CPSC-320 Intermediate Algorithm Design and Analysis Winter 2000

Homework # 4

Due in class on Wednesday, November 15.

Reminder: you should put at the TOP LEFT corner of your homework: your name (LAST name first), your student ID number, and finally your tutorial section. Put each of these on a separate line. BELOW this information, include information on your collaborations, if any. **Points will be taken off for students who do not provide this information in the above manner.** Your cooperation with this will be greatly appreciated by the TA's.

Some of the following problems ask you to find an algorithm for a problem and explain its running time. For such problems, try to find an algorithm with as fast a running time as possible, ignoring constant factors.

1. Cormen, Leiserson and Rivest, problem 16-4, page 326: Professor McKenzie is consulting for the president of A.-B. Corporation, which is planning a company party. The company has a hierarchical structure; that is, the supervisor forms a tree rooted at the president. The personnel office has ranked each employee with a conviviality rating, which is a real number. In order to make the party fun for all attendees, the president does not want both an employee and his or her immediate supervisor to attend

   (a) Describe an algorithm to make up the guest list. The goal shuld be to maximize the sum of the conviviality ratings of the guests.

   (b) What is the running time of your algorithm? (Use of big-Oh notation is fine in expressing the running time). Explain your answer.

   (c) How can the professor modify the algorithm to ensure that the president gets invited to the party, while maximizing the sum of the conviviality ratings subject to this constraint?

2. You are given two strings of n characters each and an additional parameter k. In each string there are $n - k + 1$ substrings of length k, and so there are $\Theta(n^2)$ pairs of substrings, where one substring is from one string and one is from the other. For a pair of substrings, define the *match count* as the number of opposing characters that match when the two substrings of length k are aligned. The problem is to compute the match-count for each of the $\Theta(n^2)$ pairs of substrings from the two strings.

   Find an algorithm that solves this problem. Describe your algorithm and explain what is its running time.

3. Let x and y be binary strings of length n. The *Hamming Distance* between x and y, denoted by $H(x, y)$, is defined to be the number of places where x and y disagree. That is, if $x = x_1x_2 \dots x_n$ and $y = y_1y_2 \dots y_n$ then $H(x, y)$ is the number of positions i for which $x_i \neq y_i$.

   A S set of binary strings of length n is called a (n, d)-*Hamming set* if the Hamming distance between any pair of strings in S is at least d.

   (a) Give an example of a *maximum-size* Hamming set, with $n = 3$ and $d = 2$.

(b) Following is a greedy algorithm for generating a (n, d)-Hamming set. Give an example for $n = 3$ and $d = 2$ of an ordering of the list M (containing 8 strings) such that the Hamming set returned by this greedy algorithm is *not* of maximum size.

```
Hamming-Set(n,d) {returns a (n,d)-Hamming set}
   initialize M to be a list of the 2^n binary strings
                of length n (in arbitrary order)
   initialize S to be the empty set
   repeat
     let s be the first string in M
     remove s from M and add s to the set S

     for each string s' in M do
         if H(s,s') < d then remove s' from M
   until M is empty
   return S
```

(c) For the algorithm of part (b), show that in the worst case $O(2^n m)$ operations are done (where the test H(s,s') $<$ d counts as one operation).

4. I recently found the following variant of the greedy algorithm of the previous problem in a 1997 U.S. patent (patent number 5,604,097, "Methods for Sorting Polynucleotides Using Oligonucleotide Tags" by S. Brenner):

```
Hamming-Set(n,d)
   initialize M to be a list of the 2^n strings
                of length n (in any order)
   initialize i to be 1
   repeat
      set M-old to be equal to the list M
      let s be the ith string in M
      for each string s' occurring after string s in the M's list, do
         if H(s,s') < d then remove s' from M
      i <-- i+1
   until (M-old = M) or (i > size of M)
   output M
```

Describe an input (n, d) and an initial list M on which this variant outputs a set which is *not* a (n, d)-Hamming set?