CS 255 **Spring 2014**
Analysis of Algorithms

Assignment Five

Please hand in the solutions to the following problems on Thursday, April 24, 2014.

Problem 1 Nuts and Bolts

We have a collection of n bolts of different sizes, and n corresponding nuts. We can test whether a nut and bolt are an exact match or whether the nut is too large or too small for the bolt. We can not directly compare two nuts or directly compare two bolts since the differences in sizes between 2 nuts or 2 bolts are too small to see by the naked eye.

(a) Give an algorithm that correctly pairs up the n nuts and bolts in $O(n^2)$.
(b) We are interested in finding the smallest bolt and its corresponding nut.
    i) What is the minimum number of comparisons needed (best case scenario)
    ii) What is the maximum number of comparisons needed (worst case scenario)

Problem 2

There are only two buttons inside an elevator in a building with 50 floors. The elevator goes 11 floors up if the first button is pressed and 6 floors down if the second button is pressed.
Is it possible to get from floor 32 to floor 33?
a) If it is possible:
    i) What is the minimum number of buttons one has to press?
    ii) What is the shortest time one needs to get from floor 32 to floor 33 (where time is proportional to the number of floors that are passed on the way.
b) If it is not possible, explain why not.
[An Introduction to Bioinformatics Algorithms by Jones and Pevzner, 2004]

Problem 3

Use backtracking and the second method we covered in class to solve the following problem instance of the 0/1 knapsack problem.
We have 4 objects with weights 13, 5, 7, 4, and profits 19, 8, 7, 5 and knapsack capacity 16.

Show all your steps and draw the state space tree.

<u>Problem 4</u> Contiguous Subsequence with Maximum Sum Problem

Problem 6.1 page 191 [DPV_2006].
Use the hint they suggest in the book (that you can find at Canvas).
Given: List of n positive and negative numbers: $a_1, a_2, a_3, \ldots, a_n$

Let A(k) be the maximum sum obtained by considering contiguous subsequences which end with the k^{th} element of S: a_k.

Start by expressing A(k) in terms of A(k-1), then solve the problem; in other words, then give a linear-time algorithm that finds a contiguous subsequence of maximum sum. Note: a subsequence of length zero has a sum equal to zero.

<u>Problem 5</u> The Yuckdonald's Problem

Problem 6.3 page 192 [DPV_2006].
a) Carefully read and understand the problem statement.
Consider the following problem instance:
Locations: $m_1 = 10$ miles, $m_2 = 20$ miles, $m_3 = 25$ miles, $m_4 = 30$ miles, and $m_5 = 40$ miles, with k = 10 miles, and
Profits: $p_1 = 100$, $p_2 = 100$, $p_3 = 110$, $p_4 = 100$, and $p_5 = 100$.
Show that the greedy technique, which consists in choosing the locations in decreasing order of profit, does not yield the optimal solution.

b) To find an efficient algorithm to compute the maximum expected total profit, subject to the given constraint (any two restaurants should be at least k miles apart), we first define a recurrence relation, as is usually the case with Dynamic Programming.

Let P(j) be the maximum profit obtained by considering the first j locations.

Note that in this problem, we cannot merely write P(j) in terms of P(j-1) since location j might not satisfy the constraint condition: "Any two restaurants should be at least k miles apart".

For example, in the problem instance of part a), we cannot choose the second and third locations, since $m_2 = 20$ and $m_3 = 25$; in other words, locations 2 and 3 are not at least 10 units apart. So, let us define prev[j] to be the <u>largest</u> index i, to the left of location j, and at least k miles from j; in other words, $m_i \leq m_j - k$.

Note that prev[j] should be 0 if no such index (i.e., index i) can be found.

1) Express P(j) in terms of P(j-1), with the help of prev[j].

2) Give an efficient algorithm to compute the maximum expected total profit subject to the given constraints.

Problem 6 [Assuming we cover this chapter by April 24, 2014]

Exercise 34.4-2 page 1085. See pages 10 to 12, Part Two of NP-Completeness (Lecture Notes).

$$\phi = ((x_1 \rightarrow x_2) \vee \neg((\neg x_1 \leftrightarrow x_3) \vee x_4)) \wedge \neg x_2$$

Note that formula 34.3 is on page 1082.

Hint: Show all the intermediate steps.

In other words, show ϕ', ϕ'' and ϕ'''.