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

Assume greedy is not optimal solution.

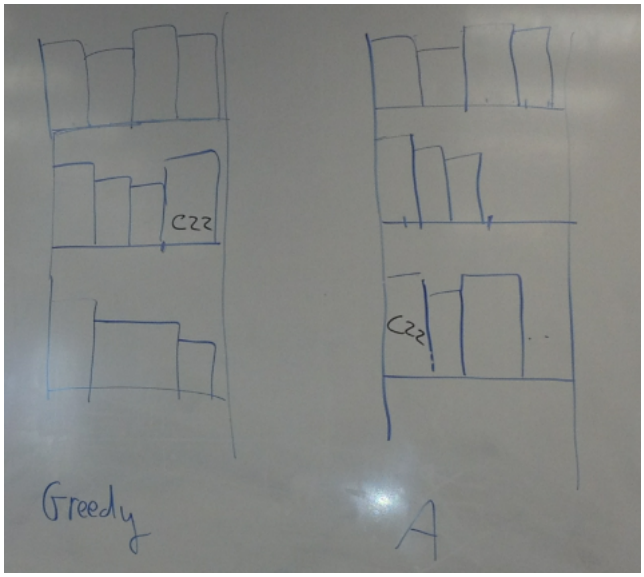
Then there exists an algorithm A which is better than greedy.

Consider A is the same as greedy for longest prefix of books. That is greedy and A has some prefix which is similar. By this we mean longest number of shelves.

Now we make A' which has C22 in above shelf. A' is the same as A but it agrees with greedy for 1 more book. Because there are still enough space to put C22 in above shelf, in order to keep "longest prefix" true, we should put C22 in above shelf, which causes a contradiction.

Thus greedy is the optimal solution.

The time complexity is $O(n)$.



Problem 12

(1) Consider there are 3 books. b_1 with $t_1 = 1$ and $h_1 = 1$, b_2 with $t_2 = 1$ and $h_2 = 2$ and b_3 with $t_3 = 1$ and $h_3 = 2$. The length L of the shelf is 2. So greedy will put b_1 and b_2 in the first shelf and b_3 in the second shelf. The total cost is 4. However, the optimal solution is to put b_1 in the first shelf while b_2 and b_3 are in the second shelf. The total cost is 3.

(2) Dynamic programming.

(3) We define subproblems by going backwards. Assuming there are n books, subproblems are what is the minimum overall height of the bookshelf layout if we only place books i through n onto the shelves.

(4) As there are n books, thus there are n subproblems.

(5) We define $cost[i]$ as the minimum overall height of placing books i through n onto the shelves. There are generally two choices: one is to put the book on the current shelf and the other is to put the book on the new shelf. For the first case, if placing the current book on the current shelf exceeds the length of the shelf, we say L, then we only have the choice to put it on the new shelf.

$cost[i] = min\{cost[i+k] + max\{H[i], .., H[i+k-1]\}\}$ where $1 \leq k \leq m$

$H[i]$ stands for the height of the i_{th} book. $T[i]$ stands for the thickness of the i_{th} book. m satisfied $\sum_{q=i}^{m} T[q] \leq L$ for each i, which stands for the maximum index of the book to place on the same shelf starting from the i_{th} book.

The base case is $cost[n] = H[n]$, which means only placing the n_{th} book onto the shelf will only cost the height of the n_{th} book.

Thus $cost[1]$ represents the final solution, the minimum cost of placing books 1 through n onto the shelves.

Pseudocode:

$cost[n] = H[n]$
for $i = n - 1$ downto 1
  $currentlen = T[i]$
  for $j = i + 1$ to n
    $currentlen = currentlen + T[j]$
    if $(currentlen \leq L)$ and $(max\{H[i], ..., H[j]\} + cost[j+1] \leq cost[i])$
      $cost[i] = max\{H[i], ..., H[j]\} + cost[j+1]$
return $cost[1]$

Assuming $max\{H[i], ..., H[j]\}$ is $O(1)$ time, inside the outer loop, we can keep a variable to track the largest height between $H[i]$ and $H[j]$, then the time complexity is $O(n^2)$.

Problem 13

(a) cchhilaptes.

In chocochilatspe, "sp" is in different order from "ps" in chips.

(b) We can create a $(n+1)*(m+1)$ matrix to store the state (true or false). $M[i][j]$ represents whether the prefix string $Z[1...i+j]$ is a shuffle of prefix $X[1...i]$ and prefix $Y[1...j]$.

The equation is $M[i][j] = ((Z[i+j] == X[i])\&\&M[i-1][j])||((Z[i+j] == Y[j])\&\&M[i][j-1])$. It means if Z is a shuffle of X and Y, one case is the last char of Z equals X's last char and whether $Z[1...i+j-1]$ is a shuffle of $X[1...i-1]$ and $Y[1...j]$, and the other case is the last char of Z equals Y's last char and whether $Z[1..i+j-1]$ is a shuffle of $X[1..i]$ and $Y[1...j-1]$. Either of the two is OK.

Pseudocode:

```
boolean M[n+1][m+1]
for i = 0 to n
    // M[i][0] means whether Z[1...i] is a shuffle of prefix X[1...i], which is definitely true
    M[i][0] = true
for j = 0 to m
    // M[0][j] means whether Z[1...j] is a shuffle of prefix Y[1...j], which is definitely true
    M[0][j] = true
for i = 1 to n
    for j = 1 to m
        M[i][j] = ((Z[i+j] == X[i])&&M[i-1][j])||((Z[i+j] == Y[j])&&M[i][j-1])
return M[n][m]
```

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`