Instructions:

1. Write neatly and legibly.

2. This is a closed-book exam. No calculator.

3. While grading, not only your final answer but also your approach to the problem will be evaluated.

4. You have to attempt four problems; You have to choose EXACTLY two problems out of 1, 2 and 3. If you solve all three, only problems 1 and 2 will be graded. Problem 4 and 5 are compulsory.

5. Total time for the exam is 120 minutes.

6. You are not allowed to use a calculator for this exam.

I have read carefully, and have understood the above instructions. On my honor, I have neither given nor received unauthorized aid on this examination.

Signature: ______________________________

Date: _____(MM) / _____(DD) / _________(YYYY)
1. [25 points] The following recursive algorithm sorts a sequence of $n$ numbers. Write down a recurrence describing the running time of the algorithm as a function of $n$. Solve the recurrence relation and state the final time complexity.

```python
def triplesort(seq):
    if n <= 1: return
    if n == 2:
        replace seq by [min(seq), max(seq)]
    return
    triplesort(first 2n/3 positions in seq)
    triplesort(last 2n/3 positions in seq)
    triplesort(first 2n/3 positions in seq)
```

Solution:
$$T(n) = 3T\left(\frac{2n}{3}\right) + \Theta(1)$$

Note that $T(n) = \Theta(n^{2.71})$
2. [25points] Choose 2 problems out of 1, 2 and 3
GREEDY ALGORITHM

Find the minimum spanning tree for the following graph using:
(a) Prim’s Algorithm
(b) Kruskal’s Algorithm
Illustrate how MST is found each step for both algorithms.

Solution:
3. [25 points] [Choose 2 problems out of 1, 2 and 3] DYNAMIC PROGRAMMING

Given bit strings $X = x_1 \ldots x_m$, $Y = y_1 \ldots y_n$ and $Z = z_1 \ldots z_{m+n}$, if $Z$ can be obtained by interleaving the bits in $X$ and $Y$ in a way that maintains the left-to-right order of the bits in $X$ and $Y$, then we say $Z$ is an interleaving of $X$ and $Y$. For example if $X = 101$ and $Y = 01$ then $x_1x_2y_1x_3y_2 = 10011$ is an interleaving of $X$ and $Y$, whereas 11010 is not. Give the most efficient algorithm you can to determine if $Z$ is an interleaving of $X$ and $Y$ and analyze its time complexity as a function $m = |X|$ and $n = |Y|$.

Solution:
Let $c[i, j]$ be true if and only if $z_1 \ldots, z_{i+j}$ is an interleaving of $x_1, \ldots, x_i$ and $y_1, \ldots, y_j$. We use the convention that if $i = 0$ then $x_i = \lambda$ (the empty string) and if $j = 0$ then $y_j = \lambda$. The subproblem $c[i, j]$ can be recursively defined as shown (where $c[m, n]$ gives the answer to the original problem):

- $c[i, j] = \text{true}$ if $i = j = 0$
- $c[i, j] = \text{false}$ if $x_i \neq z_{i+j}$ and $y_j \neq z_{i+j}$
- $c[i, j] = c[i-1, j]$ if $x_i = z_{i+j}$ and $y_j \neq z_{i+j}$
- $c[i, j] = c[i, j-1]$ if $x_i \neq z_{i+j}$ and $y_j = z_{i+j}$
- $c[i, j] = c[i-1, j] \lor c[i, j-1]$ if $x_i = y_j = z_{i+j}$

The time complexity is $O(nm)$ since there are $n \times m$ subproblems each of which is solved in constant time. Finally, the $c[i, j]$ matrix can be computed in row major order.
4. **[1 page][25 points][Compulsory] BRANCH AND BOUND**

**Job sequencing with deadlines problem** is to choose a subset of jobs such that each job in the subset can be completed by its deadline and the sum of the profits of jobs in the subset is maximized. Each job $i$ has associated with it a three tuple $(p_i, d_i, t_i)$, where $p_i$, $d_i$ and $t_i$ are a profit, an deadline, and processing time of the job, respectively.

For the job sequencing with deadlines instance $n = 5$, $(p_1, p_2, ..., p_5) = (6, 3, 4, 8, 5)$, $(t_1, t_2, ..., t_5) = (2, 1, 2, 1, 1)$, and $(d_1, d_2, ..., d_5) = (3, 1, 4, 2, 4)$, draw the portion of the state space tree generated by:

(a) FIFO Branch-and-Bound
(b) Least Cost Branch-and-Bound

A variable tuple size formulation is used and $c(\cdot)$, $\hat{c}(\cdot)$ and $u(\cdot)$ are defined as follows.

- $c(x)$: The minimum penalty corresponding to any node in the subtree with root $x$.
- $\hat{c}(x) = \sum_{i < m, i \notin S_x} p_i$, where $S_x$ is the subset of jobs selected at node $x$, and $m = \max\{i | i \in S_x\}$.
- $u(x) = \sum_{i \notin S_x} p_i$

The entire state space tree using a variable tuple size formulation is given for each algorithm.
(a) FIFO Branch-and-Bound

![FIFO Branch-and-Bound Tree Diagram]
(b) Least Cost Branch-and-Bound
5. [1 page][25points][Compulsory] NP-complete

(a) ATLEAST-FOUR-SAT is the problem of determining whether a given propositional formula in 3-CNF form has at least four satisfying assignments. Prove that this problem is NP-complete by using the fact that 3-CNF-SAT is NP-complete. Is your proof (which shows that ATLEAST-FOUR-SAT is NP-complete) also sufficient to establish that ATLEAST-THREE-SAT (the problem of determining whether a given propositional formula in 3-CNF form has at least three satisfying assignments) is also NP-complete? If yes, explain why. If not, what changes will you make to the proof? [10 + 5 = 15 points]

SOLUTION:
Take a 3-CNF formula \( p = (x_{11} \lor x_{12} \lor x_{13} \lor x_{14}) \land (x_{21} \lor x_{22} \lor x_{23} \lor x_{24}) \land \ldots \land (x_{k1} \lor x_{k2} \lor x_{k3} \lor x_{k4}) \) and convert it to a formula given as \( q = (x_{11} \lor x_{12} \lor x_{13} \lor x_{14}) \land (x_{21} \lor x_{22} \lor x_{23} \lor x_{24}) \land \ldots \land (x_{k1} \lor x_{k2} \lor x_{k3} \lor x_{k4}) \land (y \lor \bar{y}) \land (z \lor \bar{z}) \) where \( y \) and \( z \) are new propositional variables. As \( q \) has two clauses more than \( p \), this is a polynomial time reduction. Moreover we can see that \( q \) has at least four satisfying assignments if and only if \( p \) has at least one satisfying assignment. Furthermore, \( q \) is an input to ATLEAST-FOUR-SAT. Now we see that ATLEAST-FOUR-SAT is at least as hard as 3-SAT, which proves that ATLEAST-FOUR-SAT is NP-hard. It is easy to see that ATLEAST-FOUR-SAT is in NP. Hence ATLEAST-FOUR-SAT is NP-complete. The exact same proof will work for ATLEAST-THREE-SAT also because formulae that have at least four satisfying assignments are a subset of those formulae that have at least three satisfying assignments. As we look for worst case complexity, the same proof shows us that ATLEAST-THREE-SAT is also NP-complete.

(b) Suppose \( S_1 \) and \( S_2 \) are two problems. Assuming that \( P \neq NP \), your task is to identify which of the following statements imply that \( S_2 \) is NOT in \( P \). For this, mark ‘YES’ in front of a statement if it implies that \( S_2 \) is NOT in \( P \) assuming that \( P \neq NP \), and ‘NO’ otherwise. You are not required to provide justification. [10 points]

i. \( S_2 \) is in NP.
ii. \( S_2 \) is NP-complete.
iii. \( S_2 \) is NP-hard.
iv. \( S_1 \) is in NP and \( S_1 \) is polynomial-time reducible to \( S_2 \).
v. \( S_1 \) is in NP and \( S_2 \) is polynomial-time reducible to \( S_1 \).
vi. \( S_1 \) is NP-complete and \( S_1 \) is polynomial-time reducible to \( S_2 \).
vii. \( S_1 \) is NP-complete and \( S_2 \) is polynomial-time reducible to \( S_1 \).
viii. \( S_1 \) is NP-hard and \( S_1 \) is polynomial-time reducible to \( S_2 \).
ix. \( S_1 \) is NP-hard and \( S_2 \) is polynomial-time reducible to \( S_1 \).
x. \( S_2 \) has a \( \Omega(2^n) \) lower bound on the space complexity.

SOLUTION:
(b), (c), (f), (h) and (j) are the only statements that imply that \( S_2 \) is NOT in \( P \).
i. $S_2$ is in NP. (NO, because $P \subset NP$ even if $P \neq NP$)

ii. $S_2$ is NP-complete. (YES)

iii. $S_2$ is NP-hard. (YES)

iv. $S_1$ is in NP and $S_1$ is polynomial-time reducible to $S_2$. (NO, $S_2$ is at least as hard as $S_1$, and $S_1$ could be in $P$, because $P \subset NP$ even if $P \neq NP$)

v. $S_1$ is in NP and $S_2$ is polynomial-time reducible to $S_1$. (NO, because $S_1$ is at least as hard as $S_2$, and so $S_2$ could very well be in $P$ even if $S_1$ were not in $P$).

vi. $S_1$ is NP-complete and $S_1$ is polynomial-time reducible to $S_2$. (YES, because $S_2$ is at least as hard as $S_1$ and $S_1$ being NP-complete is not in $P$ if $P \neq NP$).

vii. $S_1$ is NP-complete and $S_2$ is polynomial-time reducible to $S_1$. (NO, all this tells us is that $S_1$ is at least as hard as $S_2$, so even if $S_1$ is NP-complete, $S_2$ could very well be in $P$).

viii. $S_1$ is NP-hard and $S_1$ is polynomial-time reducible to $S_2$. (YES, because $S_2$ is at least as hard as $S_1$ and $S_1$ being NP-hard is not in $P$ if $P \neq NP$).

ix. $S_1$ is NP-hard and $S_2$ is polynomial-time reducible to $S_1$. (NO, all this tells us is that $S_1$ is at least as hard as $S_2$, so even if $S_1$ is NP-hard, $S_2$ could very well be in $P$).

x. $S_2$ has a $\Omega(2^n)$ lower bound on the space complexity. (YES, clearly $S_2$ also has a lower-bound of $\Omega(2^n)$ on its time complexity, and hence $S_2$ is clearly not in $P$. In fact, for this, we don’t even require that $P \neq NP$.)