Name: Inchara Raveendra

SCU ID: 00001653600

Assignment - 6

16.1-1

Give a dynamic-programming algorithm for the activity-selection problem, based on recurrence (16.2). Have your algorithm compute the sizes $c[i, j]$ as defined above and also produce the maximum-size subset of mutually compatible activities.

Assume that the inputs have been sorted as in equation (16.1). Compare the running time of your solution to the running time of GREEDY-ACTIVITY-SELECTOR.

```
DYNAMIC-ACTIVITY-SELECTOR (s, f, n)
    let c[0....n+1, 0....n+1] and act[0....n+1, 0....n+1] be
    new tables
    for i = 0 to n
        c[i, i] = 0
        c[i, i+1] = 0
    c[n+1, n+1] = 0
    for l = 2 to n+1
      for i = 0 to n - l + 1
          j = i + 1
          c[i, j] = 0
          k = j - 1
          while f[i] < f[k]
             if f[i] ≤ s[k] and f[k] ≤ s[j] and
             c[i, k] + c[k, j] + 1 > c[i, j]
```

```
            c[i,j] = c[i,k] + c[k,j] + 1
            act[i,j] = k
        k = k - 1
print "Maximum size set of mutually compatible
activities has size" + c[0, n+1]
print "The set contains"
PRINT-ACTIVITIES (c, act, 0, n+1)

PRINT-ACTIVITIES (c, act, i, j)
    if c[i, j] > 0
        k = act[i, j]
        print k
        PRINT-ACTIVITIES (c, act, i, k)
        PRINT-ACTIVITIES (c, act, k, j)
```

GREEDY-ACTIVITY-SELECTOR runs in $\Theta(n)$ time and DYNAMIC-ACTIVITY-SELECTOR runs in $O(n^3)$ time

16.1-2

Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start that is compatible with all previously selected activities. Describe how this approach is a greedy algorithm, and prove that it yields an optimal solution.

The above approach makes locally optimal choices

at each step that maximizes the number of activities selected at each step.
To prove it results in an optimal solution, we can use mathematical induction.

Assume all activities are sorted in increasing order of their finish times. $A_1, A_2, \ldots, A_n$ is a set of 'n' activities where each activity A_i has a start time s_i and finish time f_i

Base case: $n = 1$
Only one activity A_1 is available and it has to be selected, it being the last activity to start and mutually compatible with itself.

For $n = K$, where $K > 1$ assume the approach yields an optimal solution
Consider $K+1$ activities, $A_1, A_2, \ldots, A_K, A_{K+1}$
From assumption, we know selecting last activity to finish gives us the optimal solution and is compatible with first 'K' activities

There are two outcomes either A_{k+1} is compatible with A_k or not. If compatible with previous 'k' activities, we can select A_{k+1} and get an optimal solution for k+1 activities. If not, we can ignore A_k and apply the same approach for first 'k' activities to find an optimal solution by induction.

Thus, we have shown that the above approach of selecting last activity to finish first yields an optimal solution.

16.1-3

Not just any greedy approach to the activity-selection problem produces a maximum-size set of mutually compatible activities. Give an example to show that the approach of selecting the activity of least duration from among those that are compatible with previously selected activities does not work. Do the same for the approaches of always selecting the compatible activity that overlaps the fewest other remaining activities and always selecting the compatible remaining activity with the earliest start time.

a) Selecting least duration

A_i	1	2	3	4	5
s	1	3	4	6	7
f	4	5	7	8	9

We select A_2 first then A_4 since they have the least duration and mutually compatible $\{2, 4\}$

But this is not optimal since the optimal solution is $\{1, 3, 5\}$

b) Fewest overlaps

A_i	1	2	3	4	5	6	7	8	9	10	11
s	0	1	1	1	2	3	4	5	5	5	6
t	2	3	3	3	4	5	6	7	7	7	8
# of overlaps	3	4	4	4	4	2	4	4	4	4	3

We select A_6 first and only 2 other activities

This is not optimal since the optimal solution is $\{A_1, A_5, A_7, A_{11}\}$

c) Earliest start time

A_i	1	2	3
s	1	2	4
t	10	3	5

We select A_1 first $\{A_1\}$

and none others

This is not optimal since $\{A_2, A_3\}$ is the optimal solution

From all the above cases, we can say that just any greedy approach of activity selection does not give an optimal solution

16.2-1
Prove that the fractional knapsack problem has the greedy-choice property.

Let I be the instance of knapsack problem.
$n \rightarrow$ number of items
$v_i \rightarrow$ value of item i
$w_i \rightarrow$ weight of item i
$W \rightarrow$ capacity of knapsack

Items are arranged in increasing order of v_i/w_i and that $W \geq w_n$
Let $S = (s_1, s_2, s_3, \ldots\ldots, s_n)$ be a solution. the greedy algorithm assigns $s_n = \min(w_n, W)$ and solves the subproblem,

$I' = (n-1, \{v_1, v_2, \ldots\ldots, v_{n-1}\}, \{w_1, w_2, \ldots\ldots, w_{n-1}\}, W - w_n)$
until $W = 0$ or $n = 0$

We can prove this strategy gives an optimal solution

using contradiction.
Suppose optimal solution to I is $s_1, s_2, \ldots, s_n$ where $s_n < \min(w_n, W)$. Let i be the smallest number such that $s_i > 0$
By decreasing s_i to $\max(0, W - w_n)$ and increasing s_n by same amount, we get a better solution.
So, the assumption must be false and the problem has a greedy-choice property.

16.2-2
Give a dynamic-programming solution to the 0-1 knapsack problem that runs in $O(n\,W)$ time, where n is the number of items and W is the maximum weight of items that the thief can put in his knapsack.

The 0-1 knapsack problem involves selecting a subset of items to maximize the value while total weight of selected items does not exceed a given capacity. Either the item is taken completely or left behind.

Using dynamic programming, we construct a table of size $(n+1) \times (W+1)$
Each entry $T[i][j]$ represents the max value that can be obtained using first 'i' items and max

weight limit of 'j'

Recurrence relation :

$T[i][j] = \max(T[i-1][j], T[i-1][j-w[i]] + v[i])$

Base case : $T[0][j] = 0$

We cannot select any item(s) if no items are available

Maximum value can be obtained using all 'n' items and a knapsack with max weight 'W' given by $T[n][W]$. Therefore, the solution runs in $O(nW)$ time since we have to fill $n \times W$ entries of the table and each entry requires constant time to compute.

$\therefore$ Overall time complexity is $O(nW)$

16-1 Coin changing

Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.

a. Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.

This can be achieved by using higher value coins first and repeating the process for remaining

amount. Such a greedy approach provides optimal solution since we start by choosing local optimal solution leading to global optimal solution.

a) Pennies and Nickels

At most, 4 pennies can be used. Anything larger would be replaced by 1 nickel. This reduces total coins by 4. When remaining money is greater than 5, we can start with using as many nickels and then go to pennies.

b) Pennies, nickels and dimes

At most, 1 nickel because 2 nickels = 1 dime This reduces total coins by 1. If remaining money is greater than 10, we start with using dimes then nickel and pennies the last.

c) Pennies, nickels, dimes and quarters

At most 2 dimes because 3 dimes = 1 quarter + 1 nickel. This reduces total coins by 1. If remaining money is greater than 30, we could start with using combination of quarters and nickels.

Then all nickels can be replaced by dimes.

d) For the difference between 25 and 30, we could use 2 dimes, 1 nickel and n-25 pennies. A quarter replaces the 2 dimes and a nickel. This reduces the total coins by 2

Therefore, greedy algorithm provides optimal solution for coin change problem.

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

Also, find all the optimal solutions to the activity selection problem on page 415.

First select A_1 $\quad S_1 = \{A_2, A_3, A_4, A_5, A_6, A_7, A_8, A_9, A_{10}, A_{11}\}$

Next select A_4 $\quad S_4 = \{A_5, A_6, A_7, A_8, A_9, A_{10}, A_{11}\}$
$\{A_1, A_4\}$

Then select A_8 $\quad S_8 = \{A_9, A_{10}, A_{11}\}$
$\{A_1, A_4, A_8\}$

Select A_{11} $\quad S_{11} = \phi$

$\therefore$ Optimal solution $= \{A_1, A_4, A_8, A_{11}\}$ $\rightarrow$ (1)

If we select A_2 first $\quad S_2 = \{A_4, A_6, A_7, A_8, A_9, A_{10}, A_{11}\}$

Next we select A_4 $\quad S_4 = \{A_8, A_9, A_{10}, A_{11}\}$

Then we select A_8 $\quad S_8 = \{A_{11}\}$

Select A_{11} $\quad S_{11} = \phi$

$\therefore$ optimal solution $= \{A_2, A_4, A_8, A_{11}\} \rightarrow (2)$

Similarly, $\{A_2, A_4, A_9, A_{11}\} \rightarrow (3)$

(1), (2) and (3) are the optimal solutions to the given activity selection problem