1. Consider the following set of points.

(a) (2 points) Draw the convex hull on the above set of points.
See above.

(b) (3 points) Which algorithm, Jarvis-March or Graham-Scan, will perform better on the above set of points? Justify your answer.

Graham-Scan will perform better. Graham-Scan is $O(n \lg n)$, where $n$ is the number of points, and Jarvis-March is $O(nh)$, where $h$ is the number of points on the convex hull. So, $h << \lg n$ for Jarvis-March to be preferred, but here $h = 4$ and $\lg n = \lg 8 = 3$.

2. (10 points) The maximum independent set (IS) problem is to find the largest set of vertices in an undirected graph such that no two vertices in the set are connected by an edge in the graph. The corresponding decision problem is

$$\text{IS} = \{(G, k) \mid \text{there is a maximum independent set in } G \text{ of size } k\}$$

Prove that IS is NP-Complete. Hint: Reduce from the CLIQUE problem.

- First, we must show that IS $\in$ NP by showing that a solution can be verified in polynomial time. A solution to IS is a set of points $P$. We first verify that $|P| = k$. Then, we verify for each pair of points $u, v \in P$ there does not exist an edge $(u, v)$ in $G$, which takes $P^2$ time. Thus, the solution can be verified in polynomial time, and IS $\in$ NP.
• Second, we must show that IS ∈ NP-Hard by showing that a known NP-Complete problem reduces to IS; namely, we show that CLIQUE \leq^P IS, i.e., there is a polynomial-time reduction \( f \) of any CLIQUE problem to an IS problem such that \( \langle G, k \rangle \in \text{CLIQUE} \) if and only if \( f(\langle G, k \rangle) \in \text{IS} \). The reduction \( f(\langle G, k \rangle) = \langle \bar{G}, k \rangle \), where \( \bar{G} \) is the compliment of \( G \), i.e., edge \((u, v) \in G\) if and only if \((u, v) \notin G\). \( \bar{G} \) can be computed from \( G \) in \( O(V^2) \) time, where \( V \) is the number of vertices in \( G \). If \( G \) has a clique of size \( k \), then the vertices comprising the clique will have no edges to each other in \( \bar{G} \), and thus form an independent set of size \( k \). Therefore, the NP-Complete CLIQUE problem is poly-time reducible to IS, which proves IS ∈ NP-Hard.

• Finally, since IS ∈ NP and IS ∈ NP-Hard, then IS is NP-Complete.

3. Consider the following approximation algorithm for the CLIQUE problem.

\[
\text{APPROX-CLIQUE}(G)
\]

1. \( C = \{\} \)
2. while \( G \) is not a clique
3. \( u = \text{vertex in } (G - C) \) with maximum degree
4. \( C = C \cup \{u\} \)
5. foreach vertex \( v \) in \( (G - C) \)
6. if edge \((u, v) \) is not in \( G \)
7. then remove vertex \( v \) and all its edges from \( G \)
8. return \( C \)

(a) (3 points) Give an asymptotic upperbound expression for the worst-case running time of APPROX-CLIQUE in terms of the size of \( G = (V, E) \). Your expression should be polynomial in \( V \) and \( E \).

In the worst case each pass through the while-loop requires \( O(V^2) \) time to check if \( G \) is a clique, \( O(V^2) \) time to find the vertex with maximum degree, and \( O(V^2) \) time to remove vertices and edges not connected to a vertex in \( C \). In the worst-case, the while-loop will execute once for each vertex, giving a total worst-case running time of \( O(V^3) \) for APPROX-CLIQUE.

(Note that a tighter analysis could be performed based on the fact that the number of vertices \( V \) in \( G \) is reducing over time.)
(b) (4 points) Draw a graph for which the above algorithm does not find the maximum-sized clique. Indicate the maximum clique and the clique found by APPROX-CLIQUE.

For the above graph, APPROX-CLIQUE will first choose vertex 1 as the maximum-degree vertex, and then remove vertices 6 and 7, and edges (5,6), (5,7) and (6,7). Since the remaining vertices 1-5 do not form a clique, APPROX-CLIQUE will then choose either vertex 2, 3, 4 or 5 as the maximum-degree vertex and remove the others along with their edges to vertex 1. The result is a clique of size 2; whereas the maximum clique is of size 3 involving vertices 5, 6 and 7.

(c) (3 points) Give the ratio bound for this approximation algorithm in terms of $k$, the size of the maximum clique. Justify your answer.

The ratio bound, which is the size of the maximum clique divided by the size of the clique that could be found by APPROX-CLIQUE, is $k/2$. The justification can be seen in the graph below, which is a generalization of the counterexample in the previous problem. Namely, APPROX-CLIQUE will always choose vertex 1 first, which will cause all of the edges and all but one of the vertices in the clique on the right to be removed. Finally, APPROX-CLIQUE will choose one of the remaining vertices, and remove the rest, resulting in a clique of size 2. Thus, for any clique of size $k$, there exists a graph for which APPROX-CLIQUE will output a clique of size 2.