Problem 1 A:

A greedy algorithm is one that attempts to make decisions by selecting the best option at that particular stage of processing. For making change using the fewest number of coins, this algorithm would work by comparing the value of the amount that needs to be broken down and the highest coin value that can fit within that value. If that coin value fits, the remaining value of the amount that needs to be broken down is reduced by that coin value and the process is repeated until the value of the amount that needs to be broken down is 0.

For example:

Make change for $1.33 using the US dollar coin set consisting of 25¢ quarters, 10¢ dimes, 5¢ nickels, and 1¢ pennies.

$1.33. Highest value that can fit is a 25¢ quarter.

$1.33 - 25¢ = $1.08

$1.08. Highest value that can fit is a 25¢ quarter.

$1.08 - 25¢ = 83¢

83¢. Highest value that can fit is a 25¢ quarter.

83¢ - 25¢ = 58¢

58¢. Highest value that can fit is a 25¢ quarter.

58¢ - 25¢ = 33¢

33¢. Highest value that can fit is a 25¢ quarter.

33¢ - 25¢ = 8¢

8¢. Highest value that can fit is a 5¢ nickel.

8¢ - 5¢ = 3¢

3¢. Highest value that can fit is a 1¢ penny.

3¢ - 1¢ = 2¢

2¢. Highest value that can fit is a 1¢ penny.

2¢ - 1¢ = 1¢

1¢. Highest value that can fit is a 1¢ penny.

1¢ - 1¢ = 0¢

No more change to make. 9 coins used.

Problem 1 B:

One set of coin denominations which will not produce optimal solutions using the greedy algorithm is an

11¢ coin, a 9¢ coin, and a 1¢ coin.

For example:

Make change for 30¢ using the imaginary coin set consisting of 11¢, 9¢, and 1¢ coins.

30¢. Highest value that can fit is an 11¢ coin.

30¢ - 11¢ = 19¢

19¢. Highest value that can fit is an 11¢ coin.

19¢ - 11¢ = 8¢

8¢. Highest value that can fit is a 1¢ coin.

8¢ - 1¢ = 7¢

7¢. Highest value that can fit is a 1¢ coin.

7¢ - 1¢ = 6¢

6¢. Highest value that can fit is a 1¢ coin.

6¢ - 1¢ = 5¢

5¢. Highest value that can fit is a 1¢ coin.

5¢ - 1¢ = 4¢

4¢. Highest value that can fit is a 1¢ coin.

4¢ - 1¢ = 3¢

3¢. Highest value that can fit is a 1¢ coin.

3¢ - 1¢ = 2¢

2¢. Highest value that can fit is a 1¢ coin.

2¢ - 1¢ = 1¢

1¢. Highest value that can fit is a 1¢.

1¢ - 1¢ = 0¢

No more change to make. 10 coins used.

This is not an optimal solution. A better solution can be produced using the 9¢ coin.

For example:

Make change for 30¢ using the imaginary coin set consisting of 11¢, 9¢, and 1¢ coins.

30¢. Use the 9¢ coin.

30¢ - 9¢ = 21¢

21¢. Use the 9¢ coin.

21¢ - 9¢ = 12¢

12¢. Use the 11¢ coin.

12¢ - 11¢ = 1¢

1¢. Use the 1¢ coin.

1¢ - 1¢ = 0¢

No more change to make. 4 coins used.

Therefore, some coin denominations will prevent a greedy algorithm from correctly making change with the fewest number of coins.

Problem 2:

The following demonstrates why the use of a greedy algorithm will not always determine an optimal way to cut rods. The demonstration will utilize tables representing rod segments, prices, and densities.

Starting rod:

Inch 1	Inch 2	Inch 3	Inch 4
Price: $2	Price: $6	Price: $10	Price: $13
Price per inch: 2	Price per inch: 3	Price per inch: 3.3	Price per inch: 3.25

Selecting rod length of highest density 3.3 results in Total Sale of $10.

Remaining rod:

Inch 1
Price: $2
Price per inch: 2

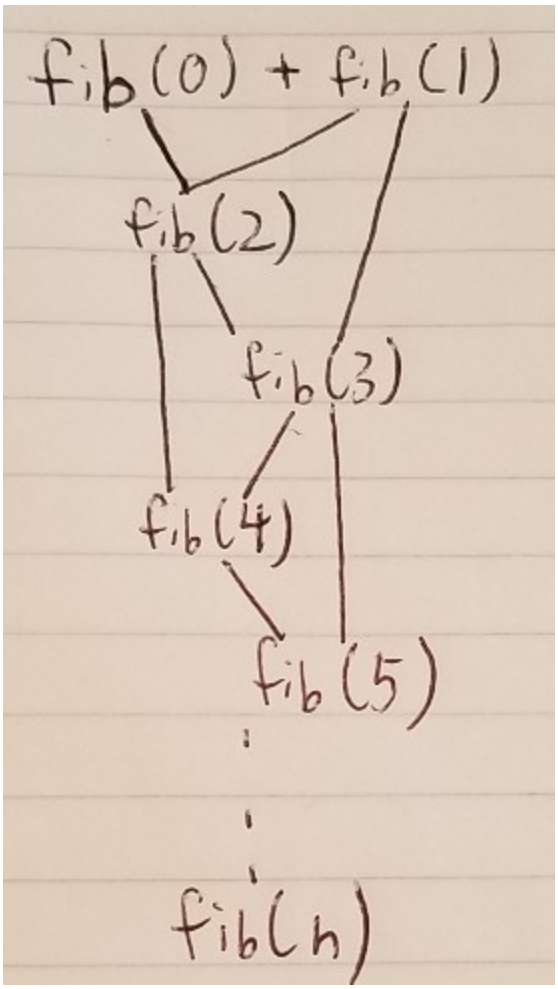
Selecting rod length of highest density 2 results in Total Sale of $12.

However, it is obvious that selling the entire rod for $13 is a more optimal solution than what the greedy algorithm offers. Thus, the greedy algorithm does not always determine an optimal way to cut rods.

Problem 3:

The dynamic Fibonacci algorithm works by using a bottom-up approach to constructing the nth Fibonacci number. It starts by producing the third Fibonacci number by adding the first two, then adding the second Fibonacci number to the third to produce the fourth, then adding the third Fibonacci number to the fourth to produce the fifth, and so on until the nth Fibonacci number is produced. The crux of the algorithm is the storing of the previous solution to produce the next Fibonacci number without having to do additional calculations. These previous solutions can easily be stored in an array data structure.

Sub-problem Graph:



The number of vertices will be n since you will not get duplicates of calculations for Fibonacci numbers. The graph would be more correct if it represented the first Fibonacci number as fib(1), but the main point remains the same. The number of edges will be 2n - 4 since previous solutions to Fibonacci calculations are used twice, once to produce the next Fibonacci number, and twice to produce the Fibonacci number after the next Fibonacci number.

Problem 4:

A dynamic-programming algorithm that can solve the 0-1 Knapsack Problem will need to rely on solutions to subproblems defined by the total benefit produced as a result of the value of the individual item to be placed in the bag and the weight of the individual item to be placed in the bag. The maximum possible benefit that can be carried can then be constructed by solutions to these previous

subproblems. The subproblems are computed by comparing the total available weight with the weight of the currently examined item. If the currently examined item is heavier than the total available weight, then the solution to the current subproblem is the solution to the previous subproblem, which means that the total benefit is unchanged and the remaining weight is unchanged. Otherwise, if the currently examined item is lighter or as heavy as the total available weight of the knapsack, then the solution to the current subproblem is either the solution to the previous subproblem (meaning that the total benefit is unchanged and the remaining weight is unchanged) or the total benefit plus the benefit of the currently examined item, whichever is higher. In the latter case with the benefit of the currently examined item being added to the total benefit, the weight is modified as well by reducing the remaining weight of the knapsack with the weight of the currently examined item. The storage of benefits and weights for each subproblem occurs in a memo table, whose horizontal indices represent the number of available items and whose vertical indices represent the discrete weights up to the maximum weight of the knapsack. Stored within each memo table cell is the benefit in the knapsack. The resulting maximum benefit in the knapsack will be found in the last cell of the memo table, that is, the cell in the last row and last column in the table.

Below is a trace of the algorithm over items (1, 3), (2, 4), (3, 5), and (4, 8) with the items represented as (w_i, v_i) using a memo table. The maximum knapsack weight is 6.

Initialize weight column cells to 0

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0				
Weight 1	Weight: 1	0				
Weight 2	Weight: 2	0				
Weight 3	Weight: 3	0				
Weight 4	Weight: 4	0				
Weight 5	Weight: 5	0				
Weight 6	Weight: 6	0				

Initialize item row cells to 0

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0				
Weight 2	Weight: 2	0				
Weight 3	Weight: 3	0				
Weight 4	Weight: 4	0				
Weight 5	Weight: 5	0				
Weight 6	Weight: 6	0				

Item 1: Weight is 1, Benefit is 3

Weight 1: Weight is 1

Check if weight of Item 1 is less than or equal to weight of Weight 1.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3			
Weight 2	Weight: 2	0				
Weight 3	Weight: 3	0				
Weight 4	Weight: 4	0				
Weight 5	Weight: 5	0				
Weight 6	Weight: 6	0				

Item 1: Weight is 1, Benefit is 3

Weight 2: Weight is 2

Check if weight of Item 1 is less than or equal to weight of Weight 2.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3			
Weight 2	Weight: 2	0	3			
Weight 3	Weight: 3	0				
Weight 4	Weight: 4	0				
Weight 5	Weight: 5	0				
Weight 6	Weight: 6	0				

Item 1: Weight is 1, Benefit is 3

Weight 3: Weight is 3

Check if weight of Item 1 is less than or equal to weight of Weight 3.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3			
Weight 2	Weight: 2	0	3			
Weight 3	Weight: 3	0	3			
Weight 4	Weight: 4	0				
Weight 5	Weight: 5	0				
Weight 6	Weight: 6	0				

Item 1: Weight is 1, Benefit is 3

Weight 4: Weight is 4

Check if weight of Item 1 is less than or equal to weight of Weight 4.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3			
Weight 2	Weight: 2	0	3			
Weight 3	Weight: 3	0	3			
Weight 4	Weight: 4	0	3			
Weight 5	Weight: 5	0				
Weight 6	Weight: 6	0				

Item 1: Weight is 1, Benefit is 3

Weight 5: Weight is 5

Check if weight of Item 1 is less than or equal to weight of Weight 5.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3			
Weight 2	Weight: 2	0	3			
Weight 3	Weight: 3	0	3			
Weight 4	Weight: 4	0	3			
Weight 5	Weight: 5	0	3			
Weight 6	Weight: 6	0				

Item 1: Weight is 1, Benefit is 3

Weight 6: Weight is 6

Check if weight of Item 1 is less than or equal to weight of Weight 6.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3			
Weight 2	Weight: 2	0	3			
Weight 3	Weight: 3	0	3			
Weight 4	Weight: 4	0	3			
Weight 5	Weight: 5	0	3			
Weight 6	Weight: 6	0	3			

Item 2: Weight is 2, Benefit is 4

Weight 1: Weight is 1

Check if weight of Item 2 is less than or equal to weight of Weight 1.

FALSE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3		
Weight 2	Weight: 2	0	3			
Weight 3	Weight: 3	0	3			
Weight 4	Weight: 4	0	3			
Weight 5	Weight: 5	0	3			
Weight 6	Weight: 6	0	3			

Item 2: Weight is 2, Benefit is 4

Weight 2: Weight is 2

Check if weight of Item 2 is less than or equal to weight of Weight 2.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3		
Weight 2	Weight: 2	0	3	4		
Weight 3	Weight: 3	0	3			

Weight 4	Weight: 4	0	3			
Weight 5	Weight: 5	0	3			
Weight 6	Weight: 6	0	3			

Item 2: Weight is 2, Benefit is 4

Weight 3: Weight is 3

Check if weight of Item 2 is less than or equal to weight of Weight 3.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3		
Weight 2	Weight: 2	0	3	4		
Weight 3	Weight: 3	0	3	7		
Weight 4	Weight: 4	0	3			
Weight 5	Weight: 5	0	3			
Weight 6	Weight: 6	0	3			

Item 2: Weight is 2, Benefit is 4

Weight 4: Weight is 4

Check if weight of Item 2 is less than or equal to weight of Weight 4.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3		
Weight 2	Weight: 2	0	3	4		

Weight 3	Weight: 3	0	3	7		
Weight 4	Weight: 4	0	3	7		
Weight 5	Weight: 5	0	3			
Weight 6	Weight: 6	0	3			

Item 2: Weight is 2, Benefit is 4

Weight 5: Weight is 5

Check if weight of Item 2 is less than or equal to weight of Weight 5.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3		
Weight 2	Weight: 2	0	3	4		
Weight 3	Weight: 3	0	3	7		
Weight 4	Weight: 4	0	3	7		
Weight 5	Weight: 5	0	3	7		
Weight 6	Weight: 6	0	3			

Item 2: Weight is 2, Benefit is 4

Weight 6: Weight is 6

Check if weight of Item 2 is less than or equal to weight of Weight 6.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3		

Weight 2	Weight: 2	0	3	4		
Weight 3	Weight: 3	0	3	7		
Weight 4	Weight: 4	0	3	7		
Weight 5	Weight: 5	0	3	7		
Weight 6	Weight: 6	0	3	7		

Item 3: Weight is 3, Benefit is 5

Weight 1: Weight is 1

Check if weight of Item 3 is less than or equal to weight of Weight 1.

FALSE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	
Weight 2	Weight: 2	0	3	4		
Weight 3	Weight: 3	0	3	7		
Weight 4	Weight: 4	0	3	7		
Weight 5	Weight: 5	0	3	7		
Weight 6	Weight: 6	0	3	7		

Item 3: Weight is 3, Benefit is 5

Weight 2: Weight is 2

Check if weight of Item 3 is less than or equal to weight of Weight 2.

FALSE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	
Weight 2	Weight: 2	0	3	4	4	
Weight 3	Weight: 3	0	3	7		
Weight 4	Weight: 4	0	3	7		
Weight 5	Weight: 5	0	3	7		
Weight 6	Weight: 6	0	3	7		

Item 3: Weight is 3, Benefit is 5

Weight 3: Weight is 3

Check if weight of Item 3 is less than or equal to weight of Weight 2.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

FALSE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	
Weight 2	Weight: 2	0	3	4	4	
Weight 3	Weight: 3	0	3	7	7	
Weight 4	Weight: 4	0	3	7		
Weight 5	Weight: 5	0	3	7		
Weight 6	Weight: 6	0	3	7		

Item 3: Weight is 3, Benefit is 5

Weight 4: Weight is 4

Check if weight of Item 3 is less than or equal to weight of Weight 4.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	
Weight 2	Weight: 2	0	3	4	4	
Weight 3	Weight: 3	0	3	7	7	
Weight 4	Weight: 4	0	3	7	8	
Weight 5	Weight: 5	0	3	7		
Weight 6	Weight: 6	0	3	7		

Item 3: Weight is 3, Benefit is 5

Weight 5: Weight is 5

Check if weight of Item 3 is less than or equal to weight of Weight 5.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	
Weight 2	Weight: 2	0	3	4	4	
Weight 3	Weight: 3	0	3	7	7	
Weight 4	Weight: 4	0	3	7	8	
Weight 5	Weight: 5	0	3	7	9	
Weight 6	Weight: 6	0	3	7		

Item 3: Weight is 3, Benefit is 5

Weight 6: Weight is 6

Check if weight of Item 3 is less than or equal to weight of Weight 6.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	
Weight 2	Weight: 2	0	3	4	4	
Weight 3	Weight: 3	0	3	7	7	
Weight 4	Weight: 4	0	3	7	8	
Weight 5	Weight: 5	0	3	7	9	
Weight 6	Weight: 6	0	3	7	12	

Item 4: Weight is 4, Benefit is 8

Weight 1: Weight is 1

Check if weight of Item 4 is less than or equal to weight of Weight 1.

FALSE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	3
Weight 2	Weight: 2	0	3	4	4	
Weight 3	Weight: 3	0	3	7	7	
Weight 4	Weight: 4	0	3	7	8	
Weight 5	Weight: 5	0	3	7	9	
Weight 6	Weight: 6	0	3	7	12	

Item 4: Weight is 4, Benefit is 8

Weight 2: Weight is 2

Check if weight of Item 4 is less than or equal to weight of Weight 2.

FALSE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	3
Weight 2	Weight: 2	0	3	4	4	4
Weight 3	Weight: 3	0	3	7	7	
Weight 4	Weight: 4	0	3	7	8	
Weight 5	Weight: 5	0	3	7	9	
Weight 6	Weight: 6	0	3	7	12	

Item 4: Weight is 4, Benefit is 8

Weight 3: Weight is 3

Check if weight of Item 4 is less than or equal to weight of Weight 3.

FALSE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	3
Weight 2	Weight: 2	0	3	4	4	4
Weight 3	Weight: 3	0	3	7	7	7
Weight 4	Weight: 4	0	3	7	8	
Weight 5	Weight: 5	0	3	7	9	

Weight 6	Weight: 6	0	3	7	12	

Item 4: Weight is 4, Benefit is 8

Weight 4: Weight is 4

Check if weight of Item 4 is less than or equal to weight of Weight 4.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

FALSE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	3
Weight 2	Weight: 2	0	3	4	4	4
Weight 3	Weight: 3	0	3	7	7	7
Weight 4	Weight: 4	0	3	7	8	8
Weight 5	Weight: 5	0	3	7	9	
Weight 6	Weight: 6	0	3	7	12	

Item 4: Weight is 4, Benefit is 8

Weight 5: Weight is 5

Check if weight of Item 4 is less than or equal to weight of Weight 5.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

TRUE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	3
Weight 2	Weight: 2	0	3	4	4	4
Weight 3	Weight: 3	0	3	7	7	7
Weight 4	Weight: 4	0	3	7	8	8

Weight 5	Weight: 5	0	3	7	9	11
Weight 6	Weight: 6	0	3	7	12	

Item 4: Weight is 4, Benefit is 8

Weight 6: Weight is 6

Check if weight of Item 4 is less than or equal to weight of Weight 6.

TRUE

Check if the benefit from the previous item with new item's weight added plus the new item's benefit is greater than the benefit from the previous item without the new item's weight added.

FALSE

		Item 0	Item 1	Item 2	Item 3	Item 4
Weight 0	Weight: 0	0	0	0	0	0
Weight 1	Weight: 1	0	3	3	3	3
Weight 2	Weight: 2	0	3	4	4	4
Weight 3	Weight: 3	0	3	7	7	7
Weight 4	Weight: 4	0	3	7	8	8
Weight 5	Weight: 5	0	3	7	9	11
Weight 6	Weight: 6	0	3	7	12	12

12 is the best benefit that can be made from the items in the knapsack.