CSCI-UA.0310-004/005 Basic Algorithms April 15, 2015

Solutions to Problem 1 of Homework 9 (12 Points)

Name: Jason Yao *Due: Wednesday, April 15*

You want to travel on a straight line from from city A to city B which is N miles away from A. For concreteness, imagine a line with A being at 0 and B being at N. Each day you can travel at most d miles (where $0 < d < N$), after which you need to stay at an expensive hotel. There are n such hotels between 0 and N, located at points $0 < a_1 < a_2 < \ldots < a_n = N$ (the last hotel is in B). Luckily, you know that $|a_{i+1} - a_i| \leq d$ for any i (with $a_0 = 0$), so that you can at least travel to the next hotel in one day. You goal is to complete your travel in the smallest number of days (so that you do not pay a fortune for the hotels).

Consider the following greedy algorithm: "Each day, starting at the current hotel a_i, travel to the furthest hotel a_j s.t. $|a_j - a_i| \leq d$, until eventually $a_n = N$ is reached". I.e., if several hotels are within reach in one day from your current position, go to the one closest to your destination.

(a) (6 points) Formally argue that this algorithm is correct using the "Greedy Stays Ahead" method.
(**Hint**: Think how to define $F_i(Z)$. For this problem, the name of the method is really appropriate.)

Solution: We define $F_i(z)$ =distance from B

$F_i(z) = N - a_i$

Show for each step i, $greedy_{ai}$ will be further out or equal to OPT_{ai}

Base case: By the definition of the greedy algorithm, we see that $greedy_{a1} \geq OPT_{a1}$

Induction step: Assuming $greedy_{a1}$ is already ahead from a_1, and we continue going d to the next step, $greedy_{ai} \geq OPT_{ai}$.

Since the distance traveled is $\leq d$, $greedy_{a(i+1)} \geq OPT_{a(i+1)}$ □

(b) (6 points) Formally argue that this algorithm is correct using the "Local Swap" method. More concretely, given some hypothetical optimal solution Z of size k and the solution Z^* output by greedy, define some solution Z_1 with the following two properties: (1) Z_1 is no worse than Z; (2) Z_1 agrees with greedy in the first day travel plan. After Z_1 is defined, define Z_2 s.t.: (1) Z_2 is no worse than Z_1; (2) Z_2 agrees with greedy in the first two days travel plan. And so on until you eventually reach greedy.

Solution:

Z = OPT, Z* = Greedy

Z_1 =no worse than Z, so

$Z_1 = Z_1^* + Z_2 + ... + Z_k$

$Z_2 = Z_1^* + Z_2^* + Z_3 + ... + Z_k$

$Z_p = Z_1^* + Z_2^* + ... + Z_p^* + Z_{p+1} + ... + Z_k$, for p: $1 \leq p \leq k$.

Thus, when p == k, $Z_k = Z_1^* + Z_2^* + ... + Z_k^*$

$\square$

CSCI-UA.0310-004/005 Basic Algorithms April 15, 2015

Solutions to Problem 2 of Homework 9 (10 points)

Name: Jason Yao *Due: Wednesday, April 15*

Recall, Fibonacci numbers are defined by $f_0 = f_1 = 1$ and $f_i = f_{i-1} + f_{i-2}$ for $i \geq 2$.

(a) (2 points) What is the optimal Huffman code for the following set of frequencies which are the first 8 Fibonacci numbers.

Solution:

□

(b) (4 points) Let $S_1 = 2 = f_0 + f_1$ and $S_i = S_{i-1} + f_i = \ldots = f_i + f_{i-1} + \ldots f_1 + f_0$ (for $i > 1$) be the sum of the first i Fibonacci numbers. Prove that $S_i = f_{i+2} - 1$ for any $i \geq 1$.

Solution:

Show $S_i = S_{i-1} + f_i = f_i - +f_{i-1} + ... + f_1 + f_0 = f_{i+2} - 1$

Base case: Let i = 1

$S_1 = f_1 + f_0 = f_3 - 1$

$S_1 = 1 + 1 = 3 - 1$

$S_1 = 2 = 2$

Assume for all S_i is true

$$S_{i+1} = (S_i + f_{i+1})$$
$$S_{i+1} = (f_{i+2} - 1) + f_{i+1}$$
$$S_{i+1} = f_{i+3} - 1$$

□

(c) (4 points) Generalize your solution to part (a) to find the shape of the optimal Huffman code for the first n Fibonacci numbers. Formally argue that your tree structure is correct, by using part (b).

Solution:

By part b, we have shown that for all S_i, in which $i \geq 2$, S_i has two children, namely S_{i-1} and f_i. Thus the tree structure is that of a very long, unbalanced tree that slopes in one direction. Assuming that this is a BST, it would slop down to to the left, since f_i would be on the right, and S_{i-1} the child on the left. □

CSCI-UA.0310-004/005 Basic Algorithms April 15, 2015

Solutions to Problem 3 of Homework 9 (12 points)

Name: Jason Yao *Due: Wednesday, April 15*

You have n items of weights $w_1, \dots, w_n$ that you want to pack. You can use boxes of capacity u each. (You can assume that $w_i \leq u$ for all i). You are allowed to put *at most two* items in a box (even if more than two could fit!). Your goal is to pack all n items using the *smallest possible* total number of boxes. Consider the following idea for a greedy solution:

(1) Take the lightest remaining item a; (2) find the heaviest item b (if it exists) that will fit in one box with a; (3) Pack these two elements a and b into one box (if no b exists, just pack a); (4) repeat steps (1)-(3) with the remaining items.

(a) (6 points) Assume that the input weights are already sorted: $w_1 \leq w_2 \leq \ldots \leq w_n$. Write the fastest pseudocode you an find to implement the above described solution. (You will get full credit only for an $O(n)$ algorithm.)

Solution:

```
int weights[];
int done[];
int capacity = u;

int greedyPack()
{
        int max = done.length;
        for (i = 1 to n)
                if (i == max || i == 0)
                        break;
                if (done[i] != 0)
                        for (j = max to 1)
                                if (weights[i] + weights[j] <= capacity)
                                        done[i], done[j] = 0;
                                        pack(weights[i], weights[j]);
                                        max = j - 1;
                                        break;
                                endif
                        endfor
                endif
        endfor
        for (k = 1 to n)
                if (done[k] != 0)
                        pack(weights[k]);
                endif
        endfor
}
```

□

(b) (6 points) Prove the correctness of this algorithm using Local Swap technique.
(**Hint**: Think of the box containing the lightest item a in the optimum solution.)

Solution:

For the general case:

Let greedy algorithm be a + b, let OPT algorithm be x + y

Show that all swaps of local hybrids are valid:

case 1:

(a, b), (x, y) → (x, b), (a, y) a swaps with x

(x, b), (a, y) → (a, b), (x, y) b swaps with y

Show that (x, b) and (a, y) are valid hybrids:

$a + b \leq u \geq x + y$

$a + b + x + y \leq 2u$

a is the lightest item, and b is the heaviest item that still fits.

Thus, $y \leq b$.

$a + b + x + b \leq 2u$

$x + b \leq u$

If $x + b \leq u$, then from $a + b + x + b \leq 2u$,

$a + y \leq u$

Therefore (x, b) and (a, y) are both valid hybrids.

case 2:

(a, b), (x, y) → (y, b), (x, a) a swaps with y

(y, b), (x, a) → (y, x), (b, a) b swaps with x

Show that (y, b) and (x, a) are valid hybrids: $a + b \leq u \geq x + y$

$a + b + x + y \leq 2u$

By symmetry of the proof for case 1, and exchanging the symbols, we show that (y, b) and (x, a) are valid hybrids. □

(c) (**Extra credit**: ? points) Help the authors of this problem and come up with GASA proof for this algorithm.

CSCI-UA.0310-004/005 Basic Algorithms April 15, 2015

Solutions to Problem 4 of Homework 9 (10 points)

Name: Jason Yao *Due: Wednesday, April 15*

Recall the following two PFEs E_1 and E_2 discussed in class. Given a message m of length n (known to encoder), $E_1(m) = 0^n\ 1\ m$; $E_2(m) = 0^{\lceil \log_2 n \rceil} 1\ \langle n \rangle_2\ m$, where $\langle n \rangle_2$ is the binary representation of integer n. Namely, E_1 encodes in unary the length of the string and then the actual string, while the better encoding E_2 starts with encoding the length n using E_1, and then putting actual string m. Recall also the overhead of E_1 is $n+1$ and the overhead of E_2 is $2\lceil \log_2 n \rceil + 1 = 2\log_2 n + O(1)$.

(a) (5 points) Extend the above idea one more iteration to design an encoding E_3 having overhead of $\log_2 n + 2\log_2 \log_2 n + O(1)$.

Solution:

$E_3(m) = 0^{\log_2 \log_2 n} 1 < \log_2 n >_2 < n >_2 m$

Let H = overhead,

$H = \log_2 n + 2\log_2 \log_2 n + O(1)$ □

(b) (5 points) Now write a fully recursive procedure (instead of stopping it after a constant number of iterations) to get an encoding E^* having overhead[1]

$$\log_2 n + \log_2 \log_2 n + \log_2 \log_2 \log_2 n + \ldots + O(1)$$

where the number of terms in the sum is the number of times one can take $\log_2$ of n until one reaches 1 (called inverse Ackerman function).

Solution:

$E_k(m) = 0^{\log_k \ldots \log n} 1 < \log_k \ldots \log_1 n >_2 < \log_{k-1} \ldots \log_1 n >_2 \ldots < \log n >_2 < n >_2 m$

Let H = overhead,

$H = \log_2 n + \log_2 \log_2 n + \ldots + O(1)$ □

[1]Turns out this overhead is the best possible one can achieve in any PFE.