Exercises on Algorithm Analysis

1. Suppose you are given a positive number n in binary representation. Describe an algorithm to find the square root of the number, that is, the largest number m such that $m^2 \leq n$. The algorithm should find the binary representation of m. Estimate the number of basic operations required as a function of the number of bits in n. Try to extend this to the cube and higher roots. Given a number n determine whether it can be written as m^k for some numbers m and k. The time required should be $O((\log n)^c)$ for some constant c.

2. In some problems, it is convenient to assume the size of any number is 1 and any basic operation on numbers takes unit time. Consider the Fibonacci numbers defined by $F(0) = F(1) = 1$ and $F(n) = F(n-1) + F(n-2)$. Suppose we write a recursive function to compute $F(n)$ as

```
long long int F(int n)
{
     if (n <= 1) return 1;
     else return F(n-1) + F(n-2);
}
```

   Estimate the number of basic operations executed by this program. It is easy to see that $F(n)$ can be computed using $O(n)$ operations. Find a way of computing $F(n)$ using $O(\log n)$ operations. Use the binary representation of n.

3. Given an array A of integers, not necessarily sorted, an element $A[i]$ is said to be a local minimum if there is no smaller element next to it, at either $A[i-1]$ or $A[i+1]$. Show that in every non-empty array, there exists a local minimum, and describe an $O(logn)$ time algorithm to find any one such element. Suppose now you have an $n \times n$ matrix with integer entries. An element $A[i][j]$ is a local minimum if there is no neighboring element smaller than it. The neighboring elements to $A[i][j]$ are $A[i-1][j], A[i+1][j], A[i][j-1]$ and $A[i][j+1]$. Find an $O(n)$ time algorithm to find a local minimum in a matrix. Hint: Reduce the problem to an $n/2 \times n/2$ matrix in $O(n)$ time.

4. Given an arbitrary sequence of left and right brackets (`'('`,`')'`), describe an $O(n)$ time algorithm to find the longest subsequence that forms a balanced parenthesis sequence. Do the same if you are allowed to circularly shift the initial sequence by any amount and find a subsequence of the resulting sequence. Find the shortest balanced sequence S such that the given sequence is a subsequence of S. Do the same if it required to be a substring of S.