CS 566 Homework 5 Jixing (Jacey) Man

Question 1:
Part 1: Sub-problems are possible combinations of all items with their values and weights.
In this case there are 4 items and 4 values and 4 weights.
So the table should be:

items	1	2	3	4
values	3	2	4	3
weights	2	3	3	2

There total 3 subset with 4 item so 4*4*4= 64, there are total 64 sub-problems.

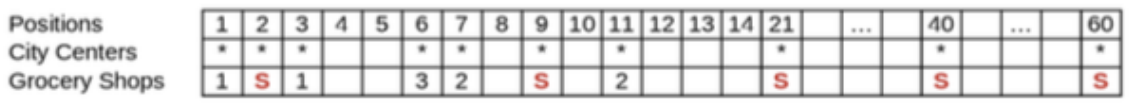
Part 2: so the input in this question is which combination of investment with House value and profit can be the largest within the 10 million budget, in this case, the house values in millions would be the value, the expected profit in the next year would be the weight.

Part 3: Please see below:

V[I,w]	0	1	2	3	4	5	6	7	8	9	10
i = 0	0	0	0	0	0	0	0	0	0	0	0
I = 1	0	0	3	3	3	3	3	3	3	3	3
I = 2	0	0	0	2	2	2	2	2	2	2	2
I = 3	0	0	0	4	4	4	4	4	4	4	4
I = 4	0	0	3	3	6	7	7	10	10	10	12

Part 4: The final output is 12 million, with V(item 4, w 10), the optimal solution is item 3 Watertown 4 million home with profit expectation 3%

Question 2:
Part 1: so in this case, according to the question description and table:



Positions	1	2	3	4	5	6	7	8	9	10	11	12	13	14	21		...		40		...		60
City Centers	*	*	*			*	*		*		*				*				*				*
Grocery Shops	1	S	1			3	2		S		2				S				S				S

The item number would be the 60 position, the city center would be value, and the grocery shop would be the weights, so in this case 60*1000*29 = 174000, the number 29 is because integer between 2-30 is 29. The count of sub-problem is 174000.

Part 2:

```
def groceryShops(A, k):
    # Write your code here——
```

```
A = [1:1000]
K =  [2:30]
    A = [[0 for w in range(A + 1)]
            for i in range(n + 1)]
    K = [[0 for w in range(K + 1)]
            for i in range(n + 1)]
    if not A:
        return 0
    n = len(A) # n is the number of city centers.

    if k >= n: # we should not have more shops than city centers.
        return 0
    A.sort()
    cost = [[0 for _ in range(n + 1)] for _ in range(n + 1)]
    """
    cost[i][j] is the minimum cost to build a shop between city
centers i and j
    This shop should be in the median of the city centers.
    """
    for i in range(n):
        for j in range(i, n):
            mid = int(i + (j - i) / 2) # This can be better than this.

            for r in range(i, mid + 1):
                cost[i + 1][j + 1] += A[mid] - A[r]
            for r in range(mid + 1, j + 1):
                cost[i + 1][j + 1] += A[r] - A[mid]
    """
    dp[i][j] is the minimum cost for the first j city centers with i
shops.
    """
    dp = [[float('inf') for _ in range(n + 1)] for _ in range(k + 1)]
    dp[0][0] = 0
    for i in range(1, k + 1):
        for j in range(1, n + 1):
            for r in range(j):
                dp[] [] = ...
return
```

Part 3: The run time complexity of the algorithm is Big O(n^3), with number of distinct sub-problems time complexity of Theta(n^2)