Part I: Short Answer - Do 5 of the following 6 problems. (6 points each, 30 total)

1. (i). Give an example of a rate of growth between $O(\sqrt{n})$ and $O(n)$.
   (ii). Give an example of a rate of growth between $O(n \log n)$ and $O(n^2)$.

In problems 2 and 3 assume we have an LUP decomposition of a matrix $A$ with $LU=PA$ where the $L$, $U$ and $P$ matrices are given below.

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 3 & 0 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} 2 & 2 & 1 \\ 0 & 4 & 0 \\ 0 & 3 & 0 \end{pmatrix}, \quad P = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

2. Given the three matrices $L$, $U$ and $P$ above what is $A$?

3. Which of the following statements about these matrices $L$, $U$, $P$ and $A$ is FALSE? For each answer that is false, briefly explain why. If it’s true just label it T (no proof needed).
   a. The determinant of $P$ is 1.
   b. $U$ is upper triangular and singular
   c. The matrix $P$ is its own inverse.
   d. If we used this LUP decomposition to solve for $X = (x_1, x_2, x_3)$ in the equation $AX=B$, where $B$ is the column matrix $(1,1,1)$ then there would be exactly one value of the matrix $X$ where $AX=B$ is true.
   e. $A$ is non-singular.

4. The bin packing problem is: We are given $n$ objects $O_1, O_2, ..., O_n$, each of size between 0 and 1. We want to pack the objects into bins of size 1 using as few size bins as possible.

Here we consider two bin packing algorithms. The First Fit (FF) algorithm is where we put each object into the first bin that we’ve used before where it fits. If there are no bins in the list where it fits then open a new bin and put the object into that bin.

The next fit (NF) bin packing algorithm is one which runs like FF except that We start with one open empty bin and never have more than one open bin at any time.

In NF, when we have an object $O$ to pack we first see if it fits in the one open bin. If so, put it in. If not, close the open bin, open an empty bin and put $O$ in the newly open bin.

   a. Give an example of a bin packing instance $I$ where the NF algorithm uses at least 2 more bins to pack $I$ than FF does. Show your work.

   b. Briefly explain why, for any set of input objects, FF always does no worse than NF.
5. T or F: We want to interpolate a polynomial $p(x)$ of degree $n$ from a list of points $a_1, a_2, a_3,\ldots, a_{n+1}$, and $p$’s values, $p(a_1), p(a_2), p(a_3),\ldots, p(a_{n+1})$. Then,
a. to do this the values $p(a_1), p(a_2), p(a_3),\ldots$ must all be different.
b. the result of the interpolation is the $n+1$ coefficients of $p(x)$.
c. having only $n$ values of $p$ are sufficient to do the interpolation.
d. the $a_i$’s must all be $n$th roots of unity.
e. we can only do this if $n$ is a power of 2.

6. Consider the following flow network $G$ and legal flow $f$ through $G$. In this graph an edge is labeled with the flow $f$ over it and with its capacity. So an edge with label $3/8$ means there is flow of 3 over that edge and the edge’s capacity is 8.

   a. Draw the residual graph $G_f$ resulting from this $G$ and $f$.
   b. Give an example of an augmenting path $p$ in $G_f$. What is the capacity of $p$ ?

Part II: Do 2 of the following 3 problems (10 points each, 20 points total)

   1. Suppose you want to find a better algorithm to multiply an $n$ by $n$ matrix by an $n$ by 1 vector. (Like Strassen’s algorithm but for matrix times vector multiplication.) Someone shows you a way to do this using 3 multiplications of $n/2$ by $n/2$ matrices by $n/2$ by 1 vectors plus $n$ more multiplications of numbers. (We are ignoring any additional additions here and you may assume $n$ is a power of 2.)

   a. What is the recurrence relation which arises from this new algorithm? (I.e. Define $T(n)$ = the number of integer multiplications carried out in the above algorithm. Write a recurrence equation for $T(n)$. Make sure to include a base case.)

   b. Compute the values $T(4)$ and $T(16)$. What is a general solution to this recurrence?

   2. In this problem we consider a reduction of the vertex cover problem (VC) to the unweighted set cover problem.

   (i). First say in detail what a reduction from VC to unweighted set cover means.
   That is, what is the input to the reduction, what is its output and what is the relationship between the input and the output in this specific case?

   (ii). Now present a reduction from VC to unweighted set cover.
   Given the input to the reduction how do you transform this input to the reduction’s output?

   (iii). Show how your reduction works on the graph $G$ below.