The Ocean University of Qingdao
Department of Computer Science
Design and Analysis of Algorithms

Assignment One

Due date: 12 p.m., November 18, 2010

1. Design a linearithmic algorithm that, in an array of n integers, finds the pair of integers that are closest to each other.

2. Suppose the denominations of the coins in a country are $(c_1, c_2, \ldots, c_n)$ with $c_1 > c_2 > \ldots > c_n$ and $c_n = 1$. The *coin changing* problem is to determine the minimum number of coins to make m cents in change, for any given m. Please do the following

   (a) Give a dynamic programming algorithm to solve the problem.

   (b) Analyze the running time of your algorithm.

   (c) Claim whether or not your algorithm is polynomial time.

3. A binary counter is implemented as an array $A[0..k-1]$, where $A[0]$ stores the lowest order bit, and $A[k-1]$ the highest, so that the number represented by A is

$$x = \sum_{0 \leqslant i < k} A[i]2^i.$$

   The following procedure shows how to increment A.

   Increment(A):

```
1: i = 0; k = length(A)
2: while i < k and A[i] = 1 do
3:     A[i] = 0; i = i + 1
4: end while
5: if i < k then
6:     A[i] = 1
7: end if
```

   Suppose we have executed the Increment for n times, What is the time complexity?

4. Given two strings S_1 and S_2 and a text T, you want to find whether there is an occurrence of S_1 and S_2 interwoven (without space) in T. For example, the string *abac* and *bbc* occur interwoven in *cabbabccdw*. Given an efficient algorithm for this problem.

5. Design an algorithm to count the number of distinct substrings of a string T in $O(m)$ time, where the length of T is m.

6. Give a linear-time algorithm that takes in a string S and finds the longest maximal pair in which the two copies do not overlap. That is, if the two copies begin at positions $p_1 < p_2$ and are of length n', then $p_1 + n' < p_2$

7. The lecture notes has mentioned that suffix tree helps the implementation of Zip-Lempel data compression. Discuss how?