please grade

CSE 548 – Analysis of Algorithms, Spring 2013

Assignment #2c

Duo Xu (#108662210)

partner: Yu-Yao Lin (#109090793)

Problem 10

(1) Subproblems:

for $k = 0$ to K

for $i = 1$ to n

Solve the knapsack problem for target integer k and set $\{s_1, s_2, \cdots, s_i\}$.

Pseudocode:

$B[0, 0] = 1$

for $k = 1$ to K

$B[0, k] = 0$

for $i = 1$ to n

for $k = 0$ to K

$B[i, k] = B[i-1, k] \vee B[i-1, k-s_i]$

return $B[n, K]$

(2) After filling out B, we can backtrack to find one possible set T.

Pseudocode:

$T = []$

if $(B[n, K] == 1)\{$

$i = n$

$k = K$

while $(i > 0 \&\& k > 0)\{$

if $(B[i-1, k-s_i] == 1)\{$

$T = T \vee s_i$

$k = k - s_i$

$\}$

$i = i - 1$

$\}$

$\}$

return T

Problem 11 *Problem from Steve Skiena*

Consider the problem of storing n books on shelves in a library. The order of the books is fixed by the cataloging system and so cannot be rearraged. Therefore, we can speak of a book b_i, where $1 \leq i \leq n$, that has a thickness t_i and height h_i. The length of each bookshelf at this library is L. Suppose all the books have the same height h (i.e. $h = h_i = h_j$ for all i, j) and the shelves are all separated by a distance of greater than h, so any book fits on any shelf. The greedy algorithm would fill the first shelf with as many books as we can until we get the smallest i such that b_i does not fit, and then repeat with subsequent shelves. Show that the greedy algorithm always finds the optimal shelf placement, and analyze the time complexity.

Problem 12 *Problem from Steve Skiena*

This is a generalization of the previous problem. Now consider the case where the height of the books is not constant, but we have the freedom to adjust the height of each shelf to that of the tallest book on the shelf. Thus the cost of a particular layout is the sum of the heights of the largest book on each shelf.

(1) Give an example to show that the greedy algorithm of stuffing each shelf as full as possible does not always give the minimum overall height.

(2) What technique should we use to solve this problem?

(3) What are the subproblems?

(4) How many subproblems are there?

(5) Give an algorithm for this problem, and analyze its time complexity.

Problem 13

Suppose you are given three strings of characters: X, Y, and Z, where

```
|X| = n, |Y| = m, and |Z| = n+m.
```

Z is said to be a shuffle of X and Y iff Z can be formed by interleaving the characters from X and Y in a way that maintains the left-to-right ordering of the characters from each string.

(a) Show that cchocohilaptes is a shuffle of chocolate and chips, but chocochilatspe is not.

(b) Give an efficient dynamic-programming algorithm that determines whether Z is a shuffle of X and Y.

Hint: The values the dynamic programming matrix you construct should be Boolean, not numeric.

Problem 14

Here is the strategy:

Call the number you saw "x". Use the logistic function to calculate $p(x) = 1/(1+e^{-x})$. Choose a random real number between 0 and 1. If it's lower than $p(x)$, guess "higher". Otherwise, guess "lower".

In other words, guess "higher" with probability $p(x)$, and "lower" with probability $1 - p(x)$.

This works because $p(x)$ is a function which is 0 at negative infinity, 1 at positive infinity, and increases monotonically in between. So for any pair of numbers, if you call the smaller one A and the larger one B, you have a (slightly) better chance of guessing higher if you saw B than if you saw A. Your overall chances of guessing correctly given that there is a 50% chance youre seeing A and a 50% chance youre seeing B are:

$P = 0.5 * (1 - p(A)) + 0.5 * p(B)$

Since $p(A)$ is always smaller than $p(B)$, P is always greater than 50%.

Reference: `http://blog.xkcd.com/2010/02/09/math-puzzle`