Homework #3

Problem 1: Algorithm Analysis

Consider the following basic problem. Given an array A consisting of n integers $A[1], A[2], ..., A[n]$, output a two-dimensional n-by-n array B in which $B[i,j]$ (for $i < j$) contains the sum of array entries $A[i]$ through $A[j]$, i.e., the sum $A[i]+A[i+1]+...+A[j]$. The value of array entry $B[i,j]$ is left unspecified whenever $i \geq j$, so it doesn't matter what the output is for these values. Below is a simple algorithm to solve this problem:

```
1  for i ← 1 to n
2    for j ← i + 1 to n
3      Add entries A[i] through A[j]
4      Store the result in B[i,j]
```

(a) Give a function $f(n)$ that is an asymptotically tight bound on the running time of the algorithm above. Using the pseudocode above, argue that the algorithm is, in fact $\Theta(f(n))$.

(b) Although the algorithm you analyzed above is the most natural way to solve the problem, it contains some unnecessary sources of high inefficiency. Give a different algorithm to solve this problem with an asymptotically better running time than the provided algorithm.

(c) What is the running time of your new algorithm?

Problem 2: Decision Trees

You are given 9 identical looking balls and told that one of them is slightly heavier than the others. Your task is to identify the defective ball. All you have is a balanced scale that can tell you which of the two sides of the scale is heavier (any number of balls can be placed on each side of the scale).

(a) Show how to identify the heavier ball in just 2 weighings.

(b) Give a decision tree lower bound showing that it is not possible to determine the defective ball in fewer than 2 weighings. Hint: Your decision tree in this case is slightly different-looking from the one shown in class.

Problem 3: Heap

Give an efficient algorithm to find all keys in a min heap that are smaller than a provided value, leaving the heap as it is at the end of running your algorithm. You are allowed access to the entire heap, and not just the root.
The provided value does not have to be a key in the min heap. Evaluate the time complexity of your algorithm.

Problem 4: Graphs

A connected, undirected graph is vertex biconnected if there is no vertex whose removal disconnects the graph. A connected, undirected graph is edge biconnected if there is no edge whose removal disconnects the graph. Give a counterexample for each of the following statements:

(a) A vertex biconnected graph is edge biconnected.

(b) An edge biconnected graph is vertex biconnected.

Problem 5: Graphs

We have a connected graph $G = (V,E)$ and a specific vertex $v \in V$. Suppose we compute a depth first search tree rooted at u and obtain a tree T that includes all nodes of G. Suppose we then compute a breadth-first search tree rooted at u and obtain the same tree T. Prove that $G = T$. (In other words, if T is both a depth-first search tree and a breadth-first search tree rooted at u, then G cannot contain any edges that do not belong to T.)