CS 266 Homework 4

Zachary DeStefano, PhD Student, 15247592

Due Date: May 8

Problem 4.11

Give an example of a 2-dimensional linear program that is bounded, but where there is no lexicographically smallest solution.

This happens in the case where an entire edge of the bounded region is the smallest solution.
Here is a simple example of that. These are the constraints:
$x \leq 4$ and $x \geq 0$
$y \leq 4$ and $y \geq 0$
Our objective function to minimize is then y.
The whole edge where $y = 0$ is the best solution.

Problem 4.14

4.14 Here is a paranoid algorithm to compute the maximum of a set A of n real numbers:
What is the worst-case running time of this algorithm?
What is the expected running time (with respect to the random choice in line 3)?

Worst-case running time

In the worst-case, it recurses every time and checks $n-1$ elements as well as checks all $n-1$ elements in the paranoid step. Thus, if you let c and k be constants, then we have

$$T(n) = c + T(n-1) + k * (n-1)$$

We can simplify this to

$$T(n) = c * n + k + T(n-1)$$

After doing out the recursion, we end up with

$$T(n) = c * (n + (n-1) + (n-2) + ... + 1) + k * n$$

We have an arithmetic series so we can simplify this to the following

$$T(n) = c\frac{n(n+1)}{2} + k * n$$

Simplifying this into big-Oh notation, we have

$$T(n) = O(n^2)$$

Average running time

In the average case, the check of the all the elements again only occurs if we picked the maximum element of the current set. If the current set has n elements then this probability is $\frac{1}{n}$. The recursion will happen regardless of which element we picked.

This makes our equation the following:

$$T(n) = k + c\frac{1}{n}(n-1) + T(n-1)$$

We can replace the $(n-1)$ by n for simplifying and we will then have

$$T(n) \leq k + T(n-1)$$

Expanding this out we have

$$T(n) \leq k * n$$

Thus we now have

$$T(n) = O(n)$$

This gives us the expected running time.

Problem 5.1

We can use the Divide and Conquer Theorem for this problem, which states the following:
Let $a \geq 1, b > 1$ and $c, k \geq 0$ be constants.
If $T(n)$ defined on nonnegative integers by $T(n) = aT(\frac{n}{b}) + c * n^k$
T(n) can be bounded asymptotically as follows:
if $a > b^k$ then $T(n) = \Theta(n^{log_b(a)})$
if $a = b^k$ then $T(n) = \Theta(n^k log(n))$
if $a < b^k$ then $T(n) = \Theta(n^k)$

For this problem, it holds that $a = 2$, $b = 4$, $c = 2$, and $k = 0$.
It then happens that $a > b^k$, thus our solution is $Q(n) = \Theta(n^{log_b(a)})$.
As it happens $log_4(2) = \frac{1}{2}$, thus we have $Q(n) = \Theta(n^{\frac{1}{2}})$
This proves that $Q(n) = O(\sqrt{n})$ and $Q(n) = \Omega(\sqrt{n})$.

TODO: Proof with points and query rectangle

Problem 5.10