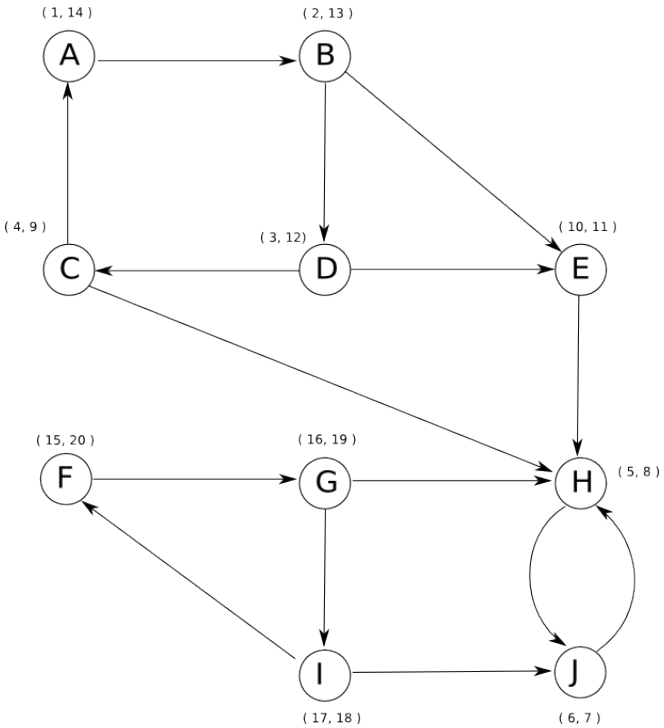
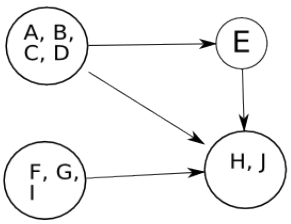
# Midterm 1 Solutions

## Problem 1 (*15 points)*

Do a depth-first search of the graph below. Process nodes (and edges out of a node) in alphabetic order. Show the pre and post order numbers. **See Below** How many strongly connected components are there (no need to show how to find them)? **4.** What is the graph after you shrink them? **See Below** What is the minimum number of edges you have to add to make the graph strongly connected? **2.**



# Problem 2 (*15 points)*

1. (*10 points)* In an RSA cryptosystem, $p = 5$ and $q = 7$. Find the smallest encryption exponent $e > 1$, which is legal for RSA encryption, and the corresponding $d$. Encrypt the message 3. How many legal $(e, d)$ pairs are there, not including $(e = 1, d = 1)$? Would they all result in different encryptions of 3?

2. (*5 points)* What is the FFT of $(1, 0, 0, 1)$? What is the appropriate value of $\omega$ in this case? And of which sequence is $(1, 0, 0, 1)$ the FFT?

## Solution to Part 1

The public exponent $e$ must be relatively prime to $(p-1)(q-1) = 24$. The smallest such $e > 1$ is $e = 5$. The private key $d$ is the inverse modulo 24 of $e$, so the corresponding $d$ is $5^{-1} \mod 24 = 5$ since $5 \times 5 = 25 \equiv 1 \mod 24$. The encrypted message is $m^e \mod N$, where $m = 3$, $e = 3$, and $N = pq = 35$. This gives $3^5 \mod 35 \equiv 33$.

For an ($e$, $d$) pair to be valid, $e$ must be relatively prime to $(p-1)(q-1) = 24$. We also assume $1 < e < 24$. This gives 7 possible values for $e$: 5, 7, 11, 13, 17, 19, and 23. For each $e$, there is a unique inverse modulo 24, and hence a unique $d$. Therefore, there are 7 valid ($e$, $d$) pairs.

They do not all result in different encryptions of $m = 3$ because $3^{12} \equiv 1 \mod 35$. Therefore, $3^5 \mod 35$ and $3^{17} \mod 35$ are the same encrypted message. Similarly, $3^7 \mod 35$ and $3^{19} \mod 35$ are the same, as well as $3^{11} \mod 35$ and $3^{23} \mod 35$.

## Solution to Part 2

When $n = 4$, $\omega = e^{(2\pi i/4)} = i$, so the FFT matrix $M_4(\omega)$ is

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix}.$$

Therefore, the FFT of $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ is

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & i & -1 & -i \\ 1 & -1 & 1 & -1 \\ 1 & -i & -1 & i \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1-i \\ 0 \\ 1+i \end{pmatrix}.$$

When $n = 4$, the inverse FFT matrix is $\frac{1}{4}M_4(\omega^{-1})$, which is

$$\frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix}.$$

Therefore, the inverse FFT of $\begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix}$ is

$$\frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{4}(1+i) \\ 0 \\ \frac{1}{4}(1-i) \end{pmatrix}.$$

## Problem 3 (Dijkstra) (*15 points)*

We are given a directed graph $G$ with positive weights on its edges. We wish to find a shortest path from $s$ to $t$, and, among all shortest paths, we want the one in which the longest edge is as short as possible. How would you modify Dijkstra's algorithm to this end?

### Solution to Problem 3

When Dijkstra's algorithm runs, it normally uses a variable dist($u$) to store the current best shortest path distance to a node $u \in V$, and prev($u$) to store the last node needed to reach node $u \in V$ in the current best shortest path found. In order to solve this new problem, we need to create a new variable long($u$) for all nodes $u \in V$, which keeps track of the length of the longest edge which needs to be used in order to reach node $u$ in the current best shortest path. To this end, we can define our algorithm as follows:

****Initialize Variables****
For all $u \in V$
    dist($u$) = $\infty$
    long($u$) = $\infty$
    prev($u$) = nil
dist($s$) = 0
long($s$) = 0
$H$ = makequeue($V$)

****Begin Main Loop****
While $H$ is not empty
    $u$ = deletemin($H$)
    for all edges $(u, v) \in E$
        if (dist($v$) > dist($u$) + l($u$,$v$)) OR
                ((dist($v$) == dist($u$) + l($u$,$v$)) AND (long($v$) > max(long($u$), l($u$,$v$))))
            dist($v$) = dist($u$) + l($u$,$v$)
            long($v$) = max(long($u$),l($u$,$v$))
            prev($v$) = $u$

Note that at the end of the algorithm prev($u$) stores the last edge used in the shortest path to node $u$, where the shortest path chosen is a path amongst all shortest paths, where the longest edge in the path is as small as possible. Furthermore, we can reconstruct the requisite shortest path from $s$ to $t$ using the prev($u$) pointers.

## Problem 4 (Divide-and-Conquer) (*10 points)*

How many lines does this algorithm print? Write a recurrence and solve it.

```
function printalot(n:  an integer power of 2)
    if n > 1 {
        printalot(n/2)
        printalot(n/2)
        printalot(n/2)
        for i = 1 to n^4 do
            printline(``are we done yet?'')
    }
```

### Solution to Problem 4

If $T(n)$ is the number of lines printed for input $n$, then $T(n) = 3T(n/2) + n^4$ with a base case of $T(1) = 0$.

For an asymptotic solution, we can apply the master theorem with $n^{\log_b a} = n^{\log_2 3}$ and $f(n) = n^4$. Since $f(n)$ polynomially dominates $n^{\log_b a}$ (see handout on website), $T(n) = \Theta(f(n)) = \Theta(n^4)$ by the master theorem.

For an exact solution, we expand the recurrence as follows:

$$T(n) = n^4 + 3(n/2)^4 + 9(n/4)^4 + \ldots + 3^k(2)^4,$$

where $k = \log_2 n - 1$ since $T(1) = 0$.

This gives $T(n) = \sum_{i=0}^{k} 3^i(n/2^i)^4 = n^4 \sum_{i=0}^{k}(3/16)^i = n^4((\frac{3}{16}^{\log_2 n} - 1)/(\frac{3}{16} - 1))$ (by the geometric series formula and using $k = \log_2 n - 1$ ). This can be further simplified to $T(n) = \frac{16}{13}n^4(1 - n^{\log_2(3/16)}) = \frac{16}{13}(n^4 - n^{\log_2 3})$.

## Problem 5 (True or False?) (*25 points*)

Just circle the right answer. No need for justification. Guess all you want, no points subtracted for wrong answers.

- **True** If started with two consecutive Fibonacci numbers, Euclid's algorithm will go through all Fibonacci numbers that come before these two.
  **Solution:** Note that Euclid's gcd algorithm will compute $gcd(a, b)$ as $gcd(b, a \bmod b)$ when $a > b$ recursively. If $a$, $b$, and $c$ are Fibonacci numbers such that $a = b + c$, then $a \bmod b = c$. We can subtract exactly one $b$ from $a$, since $c$ is less than $b$. The recursion will repeat with whatever came before $b$ and $c$.

- **False** $2^{81} = 6 \bmod 7$
  **Solution:** $2^3 \bmod 7 = 1$, so $2^{3*27} \bmod 7 = 1$.

- **True** $2^{81} = 2 \bmod 15$
  **Solution:** $2^4 \bmod 15 = 1$, so $2^{81} = 2^{80} * 2 = 2 \bmod 15$

- **True** The inverse FFT of $(0, 0, 0, 0)$ is $(0, 0, 0, 0)$.
  **Solution:** The vector $(0, 0, 0, 0)$ times anything will result in $(0, 0, 0, 0)$.

- **True** In the FFT, for any integer $j$ between 1 and $n - 1$, $\omega^j + \omega^{2j} + \omega^{3j} + \ldots + \omega^{(n-1)j} = -1$.
  **Solution:** Use the finite geometric series formula to show it's true for all $j$ not an integer multiple of $n$. $\omega^j + \omega^{2j} + \ldots + \omega^{(n-1)j} = (\omega^{nj} - \omega^j)/(\omega^j - 1) = (1 - \omega^j)/(\omega^j - 1)$ (since $\omega^n = 1$) $= -1$ when $j$ is not a multiple of $n$.

- **False** The solution of $T(n) = 2 \cdot T(n/2) + n^2$ is $\Theta(n^2 \log n)$.
  **Solution:** By the Master theorem, it should be $\Theta(n^2)$

- **True** If all edge weights are $1, 2$, or 3, the shortest path problem can be solved in linear time.
  **Solution:** Insert extra nodes into every edge of length 2 or 3 so that every edge is unit length, and then run a Breadth-First search.

- **True** In dense graphs (graphs with about $V^2$ edges), Dijkstra's algorithm takes $O(|E|)$ time.
  **Solution:** You can use an array implementation, which takes $O(|V|^2)$ time.

- **True** In depth-first search, a back edge is a sure sign that the directed graph is not a dag.
  **Solution:** A back edge would make a cycle.

- **False** In depth-first search, a cross edge is a sure sign that the directed graph is not biconnected.
  **Solution:** Consider a full, four-node digraph, where only one link is missing (say it is $C- > D$). The DFS tree will have $C$ and $D$ as siblings, and the edge $(D, C)$ will be a cross edge. The graph is still biconnected, since removing any node will not separate the digraph.

- **True** Two biconnected components of a graph may have a node in common.
  **Solution:** Picture two 3-cycles on an undirected graph of five nodes where only one node is connected to the other four. The graph isn't completely biconnected, since removing the main node will break it. Each component is biconnected, and they share the center node.

- **False** Dijkstra's algorithm works even if there are negative edge weights, as long as $w + w' \geq 0$ for any two distinct edges with weights $w, w'$.
  **Solution:** Consider a digraph with a node $A$ connected to $B$ and $C$ with edges $(A, B) = 5$ and $(A, C) = 3$. Make the edge $(B, C) = -5$. Make an edge $(C, D) = 2$. Dijkstra will compute the shortest path to $C = 3$ and then process node $C$. When it processes $C$, it will set the distance to $D$ to 5. When it processes $B$, it will update $dist(C)$, but that change will not propagate to $D$.

- **False** Calculating $a^{53}$ mod $b$ takes 6 multiplications.
  **Solution:** It takes 8 multiplications: $a * a^{26} * a^{26}$, $a^{26} = a^{13} * a^{13}$, $a^{13} = a * a^6 * a^6$, $a^6 = a^3 * a^3$, and $a^3 = a * a * a$.

- **False (not graded)** The word "algorithm" comes from a term in Arabic meaning "tent-pitching method."
  **Solution:** The word comes from the name Al Khwarizmi, a mathematician from the ninth century who lived in Baghdad and who laid out the basic steps for arithmetic operations. However, he may have come from a long and noble line of tent pitchers.