Problem 1. For every natural number n, there are n distinct complex numbers r_i ($i = 0, 1, \ldots, n-1$), such that $r_i^n = 1$. These are called the n^{th} roots of unity. We are going to work with the 3^{rd} roots of unity. Unless stated otherwise, make sure to show your work.

(a) We can write a 3^{rd} root of unity generically as $a + bi$. Simplify $(a + bi)^3$, writing it as $A + Bi$. We call A the real part, and B the imaginary part.

(b) Find the three 3^{rd} roots of unity by setting the real part (A) equal to 1 and the imaginary part (B) equal to 0.

(c) Calculate the sum of the three 3^{rd} roots of unity.

(d) Calculate the product of the three 3^{rd} roots of unity.

(e) Draw the unit circle centered at the origin on the Cartesian Coordinates. You can do this very roughly on regular paper. Show (approximately) where each of the three 3^{rd} roots of unity are on the unit circle.

(f) Euler's formula says that

$$e^{ix} = \cos x + i \sin x$$

Calculate the value of x for each of your 3^{rd} roots of unity. Do not justify.

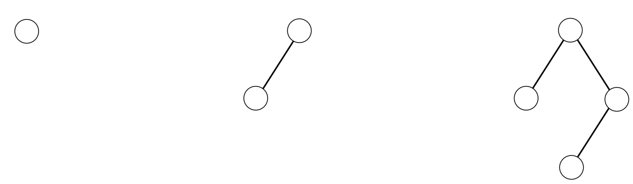
(g) Calculate the product of the three 3^{rd} roots of unity, written in the form e^{ix}.

(h) Redraw the unit circle centered at the origin on the Cartesian Coordinates. Show (approximately) where each of the three 3^{rd} roots of unity, written in the form e^{ix}, are on the unit circle.

Problem 2. A *binomial tree of order k* is defined recursively as follows:

- A binomial tree of order 0 is a single node.
- For $k > 0$, a binomial tree of order k is formed by taking two binomial trees of order $k - 1$ and making the root node of one the rightmost child of the root node of the other.

Here are the first three binomial trees (of orders 0,1,2).



(a) Draw the binomial tree of order 3.

(b) Let $S(k)$ be the number of nodes in the binomial tree of order k.

   i. Guess a formula for $S(k)$.

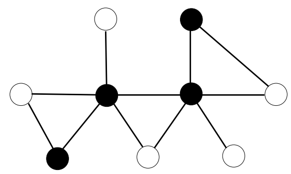
   ii. Prove your formula using Structural Induction. (If you are not sure about Structural Induction, use Weak Induction instead and state that you are doing so.)

(c) Let $L(k, j)$ be the number of nodes at level j of a binomial tree of order k. (The root is at level 0.)

   i. Guess a formula for $L(k, j)$.

   ii. Prove your formula using Structural Induction. (If you are not sure about Structural Induction, use Weak Induction instead and state that you are doing so.)

Problem 3. An edge in a graph is *incident* to a vertex if the vertex is one of the two endpoints of the edge. A *vertex cover* of a graph is a subset of the vertices so that every edge in the graph is incident to at least one vertex in the subset. For example, the filled in vertices are a vertex cover of size four for the following graph.



Given an undirected graph $G = (V, E)$ and an integer k, the *vertex cover problem* is to determine if G has a vertex cover of size k. The vertex cover problem is NP-complete.

Let n be the number of vertices in a graph, and m be the number of edges.

(a) Let an undirected graph $G = (V, E)$ be represented by an $(n \times n)$ adjacency matrix A, and a subset of vertices C be represented by an array (with $|C|$ entries, each a distinct value from the set $\{1, 2, \ldots, n\}$). Give an $O(n^2)$ algorithm that determines if C is a vertex cover of G, using $O(n^2)$ space. Write the pseudo-code.

(b) Let an undirected graph $G = (V, E)$ be represented by an adjacency list, and a subset of vertices C represented by an array (with $|C|$ entries, each a distinct value from the set $\{1, 2, \ldots, n\}$). Note that edge (x, y) will be on the lists of edges for both vertices x and y. You can assume that the two copies of the same edge have pointers to each other. Give an $O(m + n)$ algorithm that determines if C is a vertex cover of G, using $O(m + n)$ space. Write the pseudo-code.

(c) Give a “brute force” algorithm that, given a graph $G = (V, E)$, represented by an $(n \times n)$ adjacency matrix A, and an integer k, determines if G has a vertex cover of size k. Your algorithm should check every possible subset of vertices of size k. Write the pseudo-code.

Problem 4. **Optional problem, will not be graded.** We repeat Problem 1 for general n.

(a) What are the n n^{th} roots of unity?

(b) What is the product of the n n^{th} roots of unity?

(c) What is the sum of the n n^{th} roots of unity?

Problem 5. **Optional problem, will not be graded.** Here is an equivalent definition of a binomial tree: A *binomial tree of order k* is formed as a root node with k children, where child i is the root of a binomial tree of order i for $0 \leq i < k$. Repeat Problem 2 with this new definition. Use either Structural Induction or Strong Induction.

Problem 6. **Optional problem, will not be graded.** Let $G = (V, E)$ be an undirected graph. Give an $O(kn)$ algorithm to determine if a particular subset of vertices of size k is a vertex cover of G.

Problem 7. **HARD Challenge problem, will not be graded.** This last result implies that for constant k, it takes $O(kn^{k+1}) = O(n^{k+1})$ time to determine if a graph has a vertex cover of size k. Find more efficient algorithms for constant k. (Incredibly, it can be done in linear time, i.e. $O(n)$ time.)