CS320 – Test 2 Study Review
You have access to a really cool Fibonacci heap implementation, and your boss is really enthralled with the idea of using it. You are asked to use this for a Kruskal’s implementation.

**QUESTION:** What should you do?

a. Go for it – a Fibonacci heap can most likely help
b. Ask whether the graphs you’ll be dealing with are very dense with lots of edges or not – if they are dense then Fibonacci heaps are ok
c. Ask whether the application will be OK with slightly longer times to come up with “answers” because of amortization
d. Convince your boss that Fibonacci heaps are too complex
e. Convince your boss that actually a Fibonacci heap cannot be used as part of Kruskal’s. Rather, a tree-based disjoint set is better.
You have access to a really cool Fibonacci heap implementation, and your boss is really enthralled with the idea of using it. You are asked to use this for a Kruskal’s implementation.

**QUESTION:** What should you do?

a. Go for it – a Fibonacci heap can most likely help  
b. Ask whether the graphs you’ll be dealing with are very dense with lots of edges or not – it they are dense then Fibonacci heaps are ok  
c. Ask whether the application will be OK with slightly longer times to come up with “answers” because of amortization  
d. Convince your boss that Fibonacci heaps are too complex  
e. Convince your boss that actually a Fibonacci heap cannot be used as part of Kruskal’s. Rather, a tree-based disjoint set is better.
We’re trying to find the least amount of cable to lay a fiber-optic cable network for a neighborhood. All houses should have some connection (possibly through other houses) to the central hub.

**QUESTION:** What kind of an algorithm should we use?
We’re trying to find the least amount of cable to lay a fiber-optic cable network for a neighborhood. All houses should have some connection (possibly through other houses) to the central hub.

**QUESTION:** What kind of an algorithm should we use?

**Minimum Spanning Tree**
We’re trying to find the least amount of cable to lay a fiber-optic cable network for a neighborhood. All houses should have some connection (possibly through other houses) to the central hub.

**QUESTION:** What algorithm choices do we have?
We’re trying to find the least amount of cable to lay a fiber-optic cable network for a neighborhood. All houses should have some connection (possibly through other houses) to the central hub.

**QUESTION:** What algorithm choices do we have?

- **Kruskal’s algorithm**
- **Prim’s algorithm**
We’re going to use model checking to find out if a regulation is violated in a system. We build a state machine model of the system and add the conditions that are not allowed, then run a model checker to see if we ever get into states where the disallowed conditions occur. Model checkers explore every single path in a system looking to see if the disallowed states are ever reached. Model checkers quickly fail with even moderately complex systems due to this state explosion problem. We must reduce the size of the graph that we’re going to input into the model checker, by grouping nodes into conceptual units.

**QUESTION:** What algorithm can potentially help?
We’re going to use model checking to find out if a regulation is violated in a system. We build a state machine model of the system and add the conditions that are not allowed, then run a model checker to see if we ever get into states where the disallowed conditions occur. Model checkers explore every single path in a system looking to see if the disallowed states are ever reached. Model checkers quickly fail with even moderately complex systems due to this state explosion problem. We must reduce the size of the graph that we’re going to input into the model checker, by grouping nodes into conceptual units.

**QUESTION:** What algorithm can potentially help?

*Strongly connected components*
Your co-worker challenges you regarding whether it is correct to choose the shortest edge for Kruskal’s algorithm or the lightest weight edge when changing node weight using Prim’s algorithm. Your argument should be the following. Fill in the blanks using \[ \text{respects, breaks, subset, adding, growing, safe, } \cup, \leq, \geq, \in, \subseteq \]

Assume \( G = (V, E) \) and \( A \) is a \( 1 \) of \( E \) included in some MST, \( T \). \( (S, V-S) \) is a cut that \( 2 \) \( A \), and \( (u,v) \) is a light edge crossing the cut that is not in \( T \). Since \( u \) and \( v \) are on opposite sides of the cut there has to be a path between them, in \( T \), with an edge that crosses the cut. Let \( (x,y) \) be that edge. Since \( (x,y) \) is on the unique simple path (UNIQUE since \( T \) is an MST), then removing \( (x,y) \) \( 3 \) the tree, and \( 4 \) \( (u,v) \) creates a new MST, \( T' \). \( w(T') = w(T) - w(x,y) + w(u,v) \). Further, since \( (u,v) \) is a light edge, then \( w(T') \leq 5 \ w(T) \). Since \( T \) is an MST, then \( T' \) must be one too. \( A \subseteq T' \) since \( A \subseteq T \) (because the cut respected \( A \)) and \( (x,y) \notin A \). So, \( A \) \( 6 \) \( \{(u,v)\} \) \( 7 \) \( T' \) (because we are \( 8 \) \( T' \) by adding an edge to cross the cut). Since \( T' \) is an MST, \( (u,v) \) is \( 9 \) for \( A \).
Your co-worker challenges you regarding whether it is correct to choose the shortest edge for Kruskal’s algorithm or the lightest weight edge when changing node weight using Prim’s algorithm. Your argument should be the following. Fill in the blanks using [respects, breaks, subset, adding, growing, safe, $\cup$, $\leq$, $\geq$, $\in$, $\subseteq$]

Assume $G=(V,E)$ and $A$ is a subset of $E$ included in some MST, $T$. $(S, V-S)$ is a cut that respects $A$, and $(u,v)$ is a light edge crossing the cut that is not in $T$. Since $u$ and $v$ are on opposite sides of the cut there has to be a path between them, in $T$, with an edge that crosses the cut. Let $(x,y)$ be that edge. Since $(x,y)$ is on the unique simple path (UNIQUE since $T$ is an MST), then removing $(x,y)$ breaks the tree, and adding $(u,v)$ creates a new MST, $T'$. $w(T') = w(T) - w(x,y) + w(u,v)$. Further, since $(u,v)$ is a light edge, then $w(T') \leq w(T)$. Since $T$ is an MST, then $T'$ must be one too. $A \subseteq T'$ since $A \subseteq T$ (because the cut respected $A$) and $(x,y) \notin A$. So, $A \cup \{(u,v)\} \subseteq T'$ (because we are growing $T'$ by adding an edge to cross the cut). Since $T'$ is an MST, $(u,v)$ is safe for $A$. 
QUESTION: Subpaths of a shortest path are:

a. Always negative in total edge weight
b. Always less in total edge weight than the parent path
c. Always shortest paths
d. Sometimes shortest paths
QUESTION: Subpaths of a shortest path are:

a. Always negative in total edge weight
b. Always less in total edge weight than the parent path
C. Always shortest paths
d. Sometimes shortest paths
Fill in the blanks:

During Prim’s algorithm, we perform _______________ extract-mins and up to _______________ reduce-keys on the priority queue. These operations dominate the runtime, since building the priority queue is linear for the implementations we have talked about, and the bookkeeping work is also linear.
Fill in the blanks:

During Prim’s algorithm, we perform $|V|$ extract-mins and up to $|E|$ reduce-keys on the priority queue. These operations dominate the runtime, since building the priority queue is linear for the implementations we have talked about, and the bookkeeping work is also linear.
**QUESTION:** Using a binary heap for the priority queue, what is the runtime for Prim’s?
QUESTION: Using a binary heap for the priority queue, what is the runtime for Prim’s?

\[ O(V \log V + E \log V) = O(E \log V) \]
**QUESTION:** Using a flat array for the priority queue gives us $O(1)$ cost per reduce-key and $O(n)$ cost per extract-min. What is the runtime for Prim’s using the flat array?
QUESTION: Using a flat array for the priority queue gives us $O(1)$ cost per reduce-key and $O(n)$ cost per extract-min. What is the runtime for Prim’s using the flat array?

$O(E + V^2) = O(V^2)$
QUESTION: When is this runtime better than the binary heap version?
**QUESTION:** When is this runtime better than the binary heap version?

\[ E \lg V = V^2; \ E = \frac{V^2}{\lg V}, \text{ so when } E > \frac{V^2}{\lg V} \]
QUESTION: What is the greedy step for Kruskal’s algorithm?
QUESTION: What is the greedy step for Kruskal’s algorithm?

Consider the least-weight edge that hasn’t been considered.
**QUESTION:** Which of the following can a BFS do that a DFS cannot?

a. Find connected components of a graph  
b. Determine if a path exists between two nodes  
c. Find the shortest path between two nodes
QUESTION: Which of the following can a BFS do that a DFS cannot?

a. Find connected components of a graph
b. Determine if a path exists between two nodes
C. Find the shortest path between two nodes
**QUESTION:** A MST can be calculated for:

a. A connected, unweighted, undirected graph
b. Any weighted, undirected graph
c. An unconnected, weighted, directed graph
d. A connected, weighted, undirected graph
e. A connected, unweighted, directed graph
**QUESTION:** A MST can be calculated for:

a. A connected, unweighted, undirected graph  
b. Any weighted, undirected graph  
c. An unconnected, weighted, directed graph  
d. A connected, weighted, undirected graph  
e. A connected, unweighted, directed graph
QUESTION: Do graphs \textit{with negative cycles} have minimum spanning trees?
QUESTION: Do graphs with negative cycles have minimum spanning trees?

yes
Consider the following graph. We want to find an MST for the graph using Kruskal’s algorithm.

**QUESTION:** Draw the disjoint sets after the first 3 edges have been considered.
Consider the following graph:

**QUESTION:** Draw the disjoint sets after the first 3 edges have been considered.

- **d b**
- **f e**
- **a c**
Consider the following graph. We want to find an MST for the graph using Kruskal’s algorithm.

**QUESTION:** Draw the disjoint sets after the next edge (the 4th edge) has been considered.
Consider the following graph:

**QUESTION:** Draw the disjoint sets after the next edge (the 4th edge) has been considered.

**Diagram:**
- Vertices: a, b, c, d, e, f
- Edges: (a, c), (b, d), (c, b), (d, f), (b, e), (c, a)

**Disjoint Sets: d b a c, f e**
Consider the following graph. We want to find the shortest paths to all nodes from the starting node ‘a’ using Dijkstra’s algorithm.

**QUESTION:** What are the key and predecessor values for each node after we run the algorithm?
Consider the following graph. We want to find the shortest paths to all nodes from the starting node ‘a’ using Dijkstra’s algorithm.

**QUESTION:** What are the key and predecessor values for each node after we run the algorithm?

<table>
<thead>
<tr>
<th>Node</th>
<th>Key</th>
<th>( \pi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>∅</td>
</tr>
<tr>
<td>b</td>
<td>6.5</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>4.5</td>
<td>e</td>
</tr>
</tbody>
</table>
Consider the following graph. We want to find the shortest paths to all nodes from the starting node ‘a’ using Dijkstra’s algorithm.

<table>
<thead>
<tr>
<th>Node</th>
<th>Key</th>
<th>π</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>Ø</td>
</tr>
<tr>
<td>b</td>
<td>6.5</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>4.5</td>
<td>e</td>
</tr>
</tbody>
</table>

**QUESTION:** Draw the shortest paths tree from the table.
Consider the following graph. We want to find the shortest paths to all nodes from the starting node ‘a’ using Dijkstra’s algorithm.

<table>
<thead>
<tr>
<th>Node</th>
<th>Key</th>
<th>π</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>Ø</td>
</tr>
<tr>
<td>b</td>
<td>6.5</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>4.5</td>
<td>e</td>
</tr>
</tbody>
</table>

**QUESTION:** Draw the shortest paths tree from the table.
Consider the following graph. We want to find the shortest paths to all nodes from the starting node ‘a’ using Dijkstra’s algorithm.

<table>
<thead>
<tr>
<th>Node</th>
<th>Key</th>
<th>π</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>∅</td>
</tr>
<tr>
<td>b</td>
<td>6.5</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>4.5</td>
<td>e</td>
</tr>
</tbody>
</table>

**QUESTION:** What is the weight of the shortest path from ‘a’ to ‘f’?
Consider the following graph. We want to find the shortest paths to all nodes from the starting node ‘a’ using Dijkstra’s algorithm.

<table>
<thead>
<tr>
<th>Node</th>
<th>Key</th>
<th>π</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>Ø</td>
</tr>
<tr>
<td>b</td>
<td>6.5</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>4.5</td>
<td>e</td>
</tr>
</tbody>
</table>

**QUESTION:** What is the weight of the shortest path from ‘a’ to ‘f’?

**4.5**
Consider the following graph. We want to find the shortest paths to all nodes from the starting node ‘a’ using Dijkstra’s algorithm.

<table>
<thead>
<tr>
<th>Node</th>
<th>Key</th>
<th>π</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>∅</td>
</tr>
<tr>
<td>b</td>
<td>6.5</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>4.5</td>
<td>e</td>
</tr>
</tbody>
</table>

**QUESTION:** What nodes are traversed, in order, for the path from ‘a’ to ‘f’?
Consider the following graph. We want to find the shortest paths to all nodes from the starting node ‘a’ using Dijkstra’s algorithm.

<table>
<thead>
<tr>
<th>Node</th>
<th>Key</th>
<th>$\pi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>b</td>
<td>6.5</td>
<td>f</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>e</td>
<td>3</td>
<td>c</td>
</tr>
<tr>
<td>f</td>
<td>4.5</td>
<td>e</td>
</tr>
</tbody>
</table>

**QUESTION:** What nodes are traversed, in order, for the path from ‘a’ to ‘f’?

a, c, e, f
Consider 2 nodes \( u \) and \( v \).

**QUESTION:** If a DFS grays node \( u \) before node \( v \), we know that a DFS of the transpose will:

a. Gray node U before node V
b. Gray node V before node U
c. Blacken node U before node V
d. Blacken node V before node U
e. None of the above
Consider 2 nodes $u$ and $v$.

**QUESTION:** If a DFS grays node $u$ before node $v$, we know that a DFS of the transpose will:

- a. Gray node U before node V
- b. Gray node V before node U
- c. Blacken node U before node V
- d. Blacken node V before node U
- e. None of the above
Consider 2 nodes $u$ and $v$.

**QUESTION:** If a DFS blackens node $u$ while node $v$ is gray, which of the following are true?

a. There is a path from $V$ to $U$
b. There is a path from $U$ to $V$
c. There is no path from $V$ to $U$
d. There is no path from $U$ to $V
Consider 2 nodes $u$ and $v$.

**QUESTION:** If a DFS blackens node $u$ while node $v$ is gray, which of the following are true?

a. There is a path from $V$ to $U$

b. There is a path from $U$ to $V$

c. There is no path from $V$ to $U$

d. There is no path from $U$ to $V
Consider 2 nodes $u$ and $v$.

**QUESTION:** If a DFS blackens node $u$ while node $v$ is gray, which of the following are **false**?

a. There is a path from $V$ to $U$
b. There is a path from $U$ to $V$
c. There is no path from $V$ to $U$
d. There is no path from $U$ to $V
Consider 2 nodes \( u \) and \( v \).

**QUESTION:** If a DFS blackens node \( u \) while node \( v \) is gray, which of the following are **false**?

a. There is a path from \( V \) to \( U \)
b. There is a path from \( U \) to \( V \)
c. There is no path from \( V \) to \( U \)
d. There is no path from \( U \) to \( V \)
QUESTION: Prim’s and Kruskal’s MST algorithms are:

a. Recursive
b. Divide and conquer
c. Dynamic programming
d. Greedy
QUESTION: Prim’s and Kruskal’s MST algorithms are:

a. Recursive
b. Divide and conquer
c. Dynamic programming
d. Greedy
DFS can be used as a MST algorithm! Run DFS on the graph, and, if you find a cycle, delete a heaviest edge from the cycle, color all the nodes white again, and repeat, stopping when the graph is acyclic. This final acyclic graph is a MST for the original. Where Kruskal’s and Prim’s start empty and add edges until they have a MST, this algorithm starts with the original graph and deletes edges until it has a MST.

**QUESTION:** What is the worst case runtime? Assume each DFS finds only one cycle, and hence deletes only one edge. (Hint: how many edges do you start with, how many edges will you end with, what is the cost of each DFS)
DFS can be used as a MST algorithm! Run DFS on the graph, and, if you find a cycle, delete a heaviest edge from the cycle, color all the nodes white again, and repeat, stopping when the graph is acyclic. This final acyclic graph is a MST for the original. Where Kruskal’s and Prim’s start empty and add edges until they have a MST, this algorithm starts with the original graph and deletes edges until it has a MST.

**QUESTION:** What is the worst case runtime? Assume each DFS finds only one cycle, and hence deletes only one edge. (Hint: how many edges do you start with, how many edges will you end with, what is the cost of each DFS)

**DFS cost:** $\Theta(V + E)$

Start with $E$ edges, end with $V - 1$ edges:

$$(E - (V - 1))(V + E) = O((E + V)(E - V))$$

$$= O(E^2 - V^2)$$
QUESTION: Does Kruskal’s algorithms use a heap?
QUESTION: Does Kruskal’s algorithms use a heap?

No
Thinking of Prim’s in terms of the conceptual “cut” helps us choose safe edges.

**QUESTION:** Which groups of vertices does the “cut” separate? It separates the _________________ nodes from the rest.
Thinking of Prim’s in terms of the conceptual “cut” helps us choose safe edges.

**QUESTION:** Which groups of vertices does the “cut” separate? It separates the *growing tree* nodes from the rest.
Consider the set of graphs where Dijkstra’s and Bellman-Ford could return different shortest path trees.

**QUESTION:** What are the properties of such graphs?
Consider the set of graphs where Dijkstra’s and Bellman-Ford could return different shortest path trees.

**QUESTION:** What are the properties of such graphs?

1. They have negative edges in which case Dijkstra’s doesn’t necessarily return a shortest path.

2. They have multiple shortest paths of the same weight, and since Dijkstra’s chooses based on a greedy strategy and Bellman-ford chooses based on the order of edge relaxations the paths could be different.
Assume we want to find Single Source Shortest Paths. (Assume any use of Dijkstra’s algorithm uses a binary heap.)

**QUESTION:** What is the best complexity we can achieve for a unit graph (all edges are weight one)?
Assume we want to find Single Source Shortest Paths. (Assume any use of Dijkstra’s algorithm uses a binary heap.)

**QUESTION:** What is the best complexity we can achieve for a unit graph (all edges are weight one)?

**BFS:** $O(V + E)$
Assume we want to find Single Source Shortest Paths. (Assume any use of Dijkstra’s algorithm uses a binary heap.)

**QUESTION:** What is the best complexity we can achieve for graphs without negative edge weights?
Assume we want to find Single Source Shortest Paths. (Assume any use of Dijkstra’s algorithm uses a binary heap.)

**QUESTION:** What is the best complexity we can achieve for graphs without negative edge weights?

Dijkstra’s: $O(E \lg V)$
Assume we want to find Single Source Shortest Paths. (Assume any use of Dijkstra’s algorithm uses a binary heap.)

**QUESTION:** What is the best complexity we can achieve for graphs with negative edge weights?
Assume we want to find Single Source Shortest Paths. (Assume any use of Dijkstra’s algorithm uses a binary heap.)

**QUESTION:** What is the best complexity we can achieve for graphs with negative edge weights?

*Bellman-Ford: $O(VE)$*