CS 362, HW3

Prof. Jared Saia, University of New Mexico

Due: March 9th

1. You are on an island where each inhabitant is either a knight or a
   knave. When asked a question, knights always tell the truth, but
   knaves may either lie or tell the truth. On this island, you can only
   ask questions of the form: “Hey Person X: what is Person Y’s type?”
   Each person always answers a question about someone else in the same
   way, so there’s no reason to ask the same question twice.

   (a) Assume there are \( n \) people on the island, and a strict majority of
       them are knights. Describe an efficient algorithm to identify the
       type of each person. Hint 1: You can do this in \( o(n^2) \) questions.
       Hint 2: First try to identify 1 knight. Use recursion!

   (b) Prove that if at most half the inhabitants are knights, then it is
       impossible to solve the problem.

2. Prove by induction that any full parenthesization of \( n \) matrices has
   exactly \( n - 1 \) pairs of parenthesis.

3. Find the optimal parenthesization for a matrix-chain product whose
   sequence of dimensions is: \( (4, 2, 5, 1, 3) \). (Don’t forget to include
   the table used to compute your result)

4. A bakery sells donuts in boxes of three different quantities, \( x_1, x_2, \) and
   \( x_3 \). In the Donut Buying problem, you are given the numbers \( x_1, x_2, \) and
   \( x_3, \) and an integer \( n \) and you should return either 1) the minimum
   number of boxes needed to obtain exactly \( n \) donuts if this is possible,
   along with a set of boxes that obtains this minimum; or 2) “DOH!” if
   it is not possible to obtain exactly \( n \) donuts.

   For example if \( x_1 = 4, x_2 = 6, x_3 = 9 \) and \( n = 17 \), then you should
   return that 3 boxes suffices, with 2 boxes of size 4, and 1 box of size 9.
   However, if \( n = 11 \), you should return “DOH!” since it is not possible
   to buy exactly 11 donuts with these box sizes.
(a) For any positive $x$, let $m(x)$ be the minimum number of boxes needed to buy $x$ donuts if this is possible, or INFINITY otherwise. Write a recurrence relation for the value of $m(x)$. Don’t forget the base case(s)!

(b) Give an efficient algorithm for solving Donut Buying. How does its running time depend on $x_1$, $x_2$, $x_3$, and $n$? Is it an algorithm that runs in polynomial time in the input sizes?

5. A thief repeatedly robs the same bank. To avoid capture, he decides to never rob the bank fewer than 10 days after the last robbery. He has obtained information, for the next $n$ days, on the amount of money $b_i$ that is held at the bank on day $i$.

(a) Let $r(i)$ be the maximum amount of revenue the thief can safely obtain from day 1 through day $i$. Give a recurrence relation for $r(i)$.

(b) Describe a dynamic program based on this recurrence. What is the runtime of your algorithm?

6. Now consider the above problem when (1) for each day, there is an integer value giving the amount of work $w_i$ that the thief must perform to rob the bank on that day (due to the amount of security on that day); and (2) there is an additional constraint that the sum of work the thief ever performs is less than some value $W$. Let $r(i,j)$ be the maximum amount of revenue obtainable on days 1 through $i$, with at most $j$ total work.

(a) Give a recurrence relation for $r(i,j)$.

(b) Describe a dynamic program based on this recurrence. What is the runtime of your algorithm?

7. 

**Greed and Chocolate** You have a chocolate bar consisting of $n$ chunks. Each chunk $1 \leq i \leq n$ has some positive value $v_i$ (for example, chunks with high nougat content are more valuable!)

You must break the bar into exactly $k$ parts to share with your friends, where each part consists of some number of contiguous, unbroken chunks. The value of a part is the sum of the value of all chunks in that part. Your (greedy) friends chose their parts first, and you get the part remaining, i.e, the part of smallest value. Thus, your goal is
to break in such a way that you maximize the value of the minimum value part.
Give a dynamic program to solve this problem. What is the runtime of your algorithm?