Problem 1. [10 points]
Consider the different graph representations discussed in class, and fill out the table below. In the following table, show the worst-case running times for the best algorithms for implementing the given operations. 

**Note:** Be careful what data structures you will be using for each implementation.

<table>
<thead>
<tr>
<th>Operation</th>
<th>Adjacency Matrix</th>
<th>Adjacency List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add a vertex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove a vertex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Find a vertex</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Add an edge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Remove an edge</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Find an edge</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Problem 2. [10 points]
In order to asymptotically analyze the running time of an algorithm, one has to think about how the data will be represented and manipulated in an implementation of that algorithm, so as to bound the number of computational steps it takes. In this problem, you will use arrays to implement the different high-level operations in the Gale-Shapley algorithm presented in Chapter 1, and shown in Figure 1.

Rewrite the algorithm, with the appropriate data structures, to achieve an worst case running time complexity of $O(n^2)$.

Problem 3. [15 points]
Consider a version of the stable matching problem where there are $n$ men and $n$ women as before. Assume each man ranks the women (and vice versa), but now we allow ties in the ranking. In other words, we could have a man that is indifferent two women $w_1$ and $w_2$, but prefers either of them over some other woman $w_3$ (and vice versa). We say a woman $w$ prefers man $m_1$ to $m_2$ if $m_1$ is ranked higher on the $w$’s preference list and $m_1$ and $m_2$ are not tied.

With indifferences in the rankings, there could be two natural notions for stability. And for each, we can ask about the existence of stable matchings, as follows.

(a) **Strong Instability.** A strong instability in a perfect matching, $S$, consists of a man $m$ and a woman $w$, such that each of $m$ and $w$ prefers the other to their partner in $S$. **Does there always exist a perfect matching with no strong instability?** Either give an example of a set of men and women with preference lists for which every perfect matching has a strong instability; or give an algorithm that is guaranteed to find a perfect matching with no strong instability.

(b) **Weak Instability.** A weak instability in a perfect matching, $S$, consists of a man $m$ and a woman $w$,
Initially all $m \in M$ and $w \in W$ are free
While there is a man $m$ who is free and hasn't proposed to every woman
    Choose such a man $m$
    Let $w$ be the highest-ranked woman in $m$'s preference list to whom $m$ has not yet proposed
    If $w$ is free then
        $(m, w)$ become engaged
    Else $w$ is currently engaged to $m'$
        If $w$ prefers $m'$ to $m$ then
            $m$ remains free
        Else $w$ prefers $m$ to $m'$
            $(m, w)$ become engaged
            $m'$ becomes free
    Endif
Endif
Endwhile
Return the set $S$ of engaged pairs

---

Figure 1: The Gale-Shapley algorithm.

such that their partners in $S$ are $w'$ and $m'$, respectively, and one of the following holds:
* $m$ prefers $w$ to $w'$, and $w$ either prefers $m$ to $m'$ or is indifferent between these two choices; or
* $w$ prefers $m$ to $m'$, and $m$ either prefers $w$ to $w'$ or is indifferent between these two choices.

In other words, the pairing between $m$ and $w$ is either preferred by both, or preferred by one while the other is indifferent. Does there always exist a perfect matching with no weak instability? Either give an example of a set of men and women with preference lists for which every perfect matching has a weak instability; or give an algorithm that is guaranteed to find a perfect matching with no weak instability.

**Problem 4.** [6 points]
Proof problem - please submit separately in class.

In class, we discussed the following theorem:

**Theorem.** Let $G$ be an undirected graph on $n$ nodes. Any two of the following statements imply the third.
1 - $G$ is connected.
2 - $G$ does not contain a cycle.
3 - $G$ has $n-1$ edges.

**Prove** that this theorem is true.

**Problem 5.** [9 points]
A $d$-ary heap is like a binary min-heap, but instead of 2 children, nodes have $d$ children.

(a) How would you represent a $d$-ary heap in an array?
(b) What is the height of a $d$-ary heap of $n$ elements in terms of $n$ and $d$?
(c) Give an efficient algorithm (pseudocode) of Heap-Extract-Min in a $d$-ary min-heap. Analyze its running time in terms of $d$ and $n$. Explain your answer.