TCSS 343 - Week 6

Jake McKenzie

November 18, 2018

More problems in Dynamic Programming and Greedy Algoritms

"A distributed system is when some computer I don't know about fails and causes my computer to fail."
...
Leslie Lamport

"Pain is inevitable, suffering is optional."
...
Haruki Murakami

Consider the following program:

```
program M;
   var m,n : integer;
   var a : array [1..2] of integer;

   procedure P(x,y)
      var m : integer;
   begin
      m := 1; n:= 2;
      a[m] := 4;
      x := x + 2;
      y := y + 5;
   end;

begin
   a[1] := a[2] := m := 2;
   n := 1;
   P(a[m],a[n]);
   print(a[1],a[2]);
end
```

What values will be printed by this program using

(a) (3 points) call by *value*,
(b) (4 points) call by *name*, and
(c) (3 points) call by *reference*?

Briefly explain your answers.

Prove that if $n \in \mathbb{Z}$ and $5n + 4$ is even, then n is even using proof by contradiction. (**Motivation:** Proof by contradiction is often used in proving greedy algorithms correct.)

In mathematics, a sequence of positive real numbers s_1, s_2,... is called *superincreasing* if each element in the sequence is greater than the sum of all previous elements in the sequence:

$$s_{n+1} > \sum_{i=1}^{n} s_i$$

For example: $\{2, 3, 7, 16, 65, 321, 4546\}$ is a superincreasing sequence, but $\{1, 1, 2, 5, 15, 52, 203, 877\}$ us not a superincreasing sequence.

Describe an algorithm that takes as input superincreasing sequence $s_1, \ldots, s_n$ and a positive integer k, please find a sequnce of $s_1, \ldots, s_n$ with the sum equal to k. It is possible and desirable to find an algorithm that can accomplish this task in $O(n)$ time using dynamic progrmaming. If you think you've come up with an algorithm that can accomplish this task attempt to prove that it is correct.

The Knapsack Problem: Given an integer K and n items of different sizes such that the i^{th} item has an integer size of k_i, find a subset of the items whose sizes sum to exactly K, or determin no such subset exists.

Knapsack(S,K):

Input: S(an array of size n storing the sizes of the items)

Output: P(a two dimensional array such that P[i,k].exist() = true if there exists a solution to the knapsack problem with the first i elements and a knapsack of size k, and P[i,k].belong = true if the i^{th} element belongs to the solution)

Give an algorithm that solves this problem given these input and output parameters.