CSCI-GA.1170-001/002 Fundamental Algorithms	November 9, 2014

Solutions to Problem 1 of Homework 8 (3(+5) points)

Name: Sahil Goel *Due: Wednesday, November 12*

For each of the following suggested greedy algorithms for the ACTIVITY-SELECTION problem, give a simple example of the input where the proposed greedy algorithm fails to compute the correct optimal solution.

(a) (3 points) Select the activity a_i with the shortest duration $d_i = f_i - s_i$. Commit to scheduling a_i. Let S'_i consist of all activities a_j which do not overlap with a_i: namely, either $f_j \leq s_i$ or $f_i \leq s_j$. Recursively solve ACTIVITY-SELECTION on S'_i, scheduling the resulting activities together with a_i.

Solution: F is the final activity set initially null F= {}
Suppose the activities are like this and d_i gives the duration of activity s_i

i	1	2	3	4	5
s_i	1	2	5	8	11
f_i	3	5	10	12	16
d_i	2	3	5	4	5

Initially S=$\{a_1, a_2, a_3, a_4, a_5\}$
Now we will choose S with lowest d_i i.e. 2 corresponding to a_1.]
So we add a_1 to F

Now $F = a_1$
S will have those activities of S which doesn't overlap with a_1
S=$\{a_3, a_4, a_5\}$

Now we will choose $\min\{d_3, d_4, d_5\} = 4$ hence a_4

Now $F = \{a_1, a_4\}$
S = {}
Hence total number of activities we can perform according to this greedy approach will be equal to 2

But if we apply greedy approach where we select the activity which finishes first then total number of activities we can perform are 3 i.e $\{a_1, a_3, a_5\}$

Hence the above greedy doesn't gives the optmial solution

□

(b)* **(Extra Credit; 5 points)** For each activity a_i, let n_i denote the number of activities which do not overlap with a_i (e.g., n_i is the cardinality of the set S'_i defined above for specific a_i). Select the activity a_i with the largest number n_i of non-overlapping activities. Commit to scheduling a_i. Recursively solve ACTIVITY-SELECTION on the n_i activities in S'_i, scheduling the resulting activities together with a_i.
(**Hint**: Unlike part (a), you might need a lot of activities for this counter-example. The smallest I know uses $n = 11$ activities. So don't be discouraged if small examples are all bad.)

Solution: Suppose the activities be like:

s_i	1	1	1	1	5	10	11	13	15	19	20
f_i	5	7	8	9	11	13	20	21	21	21	21

According to the correct greedy approach:
Initially $S = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}\}$ and F={}
Initially we will select activity a_1
Now removing all overlapping activities with a_1
$S = a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}$
$F = \{a_1\}$
Now we will select a_5
$F = \{a_1, a_5\}$
Removing all overlapping activities with a_5
$S = \{a_7, a_8, a_9, a_{10}, a_{11}\}$
Now we will select a_7
$F = \{a_1, a_5, a_7\}$
Removing all overlapping activities with a_7
$S = \{a_{11}\}$
Now adding final activity to F
$F = \{a_1, a_5, a_7, a_{11}\}$
Hence total number of activities in optimal solution is 4

Now if we consider the approach in which we select the activity which intersects with least number of other activities

s_i	1	1	1	1	5	10	11	13	15	19	20
f_i	5	7	8	9	11	13	20	21	21	21	21
n_i	7	6	6	6	6	8	6	6	6	6	7

Initially $S = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7, a_8, a_9, a_{10}, a_{11}\}$ and F={}
Now we will select a_6 first since n_6 is maximum
$F = a_6$
Removing all activities from S which overlaps with a_6
$S = \{a_1, a_2, a_3, a_4, a_8, a_9, a_{10}, a_{11}\}$
Now we will select a_1 since n_1 is 7 which is maximum
$F = a_1, a_6$
Removing all overlapping activities

$S = \{a_8, a_9, a_{10}, a_{11}\}$
Now we select a_{11} as $n_11 = 7$ which is maximum
$F = a_1, a_6, a_11$
Now removing all overlapping activities
S={ }
Hence the solution we get from this approach only has 3 activities whereas the solution we got from correct greedy approach was 4. Hence this greedy doesn't give the optimal solution

$\square$

CSCI-GA.1170-001/002 Fundamental Algorithms	November 9, 2014

Solutions to Problem 2 of Homework 8 (11 Points)

Name: Sahil Goel *Due: Wednesday, November 12*

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 rearranged. The i-th book b_i, where $1 \leq i \leq n$ has a thickness t_i and height h_i stored in arrays $t[1 \ldots n]$ and $h[1 \ldots n]$. The length of each bookshelf at this library is L. We want to minimize the sum of heights of the shelves needed to arrange these books.

(a) (5 points) Suppose all the books have the same height h (i.e., $h = h_i$ for all i) and the shelves are each of height 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. Using either the Greedy Always Stays Ahead or Local Swap method, show that the greedy algorithm always finds the shelf placement with the smallest total height of shelves, and analyze its time complexity.

Solution: Using local swap method
Suppose there exists an optimal solution Z_o which gives lesser total height than the greedy solution Z_k
Z_o: Stores $\{n1_1, n1_2, \ldots, n1_k\}$ Where $n1_i$ represents number of books on i^{th} shelf according to the optimal solution.
Z_k: Stores $\{n2_1, n2_2, \ldots, n2_k\}$ Where $n2_i$ represents number of books on i^{th} shelf according to the greedy solution.

According to the greedy algorithm, we will choose as many books as we can until we find a b_i such that it doesn't fit on it. In optimal solution the number of books on first shelf if $n1_1$ and according to greedy the number of books on first shelf are $n2_1$. We know that $n2_1 \geq n1_1$ since we have chosen the maximum number of books that can be arranged on first shelf. Also since height of every book is same, adding more books won't make any difference to the height of that shelf. Therefore if we replace $n1_1$ in optimal solution set with $n2_1$, it won't worse the solution. Since total number of books are n, according to optimal solution set, the number of books remaining after first shelf is n-$n1_1$, whereas total number of books remaining if we choose $n2_1$ books in the first shelf are n-$n2_1$. Since $n2_1 \geq n1_1$, therefore total number of remaining books after first shelf will be equal to or lesser than the optimal solution. Hence it hasn't worsen the solution and we are still on optimal state.

Now we have $Z_1 = \{n2_1, n1_2, n1_3, \ldots, n1_k\}$
Now lets choose number of books on second shelf according to the greedy approach. We will again choose the maximum number of books that can be arranged on this shelf. Since adding more books will not affect the height of the shelf as height of each book is same. We know that $n2_2 \geq n1_2$, since we have chosen the maximum number of books that can be arranged

on this shelf. Observe that $n1_2$ can't be greater than $n2_2$. It is possible only when there is a very thick book in $n2_2$ which doesn't exist in $n1_2$ which is not possible, because if it doesn't exist in $n1_2$, then it must have existed in $n1_1$ but since we choose the maximum number of books in $n2_1$, therefore that case can't arise. Hence we can easily assume that $n2_2 \geq n1_2$. Since $n2_1 \geq n1_1$ and $n2_2 \geq n1_2$, hence $n2_1 + n2_2 \geq n1_1 + n1_2\dot{}$Hence total number of remaining books after step 2 i.e. $n - n2_1 - n2_2 \leq n - n2_1 - n1_2$. Hence this step has also not worsen the solution. Therefore this solution is still optimal

Continuing in this fashion till replacing with $n1_k$ with $n2_k$ will give the final greedy answer. At every step, since the solution from greedy as at least as good as optimal solution, the final solution will also be as good as optimal solution. Since we can't attain better than the optimal solution(Otherwise it won't have been a optimal solution), after replacing all the terms of optimal solution with greedy solution will give the optimal solution.

Time complexity of this solution will be O(n), since we just need to traverse the array once and check for every book if it fits on the current shelf or not. If it fits we place it there otherwise we place the book on the second shelf and continue with remaining books. Since we only need to traverse the array once time complexity is O(n)

□

(b) (6 points) Now assume that the books are not of the same height, and hence the height of any shelf is set to be the height of the largest book placed on that shelf. Show that the greedy algorithm in part (a) doesn't work for this problem. Give an alternative dynamic programming algorithm to solve this problem. What is the running time of your algorithm?

Solution: Suppose length of the shelf is 60cm
Thickness of each book is 20cm and height of books are as follows {20cm,30cm,50cm,40cm}.
If we follow the greedy approach from first part the solution we will arrive at it:
2nd shelf: b_4
1st Shelf: b_1, b_2, b_3
Total height will be = maxheight(b_1, b_2, b_3) + maxheight(b_4)
Total height H = 50 + 40 = 90 cm

But we can easily observe that the optimal solution will be
2nd shelf: b_2, b_3, b_4
1st Shelf: b_1
Total height will be = maxheight(b_1) + maxheight(b_2, b_3, b_4)
Total height H=20 + 50 = 70cm.
Hence our greedy approach doesn't give the optimal solution in this case

Using dynamic algrorithm to find the optimal solution. Suppose H[j] gives the total optimal height of arranging j books

mh:- maxheight

H[i] $= mh(b_1, b_2, \ldots, b_i)$ If all books from 1 to i fit on one shelf
H[i] $= min(mh(b_i) + H[i-1], mh(b_i, b_{i-1}) + H[i-2], \ldots, mh(b_i, \ldots, b_k) + H[i-(k+1)])$
Where k is minimum such that all books b_k to b_i fit on one shelf i.e. b_{k-1} to b_i won't fit on one shelf.

Algorithm
{
Create a 1-D array H. H_i will represent the optimal height of shelf till i books.
For i=1:n
If b_1 to b_i fit on one shelf i.e (t[1] + t[2] + ..+t[i] $\leq$ L)
else
H[i] $= min(mh(b_i) + H[i-1], mh(b_i, b_{i-1}) + H[i-2], \ldots, mh(b_i, \ldots, b_k) + H[i-(k+1)])$
% Here k is the minimum such that b_k to b_i fit on one shelf
end For
return H[n]
}

Time Complexity analysis
Since for every i, we have i-k cases such b_k to b_i fit on one shelf. Since i runs from 1 to n, total time complexity will be O(n*(k)). Since we know that k will be equal to cn for some c, hence k=O(n). Hence total time complexity of the solution will be
O(n*n) = O(n^2)

□

CSCI-GA.1170-001/002 Fundamental Algorithms	November 9, 2014

Solutions to Problem 3 of Homework 8 (12 Points)

Name: Sahil Goel *Due: Wednesday, November 12*

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: Let $F_i(Z)$ be a function which represents the distance remaining from target after i days (in solution set Z)
Z represents a set which represents the hotel number in which we stayed
Let Z_g be the solution set for greedy algorithm and Z_o be the solution set for any other algorithm which may be optimal

Base case for i=1(Comparing $F_1(Z_g)$ and $F_1(Z_o)$
According to the greedy approach mentioned, we know that we will travel to the farthest hotel we can from starting position. Therefore if we stayed in hotel h_i according to Z_g and stayed in hotel h_j according to Z_o then h_i will be nearer to the target than h_j or $h_i = h_j$. Therefore $F_1(Z_g) \leq F_1(Z_o)$. Hence after first day we will be nearer to target or from equal distance from target using greedy approach as compared to any other solution.

Now suppose it is true till i-1 i.e. $F_{i-1}(Z_g) \leq F_{i-1}(Z_o)$, we have to prove for i
Now after (i-1) days, suppose we choose a hotel h_j according to Z_o which is more closer to target than Z_g i^{th} hotel. But we know that $F_{i-1}(Z_g) \leq F_{i-1}(Z_o)$ that is after i-1 steps we were nearer or at equal distance from target according to greedy approach as compared to any other solution. Then after $(i-1)^{th}$ day, we would have travelled maximum we can according to greedy and stayed at some hotel h_i which is farthest away from h_{i-1} will which we can go. Hence h_i will be more nearer to target or at equal distance to target as is h_j. h_j can't be ahead of h_i since h_{j-1} was behind or equal to h_i. Therefore $F_i(Z_g) \leq F_i(Z_o)$

Hence this greedy approach will eventually give the optimal solution

$\square$

(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: Suppose Z is an hypothetical solution which is optimal.
Z contains a set of hotels $\{h_{i1}, h_{i2}, \ldots, h_{ik}\}$in which we will stay
Here $i1 < i2 < i3 < \ldots < ik$ i.e. i1 comes before i2 which comes before i3 and so on

According to the greedy approach, the first hotel we will choose is h_{j1} where h_{j1} is the farthest hotel which can be reached in first day. Therefore h_{j1} must exist after h_{i1} (or h_{j1} must be equal to h_{i1}). Therefore if we swap h_{i1} with h_{j1}, the solution Z_1 will be better than Z or at least equal to Z. It can't become worse because if h_{i2} can be reached from h_{i1} on second day, it can still be reached from h_{j1} on second day as h_{j1} is ahead of h_{i1}

Now for second day (Z_2)
According to greedy approach, we will choose h_{j2} which is the farthest one that can be reached from h_{j1}. Since we know that h_{j1} was either ahead of h_{i1} or may be equal to h_{i1}, therefore h_{j2} will be either ahead of h_{i2} (or may be $h_{i2} = h_{j2}$)
. h_{i2} can't be ahead of h_{j2} since it h_{j2} is the farthest we could go from h_{j1} which was already ahead of h_{i1}. Hence Z_2 is better than or at least equal to Z. It can't be worse because still h_{i3} can be reached from h_{j2} if it can be reached from h_{i2} since h_{j2} is ahead of h_{i2}

Continuing in the same fashion, after k days Z_k will be better than or equal to Z which was the optimal solution. Since Z_k can't be better than the optimal solution(otherwise it won't have been an optimal solution), Z_k gives the optimal solution.

$\square$

CSCI-GA.1170-001/002 Fundamental Algorithms	November 9, 2014

Solutions to Problem 4 of Homework 8 (10 points)

Name: Sahil Goel *Due: Wednesday, November 12*

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:
Huffman codes
1st least frequent: 1: 1111111
2nd least frequent: 1: 1111110
3rd least frequent: 2: 111110
4th least frequent: 3: 11110
5th least frequent: 5: 1110
6th least frequent: 8: 110
7th least frequent: 13: 10
8th least frequent: 21: 0

□

(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: Proof using induction
For base case S_i and i=1
$S_1 = f_0 + f_1$
$S_1 = 1 + 1$
$S_1 = 2$
To prove $f_{1+2} - 1 = 2$
$f_3 - 1 = f_2 + f_1 - 1 = f_0 + f_1 + f_1 - 1 = 1 + 1 + 1$ - 1 = 2
Hence it is true for i=1

now suppose it is true till i-1
to prove for i
$S_i = f_{i+2} - 1$
Since $S_i = S_{i-1} + f_i$
$S_i = S_{i-1} + f_i$
Since it is true till i-1
therefore $S_{i-1} = f_{i-1+2} - 1$
$S_i = f_{i-1+2} - 1 + f_i$
$S_i = f_{i+1} + f_i - 1$
Since $f_{i+2} = f_{i+1} + f_i$
Therefore

$S_i = f_{i+2} - 1$
Hence proved

□

(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: For first n numbers
h_i represents huffman code for ith term of fibonacci sequence
i varies from 1 to n, the general formula will be
$h_i = (n-1)$ 1's if i==1
$h_i = (n-i)$ 1's || 0 if $n \geq i$ >1
|| operators concatenates two strings
As we can observe from the general formula
For every new fibonacci number inserted, the previous tree shifts to one level down and new number is added to the left of the tree. So the tree will be skewed towards right

Proof:
Suppose we have a tree till i-2 terms and we have to insert i-1th and ith term i.e. f_{i-1} and f_i. We know that at the root of the previous tree will be S_{i-2} i.e. sum till previous i-2 terms.
Now we have to select the 2 minimum terms
Three options we have are
f_{i-1}, f_i, S_{i-2}
From part(b) we know that $S_{i-2} = f_i - 1$
So now three options becomes
f_{i-1}, f_i, $f_i - 1$
we know that $f_i > f_i - 1$ and $f_i > f_{i-1}$
Therefore 2 minimum terms will be $f_i - 1 = S_i$ and f_{i-1}
Hence, the previous tree is now joined with the next fibonacci number to give a new tree with height incremented by one. In other words, all the previous elements in the tree shifts down by one level more

Hence the generalized algorithm is correct

□

CSCI-GA.1170-001/002 Fundamental Algorithms	November 9, 2014

Solutions to Problem 5 of Homework 8 (14 Points)

Name: Sahil Goel *Due: Wednesday, November 12*

Little Johnny is extremely fond of watching television. His parents are off for work for the period $[S, F)$, and he wants to make full use of this time by watching as much television as possible: in fact, he wants to watch TV non-stop the entire period $[S; F)$. He has a list of his favorite n TV shows (on different channels), where the i-th show runs for the time period $[s_i, f_i)$, so that the union of $[s_i, f_i)$ fully covers the entire time period $[S, F)$ when his parents are away.

(a) (10 points) Little Johnny doesn't mind to switch to the show already running, but is very lazy to switch the TV channels, and so he wants to find the smallest set of TV shows that he can watch, and still stay occupied for the entire period $[S, F)$. Design an efficient $O(n \log n)$ greedy algorithm to help Little Johny. Do not forget to carefully argue the correctness of your algorithm, using either the "Greedy Always Stays Ahead" or the "Local Swap" argument.

Solution: Greedy Approach: At every step, we will first look for those shows that are possible to run at that time. Among those we will choose the one with the latest finish time. Suppose there are k total shows, we will initialize S set as $\{a_1, a_2, a_3, \ldots, a_k\}$ where a_i represents i^{th} show in set S

Algorithm Explanation
1. Initialize S with all the tv shows
2. Sort the shows by their finish time in ascending order
3. Initialize T as StartTime, i as one and Final as Null Set. T will represent current time in program
4. Store show a_i in a temp variable
5. Increment i. Check now if $i \leq k$ and a_i start time is less than T or not.
(a). If yes, check if this show's finish time is greater than temp finish time. If yes, replace temp with this show, and go back to 5.
(b). If no, add temp to Final and set T to temp's finish time
6. If (T< FinishTime) increment i and go back to 4.
7. Final gives the shortest set to watch shows between [S,F)

Algorithm:
{
$S \;:=\; \{a_1, a_2, a_3, \ldots, a_k\}$
Sort S(by F_i) ;
$T = StartTime$;
i=1;
Final={}; (It will store the final result)
Temp := a_1;
i=i+1;
while($T < FinishTime$)
{
while($startTime(a_i) \leq T$ && i<=k)
{
if($finishTime(a_i) > finishTime[temp]$)
{
temp := a_i;
i=i+1;
}
}
Final = Final union temp
T=$finishTime[temp]$
}
i=i+1;
temp=a_i;
}
}

Proof using local swap
Suppose Z_o is the optimal solution and Z_g is the greedy solution.

1. Initially for the first step, according to greedy approach, we will choose that show which is possible at this time and finishes the last. We replace the first show of optimal solution with show chosen according to greedy approach. Suppose first show in optimal solution was S_{i1} and first show according to greedy approach is S_{j1}, then we know that Finish time of S_{j1} $\geq$ Finish time of S_{i1}. Since we have chosen the show with latest finish time. Now suppose S_{i1+} represents all the shows that are in optimal set after 1st show. Since S_{j1} finished after S_{i1}, we can still continue from one of the shows from S_{i1+} after watching the first show S_{j1}. Hence choosing the S_{j1} by greedy method has not worsen the solution and is still equal to the optimal solution

2. Now for the second step, according to greedy approach, we will choose that show which is possible at this time(i.e. the finish time of first show) and which finishes the latest. Suppose second show according to optimal solution is S_{i2} and according to greedy is S_{j2}. Lets replace S_{i2} with S_{j2}. Since we know that S_{i1} finished late as compared to S_{j1}. S_{j2} must have started

late or equal to the time of S_{i2}. Since we always choose the show which finished last from all possible shows at that time, therefore S_{j2} must finish after or at same time with S_{i2}. Therefore we still can pick up any of the shows after j2 from S_{i2+}. Hence this swap has also not worsen the optimal solution. It still is equal to the optimal solution

Similarly choosing k shows according to greedy and finally arriving at greedy will give the solution that will be as good as the optimal solution. Hence This greedy works correctly and gives the optimal solution.

Since we only needed to sort the array once and traverse it from 1 to n
Total time complexity of the solution will be
O($n \log n$) + O(n) = O($n \log n$)

□

(b) (4 points). Assume now that Little Johnny will only watch shows from the beginning till end (except show starting before S or ending after F), but now he fetches another TV from the adjacent room, so that he can potentially watch up to two shows at a time. Can you find a strategy that will give the smallest set of TV shows that he can watch on the two TVs, so that at any time throughout the interval $[S, F)$ he watches at least one (and at most two) shows. (**Hint**: Try to examine your algorithm in part (a).)

Solution: The same greedy approach as from previous part will still work in this question. After calculating
the final set from previous part, we can watch alternate shows on different tv's. Suppose
Final := $\{(s_1, f_1), (s_2, f_2), \ldots, (s_k, f_k)\}$
Observe that Final will be sorted according to the start times
Where s_i, f_i are start and finish times of show i.
Finally we can split Final into 2 parts,
Final1 (Shows on tv1) = $\{(s_1, f_1), (s_3, f_3), \ldots, (s_{2*j-1}, f_{2*j-1})\}$
Final2 (Shows on tv2) = $\{(s_2, f_2), (s_4, f_4), \ldots, (s_{2*j}, f_{2*j})\}$

□