CISC621 Algorithm Design and Analysis, Spring 2016
Homework set IV, problems 10, 11, 12,13, due Tuesday, May 3, 8:00am

Check the "homework sheet" from the syllabus for general homework details. In particular, each homework solution is on an entirely separate (set of) sheet(s) of paper and is identified with your name(s). Do not staple solutions to two or more problems together. Submit at the *beginning* of class, May 3. Strict lateness penalty for any later than 8am.

10. [Individual Problem — modular powering]

    Consider this proposition:
    Let $n$ be a positive integer, then $a^{\phi(n)+1} = a \bmod n$ for all $a \in \mathbb{Z}_n^*$
    ($\mathbb{Z}_n^* = (1, 2, \ldots, n-1)$ is the set of nonzero elements of $\mathbb{Z}_n = (0, 1, 2, \ldots, n-1)$).

    This proposition seemingly follows from theorem 31.30 in CLRS. However that theorem is misstated. Theorem 31.30 is valid if and only if $a$ is relatively prime to $n$.

    (a) Show by example that the proposition is false when $n$ is the square of a prime, $n = p^2$.

    (b) Show that the proposition is true when $n$ is a product of two distinct primes, $n = pq$. [Hint: Use Chinese remainder theorem]

    (c) Extra credit: A positive integer $n$ is called *square free* if $a^2|n$ implies $a = 1$. Show that the proposition is true if and only if $n$ is square free.

    Remark: This exercise shows that RSA works even in the unlikely event that the message encrypted is a multiple of one of the prime factors of $n$.

11. [Individual Problem — dynamic programming] We have studied LCS, the problem of finding the largest common subsequence of two sequences, and we have studied LIS, the problem of finding the largest increasing subsequence of a single sequence. Let's put these together. Find the length of LCIS, the largest common increasing subsequence of two given sequences. Concretely, let lcis(a,m, b,n) be a function which is given two arrays of numbers, a and b, of lengths m and n, respectively. It returns the length of a longest sequence $c = c_1, \ldots, c_k$ such that $c_i < c_{i+1}$ for $i \in 1..k-1$ and $c$ is a subsequence of a and of b.

    The task is to design an efficient algorithm for lcis() and analyze it.

12. [Group Problem — Hankel matrices] *Hankel matrices* are matrices having a pattern that arises in many applications including hidden Markov models. Specifically, A Hankel matrix is a matrix $h = (h_{i,j})$ such that $h_{i,j} = h_{k,l}$ whenever $i + j = k + l$. Let indexing be zero based so that the row and column indices run from 0 to n-1 rather than from 1 to n. A $n \times n$ Hankel matrix can be represented by the $2n-1$ values in the first row and last column, $H_0, H_1, \ldots, H_{2n-2}$. We can represent h by $h_{i,j} = H_{i+j}$.

$$h = \begin{pmatrix} a & b & c & d \\ b & c & d & e \\ c & d & e & f \\ d & e & f & g \end{pmatrix}, H = (a, b, c, d, e, f, g)$$

    (a) Give a linear time algorithm to add two Hankel matrices when they are represented as indicated above.

    (b) A Hankel matrix, represented as above can be multiplied by a vector as follows.

```
mvHankel(H, v, n) {
  for i from 0 to n-1 do
     w[i] = 0;
     for j from 0 to n-1 do
        w[i] = w[i] + H[i+j]v[j];
  return w
}
```

This algorithm runs in $\Theta(n^2)$ time just as general matrix vector product does. Write and analyze an algorithm to compute Hankel matrix times vector product in $O(n \log(n))$ time.

Hint: Hankel matrix times vector product corresponds to (part of) polynomial multiplication.

(c) Design and analyze an efficient algorithm for product of two $n \times n$ Hankel matrices. You can get a complexity much better than Strassen's general puropse matrix multiplication algorithm in $O(n^{2.81}$.

13. [Group Problem — dynagram] Dynagram is a village laid out neatly with numbered streets which run east to west and numbered avenues which run north to south. As a prize to recognize your algorithmic talents, the mayor of Dynagram has a bucket of coins waiting for you at each intersection. The matrix $B[i, j]$ contains the value of the bucket at the intersection of i street with j avenue. The challenge is to make a tour from intersection (0,0) to intersection (m,n) and back to (0,0) collecting the maximal possible prize from the buckets along the way. However you are restricted to go only east and south on the way from (0,0) to (m,n) and go only west and north on the return portion of the trip. Thus until you reach (m,n) the valid moves are to go from (i,j) to (i+1,j) or to (i, j+1). Then from (m,n) onwards the valid moves are to go from (i,j) to (i-1, j) or to (i, j-1). It may or may not be to your advantage to visit some intersection (i,j) both on the way out and on the way back. However you only get one bucket of coins at (i,j)!

Design and analyze an efficient algorithm which, given the payoff matrix B and farthest intersection (m,n), maximizes the prize obtained. What is the asymptotic run time of your algorithm in terms of m and n? What is the amount of memory you need (other than the storage of the given matrix B)?