Engsci 760 Assignment 4 Report

Aiden Burgess - abur970 - 600280511

1. Coin Counting

a.

Stages: Amount of change left

States: Number of coins used

Actions: Add a coin of a remaining denomination

Costs:

1 if we add the coin

0 if we don't add the coin

b.

$$V_N(x) = \begin{cases} 1 & \text{if } x = D_N \\ \min_{d \in D}(V_{n-1}(x-d)) & \text{if possible to make change with coin} \\ \infty & \text{otherwise} \end{cases}$$

if $\frac{x}{D_N}$ is not an integer, then this means that our change cannot be split evenly into a certain denomination. This means that we have to use other coins to make up the change, or it may not be possible to use this coin, in which case we have an infinite cost.

c.

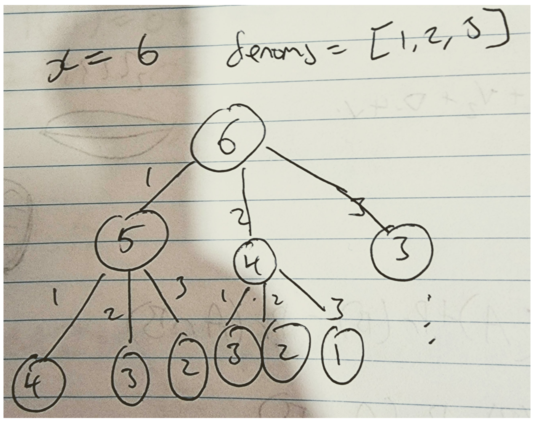
$$V_n(x) = 1 + \min_{d \in D}(V_{n-1}(x-d))$$

$$V_N(0) = 0$$

d.

Optimal substructure: We combine the solutions found from lower amounts of change to produce the solution of a higher change amount.

Overlapping subproblems: Solutions to subproblems are reused repeatedly, as combinations of different coins may lead to the same change remaining, which is the same subproblem. The image below is a simple illustration of such reuse.



e.

i.

$$V(0) = \min_{d \in D}(V(d+1))$$

We stop the recursion when the target value x has been reached

ii.

Natural ordering: start from 0 change, then iteratively calculate until we reach x change. This is a bottom up solution as opposed to recursive solution.

f.

Code is also submitted via canvas.

```
def optimalCoinChange(x, denoms):
    # Function finds the minimum number of coins required to change a
monetary
    # amount.
    # Inputs:
    # x = amount of money to be given in coins, given as an INTEGER, in
cents.
    # e.g. $1.35 is input as 135
    # denoms = denominations of coins available, in INTEGER cents,
    # given as a ROW VECTOR.
    # Output:
    # numCoins = optimal number of coins used to find x
    # Aiden Burgess - abur970

    # Initialise initial coins for change to inf, except 0, which is the
base case
    minCoinsNeeded = [float('inf') for i in range(x+1)]
    minCoinsNeeded[0] = 0

    # For each change value, iteratively calculate the minimum change
required.
    for i in range(x):
        for coin in denoms:
            total = i+coin
            if total <= x:
                minCoinsNeeded[total] = min(
                    minCoinsNeeded[total], minCoinsNeeded[i]+1)

    return minCoinsNeeded[x]
```

2. Exam Beverages

a)

Stages: Number of exams completed

States: Amount of beers available to drink

Actions: Drink 0, 1, or 2 beers

b)

The value function for this problem represents the amount of satisfaction that the student can gain from having x available beers after exam n.

c)

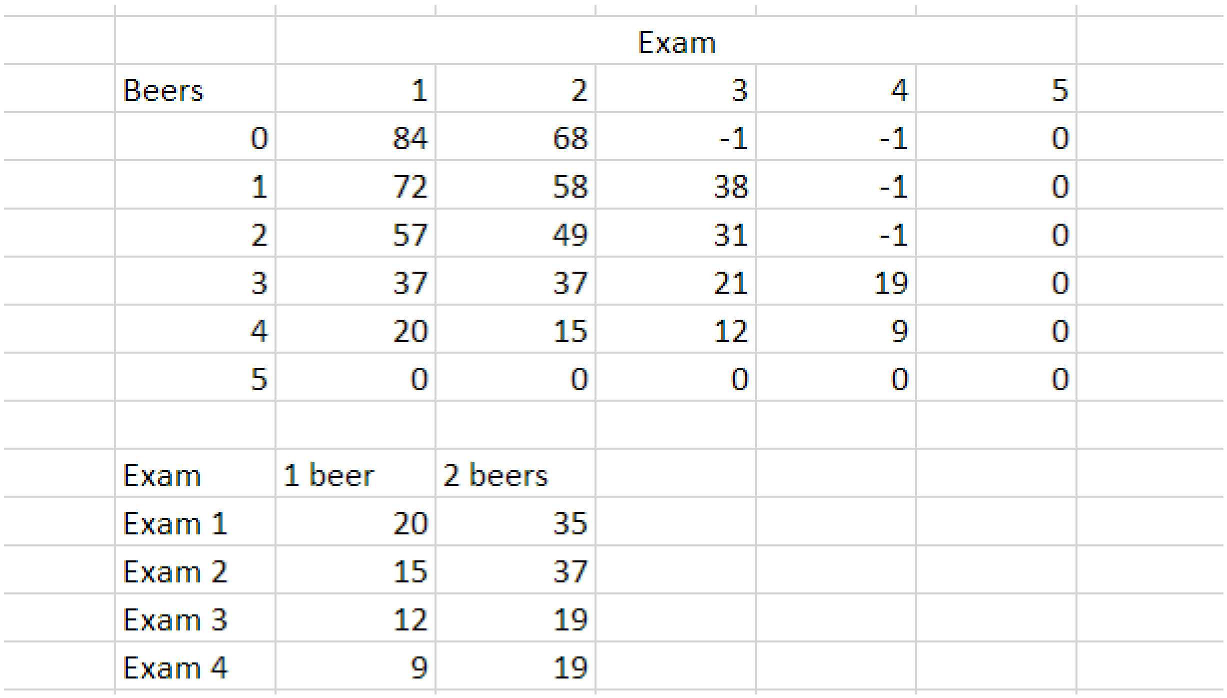
$$V_n(x) = \max(V_{n+1}(x), V_{n+1}(x-1), V_{n+1}(x-2))$$

Note for the above recursion we do not let x go below 0.

This recursion ends when x = 0

d)

The maximum amount of satisfaction that the student can gain is 84.



			Exam		
Beers	1	2	3	4	5
0	84	68	-1	-1	0
1	72	58	38	-1	0
2	57	49	31	-1	0
3	37	37	21	19	0
4	20	15	12	9	0
5	0	0	0	0	0

Exam	1 beer	2 beers
Exam 1	20	35
Exam 2	15	37
Exam 3	12	19
Exam 4	9	19

e)

The optimal policy for consuming the five beverages is: 2 after exam 1, 2 after exam 2, and 1 after exam 3

35+37+12 = 84

3.

a)

$$V_N = \int_0^3 r.\,\frac{6-2r}{9}dr$$

$$V_N = \left[\frac{3r^2}{9} - \frac{2r^3}{27}\right]_0^3$$

$$V_N = 3 - 2 = 1$$

b)

If the current candidates reward R is greater than the reward we expect from rejecting the candidate and continuing the interview process, then we accept the candidate. Otherwise we reject the candidate. i.e. $V_{N-1} = \max(R, V_N)$

c)

The expected value for V_{N-1} is dependent on what we expect from V_N. If our candidate achieves a higher reward than what we expect if we reject them (V_N) then we accept them. This represents the lower bound of the first integral. Otherwise, we have a probability determined from 1-0 f(r) to receive V_N.

$$V_{N-1} = \int_{V_N}^{3} r.\frac{6-2r}{9}dr + \int_{0}^{V_N} V_N.\frac{6-2r}{9}dr$$

$$V_{N-1} = \int_{1}^{3} r.\frac{6-2r}{9}dr + \int_{0}^{1} 1.\frac{6-2r}{9}dr$$

$$V_{N-1} = \left[\frac{3r^2}{9} - \frac{2r^3}{27}\right]_1^3 + \left[\frac{6r}{9} - \frac{r^2}{9}\right]_0^1$$

$$V_{N-1} = \frac{20}{27} + \frac{5}{9} = \frac{35}{27}$$

Hence, the equation is proven.

d)

We extend the specific behaviour from V_{N-1} to be more general for all n

$$V_n = \int_{V_{n+1}}^{3} r.\frac{6-2r}{9}dr + \int_{0}^{V_{n+1}} V_{n+1}.\frac{6-2r}{9}dr$$

$$V_n = \int_{V_{n+1}}^{3} r.\frac{6-2r}{9}dr + \int_{0}^{V_{n+1}} V_{n+1}.\frac{6-2r}{9}dr$$

$$V_n = \left[\frac{3r^2}{9} - \frac{2r^3}{27}\right]_{V_{n+1}}^3 + \left[\frac{6r}{9} - \frac{r^2}{9}\right]_0^{V_{n+1}}$$

$$V_n = \left[\frac{3r^2}{9} - \frac{2r^3}{27}\right]_{V_{n+1}}^3 + \left[\frac{6.V_{n+1}.r}{9} - \frac{V_{n+1}.r^2}{9}\right]_0^{V_{n+1}}$$

$$V_n = \left[\frac{3r^2}{9} - \frac{2r^3}{27}\right]_{V_{n+1}}^3 + \left[\frac{6.V_{n+1}.r}{9} - \frac{V_{n+1}.r^2}{9}\right]_0^{V_{n+1}}$$

$$V_n = \left(1 - \frac{V_{n+1}^2}{3} + \frac{2V_{n+1}^3}{27}\right) + \left(\frac{2V_{n+1}^2}{3} - \frac{V_{n+1}^3}{9}\right)$$

$$V_n = 1 + \frac{V_{n+1}^2}{3} - \frac{V_{n+1}^3}{27}$$

Hence the equation is proven.