Announcements

- Reading
  - Start on Chapter 4

Stable Matching Results

- Averages of 5 runs
- Much better for M than W
- Why is it better for M?
- What is the growth of m-rank and w-rank as a function of n?

<table>
<thead>
<tr>
<th>n</th>
<th>m-rank</th>
<th>w-rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>5.102</td>
<td>98.048</td>
</tr>
<tr>
<td>500</td>
<td>7.52</td>
<td>66.952</td>
</tr>
<tr>
<td>500</td>
<td>8.152</td>
<td>79.815</td>
</tr>
<tr>
<td>500</td>
<td>12.72</td>
<td>91.26</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>m-rank</th>
<th>w-rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>500</td>
<td>5.25</td>
<td>90.726</td>
</tr>
<tr>
<td>500</td>
<td>6.5528</td>
<td>77.9552</td>
</tr>
<tr>
<td>500</td>
<td>7.0492</td>
<td>140.4002</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>m-rank</th>
<th>w-rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>6.796</td>
<td>140.035</td>
</tr>
<tr>
<td>1000</td>
<td>7.005</td>
<td>150.712</td>
</tr>
<tr>
<td>1000</td>
<td>7.16</td>
<td>123.005</td>
</tr>
<tr>
<td>1000</td>
<td>7.056</td>
<td>127.869</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>m-rank</th>
<th>w-rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>7.005</td>
<td>140.035</td>
</tr>
<tr>
<td>1000</td>
<td>7.367</td>
<td>261.781</td>
</tr>
<tr>
<td>1000</td>
<td>11.126</td>
<td>174.175</td>
</tr>
<tr>
<td>1000</td>
<td>7.649</td>
<td>174.315</td>
</tr>
<tr>
<td>1000</td>
<td>7.147</td>
<td>262.622</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>n</th>
<th>m-rank</th>
<th>w-rank</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>7.826</td>
<td>257.7955</td>
</tr>
<tr>
<td>2000</td>
<td>7.505</td>
<td>263.781</td>
</tr>
<tr>
<td>2000</td>
<td>11.525</td>
<td>175.175</td>
</tr>
<tr>
<td>2000</td>
<td>7.649</td>
<td>174.315</td>
</tr>
<tr>
<td>2000</td>
<td>7.147</td>
<td>262.622</td>
</tr>
</tbody>
</table>

Graph Theory

- $G = (V, E)$
  - $V$: vertices, $|V| = n$
  - $E$: edges, $|E| = m$
- Undirected graphs
  - Edges sets of two vertices $(u, v)$
- Directed graphs
  - Edges ordered pairs $(u, v)$
- Many other flavors
  - Edge / vertices weights
  - Parallel edges
  - Self loops
- Path: $v_1, v_2, ..., v_k$, with $(v_i, v_{i+1})$ in $E$
  - Simple Path
  - Cycle
  - Simple Cycle
- Neighborhood
  - $N(v)$
- Distance
- Connectivity
  - Undirected
  - Directed (strong connectivity)
- Trees
  - Rooted
  - Unrooted

Last Lecture

- Bipartite Graphs: two-colorable graphs
- Breadth First Search algorithm for testing two-colorability
  - Two-colorable iff no odd length cycle
  - BFS has cross edge iff graph has odd cycle

Graph Search

- Data structure for next vertex to visit determines search order

Graph Search Diagram
Graph search
Breadth First Search
\[ S = \{s\} \]
while S is not empty
\[ u = \text{Dequeue}(S) \]
if u is unvisited
visit u
foreach v in N(u)
Enqueue(S, v)

Depth First Search
\[ S = \{s\} \]
while S is not empty
\[ u = \text{Pop}(S) \]
if u is unvisited
visit u
foreach v in N(u)
Push(S, v)

Breadth First Search
• All edges go between vertices on the same layer or adjacent layers

Depth First Search
• Each edge goes between vertices on the same branch
• No cross edges

Connected Components
• Undirected Graphs

Computing Connected Components in \(O(n+m)\) time
• A search algorithm from a vertex v can find all vertices in v’s component
• While there is an unvisited vertex v, search from v to find a new component

Directed Graphs
• A Strongly Connected Component is a subset of the vertices with paths between every pair of vertices.
Identify the Strongly Connected Components

Strongly connected components can be found in $O(n+m)$ time

- But it’s tricky!
- Simpler problem: given a vertex $v$, compute the vertices in $v$’s scc in $O(n+m)$ time

Topological Sort

- Given a set of tasks with precedence constraints, find a linear order of the tasks

Find a topological order for the following graph

If a graph has a cycle, there is no topological order

- Consider the first vertex on the cycle in the topological sort
- It must have an incoming edge

Lemma: If a graph is acyclic, it has a vertex with in-degree 0

- Proof:
  - Pick a vertex $v_1$, if it has in-degree 0 then done
  - If not, let $(v_2, v_1)$ be an edge, if $v_2$ has in-degree 0 then done
  - If not, let $(v_3, v_2)$ be an edge . . .
  - If this process continues for more than $n$ steps, we have a repeated vertex, so we have a cycle
Topological Sort Algorithm

While there exists a vertex v with in-degree 0
  Output vertex v
  Delete the vertex v and all outgoing edges

Details for O(n+m) implementation

- Maintain a list of vertices of in-degree 0
- Each vertex keeps track of its in-degree
- Update in-degrees and list when edges are removed
- m edge removals at O(1) cost each