

Homework Assignment: 3

Name: Jonathan Gaines

Date: 4/19/2017

1. Job Optimization

Solution	Time Slot 1	Time Slot 2	profit
1	Job 1	Job 3	55
2	Job 3	Job 1	55
3	Job 2	Job 1	65
4	Job 2	Job 3	60
(a) 5	Job 4	Job 1	70
6	Job 4	Job 3	65
7	Job 1	N/A	30
8	Job 2	N/A	35
9	Job 3	N/A	25
10	Job 4	N/A	40

- (b) The optimal schedule has Job 4 in timeslot 1 and Job 1 in timeslot 2 for a profit of \$70.
- (c) A high level greedy algorithm would choose the largest profit with a deadline of 1 or 2, then choose the largest profit with a deadline of 1. In this case, it would choose Job 4, then Job 1. The time efficiency to calculate is $O(n^2)$ and the space efficiency is $O(1)$.

2. Dynamic Programming: Change Making

- (a) The minimum number of coins needed to meet the amount is 3.
- (b) Minimum coin combinations include $\{1, 2, 5\}$ and $\{3, 3, 3\}$

n	0	1	2	3	4	5	6	7	8	9
$f(n)$	0	1	2	1	2	1	2	3	2	3

- (d) Change-making($D[j]$, n):

```

 $f[0] = 0$ 
for i = 1 to n do
    temp =  $\infty$ 
    j = 1
    while j  $\leq m$  and i  $\geq D[j]$  do
        temp = min( $f(i-D[j])$ , temp)
        j = j + 1
     $f[1] = temp + 1$ 
return  $f(n)$ 

```

3. Dynamic Programming: Knapsack Problem

	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	10	10	10	10	10	10
2	0	0	0	0	40	40	40	40	50	40	
3	0	0	0	0	40	40	40	40	50	70	
4	0	0	0	50	50	50	50	90	90	90	90

(b) The optimal subset has a value of \$90 and consists of items 2 and 4.

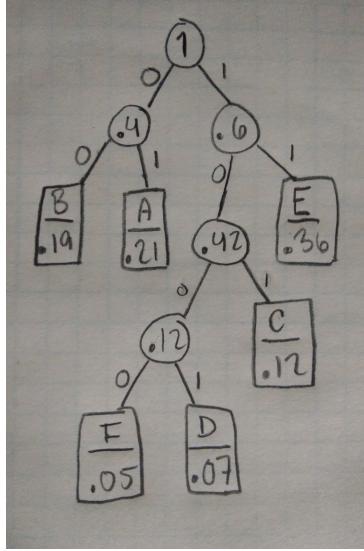
(c) Knapsack(vals[], weights[], w):

```

n = len(vals[])
for j from 0 to w:
    m[0, j] = 0
for i from 1 to n:
    for j from 0 to w:
        if weights[i] < j:
            m[i, j] = m[i-1, j]
        else:
            m[i, j] = max(m[i-1, j], m[i-1, j - weights[i]] + vals[i])
return m[n, w]
```

4. Greedy Algorithm

(a)



(b) 10000110111

(c) BADFAD