Problem 1. Coin Changing  Suppose we have a new coin worth 7 cents. We also have pennies and dimes (but no other types of coins).

(a) Show that the greedy strategy of always using the largest coin is NOT optimal in this setting. You may show this with a counter-example.

(b) Suppose we would like to make change for 70 cents or more. Is it always optimal to use one or more dimes as part of the change? Argue that it is always optimal to use a dime, or give an example (of 70 cents or more) where it is not optimal to use any dimes.

Problem 2. Recurrence Relations  Consider the following program that takes as input an array $A$ of size $n$:

```python
doSomething(A)
    for i = 1 to n:
        print "hi"
        $A = A[3...n]$ //remove the first 2 elements in A
    doSomething(A)
```

Note that the $n$ in the program refers to the current size of $A$.

(a) Write a recurrence $T(n)$, to express the running time of this program.

(b) Draw a recurrence tree for your recurrence to get a good guess of the big-Oh running time. Note: for the actual exam you will either be given the recurrence or asked about a recurrence whose solution you will have already seen or can easily guess.

(c) Use the substitution method to verify your guess from part(b).

Problem 3. Closest Points  Consider a variation of the Closest Pair of Points Problem we discussed in class. We are now given a value $k$ and we want to find any pair of points that are within distance $k$ from each other (assume that the input set of points always contains at least one such pair).

Describe how to modify the pseudocode we derived in class to account for this variation. Of course, one simple approach is to make no modifications and simply output the closest pair :) but we want a more efficient approach in case $k$ is significantly larger than the closest distance.

Does this variation change a point’s bounding box and/or the maximum number of points in the bounding box? If so, how?

Problem 4. Dynamic Programming  In class we discussed a recursive solution for the Coin Changing Problem. We now consider Dynamic Programming to solve this problem for denominations of 25, 10, and 1 cent.

Write a dynamic programming formulation to efficiently solve this problem. Specifically, use a matrix $C$, that holds the minimum number of coins to make change for $n$. Also describe how to find the actual coins. Justify the running time of your program.

Problem 5. Selection Problem  Describe how to use our algorithm for the Selection Problem to efficiently find in a list all the numbers that are smaller than the median. Describe an efficient way to solve this problem (i.e. without running the algorithm several times).