Problem Set II

1. Prove the following:
   1. $\log n^k = \theta(\log n)$ for any positive integer k.
   2. $(\log n)^k = o(n^\epsilon)$ for an positive integer k and $\epsilon > 0$.
   3. $n^{\log n} = o(2^n)$.
2. Show that an array storing binary values can be sorted in $O(n)$ time.
3. Suppose you have a linked list of n elements whose elements are sorted. To make search efficient, we will introduce additional pointers from the i^{th} node to the $(i+t)^{th}$ node in the list for $i \in \{0, t, 2t, ..\}$. With this additional pointers and making use of the fact that the list if sorted, what will be the complexity of search for an element in the list? (Express your answer as a function of n and t). What is the optimal value of t that minimizes the search cost in the worst case?
4. If we eliminate the tail recursion from the quick sort algorithm discussed in the class, show that the number of recursive calls can be limited to $O(\log n)$. (Hint: Always work with the larger partition iteratively and smaller partition recursively).
5. Recall the binary search algorithm for search on a sorted array.
   1. Show that the worst case complexity of the algorithm is $O(\log n)$.
   2. Assume that while searching for an element x in the array, the element has equal probability to be present in any of the positions in the array (and that the element indeed will be always present). Under these assumptions, show that the average complexity of binary search is $O(\log n)$.
6. Consider the following strategy for finding the k^{th} smallest element in an array $A[l..r]$, $l < r$.

```
int Find(l, r, k)
   p = partition(l,r);
   If (p == l+k) return A[p];
   If (p < l+k) return Find(p+1,r,k-p);
   If (p > l+k) return Find(l,p,k);
```

   1. Show that the worst case complexity of the algorithm is $O(n^2)$.
   2. Let X_n denote a random variable counting the cost for $Find()$ on an array of n elements. Set up a recurrance for $E(X_n)$.
   3. Show that the average complexity of the algorithm is $O(n)$. (We may assume that $T(i) \leq T(i+1)$ for each i).