces is connected. Since the graph G is tangled, there is an edge leaving the first piece, joining it to the second piece. Therefore, the entire graph is connected. This shows that $P(1), \dots, P(n)$ imply $P(n+1)$, and the claim is proved by strong induction. \square

Solution. The error is in the line: “By induction, each of these two pieces is connected.”

Our induction hypothesis states, “if a graph is tangled, then it is connected.” Our assumption $P(1), \dots, P(n)$ allows us to say that if each of the two pieces with $< n+1$ nodes were tangled, then each piece would also be connected. However, we only know that the original graph G with $n+1$ nodes is tangled, which says nothing about whether the subgraphs of G are tangled. Therefore, we cannot use the left side of the implication to prove the right side of the implication, and hence we cannot conclude that the subgraphs are connected.

In abstract terms, the error lies here: we are proving an induction hypothesis of the form “if (this), then (that)”, but we erroneously use the induction hypothesis as if it were simply of the form “(that).” \blacksquare

(b) [5 pts] Draw a tangled graph that is not connected.

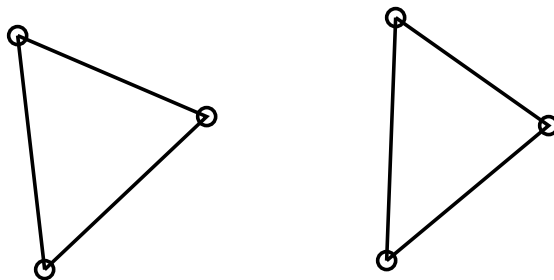


Figure 1: Tangled but non-connected graph

Solution. \blacksquare

(c) [8 pts] An n -node graph is said to be *mangled* if there is an edge leaving every set of $\lceil \frac{n}{2} \rceil$ or fewer vertices. Again, as a special case, the graph consisting of a single node is considered mangled. Prove the following claim. *Hint: Prove by contradiction.*

Claim. Every non-empty, mangled graph is connected.

Solution. The proof is by contradiction. Assume for the purpose of contradiction that there exists an n -node graph that is mangled, but not connected. Then the graph must

have at least two connected components. However, there can be at most one connected component with more than $\lceil \frac{n}{2} \rceil$ vertices, since the graph has only n vertices in total. Therefore, there exists a connected component with $\lceil \frac{n}{2} \rceil$ or fewer vertices. Since the graph is mangled, there is an edge leaving this component. But this contradicts the definition of a connected component. ■

Problem 4. [10 points] Let the nodes in a tournament be ranked according to their outdegrees, that is, node u is ranked less than or equal to v iff $outdegree(u) \leq outdegree(v)$. Prove that the sum of the outdegrees of the i lowest ranked nodes is at least $i(i - 1)/2$.

Solution. Consider the subgraph on the i lowest ranked nodes. This represents a tournament as well. So, the number of its edges is equal to $i(i - 1)/2$. Each of these edges are counted in the sum of the outdegrees of these i nodes in the original graph. ■

Problem 5. [10 points] A directed graph is *symmetric* if, whenever $x \rightarrow y$ is an edge, so is $y \rightarrow x$.

Given any finite, symmetric web graph, let

$$PR(x) ::= \frac{out-degree(x)}{e},$$

where e is the total number of edges in the graph. Show that this is a solution for the system of equations that are satisfied by the page rank as discussed in section 2 of lecture notes 9.

Solution. We need to prove that $PR(x) = outdegree(x)/e$ satisfies the system of equations

$$\forall x, PR(x) = \sum_{y \text{ s.t. } y \rightarrow x} \frac{PR(y)}{outdegree(y)}$$

together with

$$\sum_x PR(x) = 1.$$

We derive

$$\begin{aligned} \sum_{y \text{ s.t. } y \rightarrow x} \frac{PR(y)}{outdegree(y)} &= \sum_{y \text{ s.t. } y \rightarrow x} \left(\frac{1}{outdegree(y)} \right) \left(\frac{out-degree(y)}{e} \right) \\ &= \sum_{y \text{ s.t. } y \rightarrow x} \frac{1}{e} \\ &= \frac{in-degree(x)}{e} \\ &= \frac{out-degree(x)}{e} \text{ (by symmetry of the graph)} \\ &= PR(x) \end{aligned}$$

and

$$\sum_x PR(x) = \sum_x \frac{out-degree(x)}{e} = \frac{e}{e} = 1.$$

■

Problem 6. [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:

$$Doodlerank(x) = \sum_{y \rightarrow x} Doodlerank(y)$$

(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) [5 pts] Describe the set of possible settings of Doodlerank's for the nodes in the following graphs.

1. The directed path of length n .

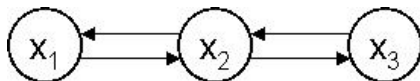
Solution. The first node in the path must get Doodlerank 0, and so, by induction on the number of nodes along the path, all nodes must get Doodlerank 0. ■

2. The directed cycle of length n .

Solution. All nodes must get the same Doodlerank. ■

(b) [5 pts] Give an example of a graph 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:



By the definition of the Doodlerank, we get the following equations:

$$Doodlerank(x_1) = Doodlerank(x_2)$$

$$Doodlerank(x_2) = Doodlerank(x_1) + Doodlerank(x_3)$$

$$Doodlerank(x_3) = Doodlerank(x_2)$$

Substituting the equations, we get $Doodlerank(x_2) = 2 \times Doodlerank(x_2)$, so all of the Doodleranks must be equal to 0. ■

Ok, the Podunk students are finally convinced that they have to use the weighting scheme from Google – that is, the equations must satisfy

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

However, the Podunk students want to make their fortune by skipping any modification of the original graph. Unlike Pagerank, the sinks in Doodlerank are not made to point to any universal supernodes and maintain an outdegree of 0.

This scheme has also a major problem!

(c) [5 pts] First show by induction that if any node x is assigned Doodlerank 0, then any node y that can reach x in a directed walk must also be assigned Doodlerank 0.

Solution. *Proof.* The proof is by induction on the length of the path. Suppose x is assigned Doodlerank 0. Let $P(n)$ be the predicate that any node that can reach x in a minimum of n steps must have Doodlerank 0.

Base Case: $n = 0$, the node is x and has Doodlerank 0.

Inductive Step: Assume $P(n)$ is true. Let y be any node that can reach x in a minimum of $n+1$ steps, where π is the length $n+1$ shortest path from y to x . Let x' be the first node after y on π . The shortest path from x' to x is length n , since any shorter path would contradict π as the shortest path from y to x . Therefore, we know by the induction hypothesis that x' has Doodlerank 0.

We know y contributes $Doodlerank(y)/outdegree(y)$ to $Doodlerank(x')$, so the only way for $Doodlerank(x') = 0$ is for $Doodlerank(y) = 0$. Thus, $P(n+1)$ is true and, since x was an arbitrary node, we have shown that any node which can reach a node of Doodlerank 0 must also have Doodlerank 0.

□
■

(d) [10 pts] In 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 think of the Doodlerank equations as describing votes by nodes for each of their neighbors. We represent a vote by x for y as a weighted edge $x \rightarrow y$.

1. Suppose V is the set of all the nodes in the graph. Let S be the set of sinks and $T = V - S$ the set of non-sinks. First show that the Doodlerank of a single node $x \in T$ (i.e. $outdegree(x) \geq 1$) must equal the sum of the weights on the outgoing edges of x .

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

2. Next show the same property holds over the entire set of non-sinks T by writing the sum of Doodleranks of nodes in T , $\sum_{x \in T} Doodlerank(x)$, as the sum of the weights of outgoing edges from all nodes $x \in T$ to nodes $y \in V$.

Solution.

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

■

3. Finally, show that any sink must be assigned Doodlerank 0.

Hint: Transform the sum of Doodleranks of non-sink nodes in T , $\sum_{x \in T} \text{Doodlerank}(x)$, into a sum of Doodleranks of all nodes in V , $\sum_{y \in V} \text{Doodlerank}(y)$. Use this to show that the sum of Doodleranks of all sink nodes $z \in S$ must be 0.

Solution.

$$\begin{aligned}
 \sum_{x \in T} \text{Doodlerank}(x) &= \sum_{\{x \rightarrow y \mid x \in T, y \in V\}} \frac{\text{Doodlerank}(x)}{\text{outdegree}(x)} && \text{(from part 2)} \\
 &= \sum_{y \in V} \sum_{\{x \rightarrow y \mid x \in T\}} \frac{\text{Doodlerank}(x)}{\text{outdegree}(x)} && \text{(by summing over } y\text{)} \\
 &= \sum_{y \in V} \sum_{x \rightarrow y} \frac{\text{Doodlerank}(x)}{\text{outdegree}(x)} && \text{(since } x \rightarrow y \Rightarrow x \in T\text{)} \\
 &= \sum_{y \in V} \text{Doodlerank}(y) && \text{(by definition of Doodlerank)}
 \end{aligned}$$

Since $T = V - S$, it must be the case that

$$\sum_{x \in T} \text{Doodlerank}(x) = \sum_{y \in V} \text{Doodlerank}(y) - \sum_{z \in S} \text{Doodlerank}(z)$$

But $\sum_{x \in T} \text{Doodlerank}(x) = \sum_{y \in V} \text{Doodlerank}(y)$, so $\sum_{z \in S} \text{Doodlerank}(z) = 0$.

Since the Doodleranks cannot be negative, it must follow that for every sink z , $\text{Doodlerank}(z) = 0$. ■

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! ■