COMP361

Assignment 3

Knapsack Problem

William Kilty

300473541

2:

```
calculate(Item[] items, int limit) {
        values[items.length+1][limit+1] <- 0 //initialise everything to zero
        for (item = 1; < values.length; ++) {
                for (weight = diag; < values[item].length; ++) {
                        <- Barometer here (each time values[x][y] is assigned)
                        values[item][weight] = weight >= items[item-1].weight ?
                        Math.max(values[item-1][weight-items[item-1].weight] +
                        items[item-1].value, values[item-1][weight]) :
                        values[item-1][weight];
                }
        }
        return values[items.length][limit]
}
```

The meat of the algorithm is the part within both for-loops. This works on the following logic:

if the available weight is enough to hold the new item we take the largest of either:

- the best case without this item,
- the best case without the weight of this item with this item's value added

otherwise, we take the best case without this item

In other words, we either take the item, or we don't. We choose the option that gives us the biggest result. And since those two options are the only options, the largest of the two is the optimal answer.

So if we're taking the optimal answer at each step, we can be sure that each step beforehand that we're basing this step off of will be optimal too.

If we have a bag filled this way, the final value at the end of the loop will be a bag with the highest value possible - the optimal solution to the problem.

My test case generator works by creating a list of random length containing items with random weights and values, and then picking a random weight limit. This is then put through the algorithm and the results are analysed to check correctness.

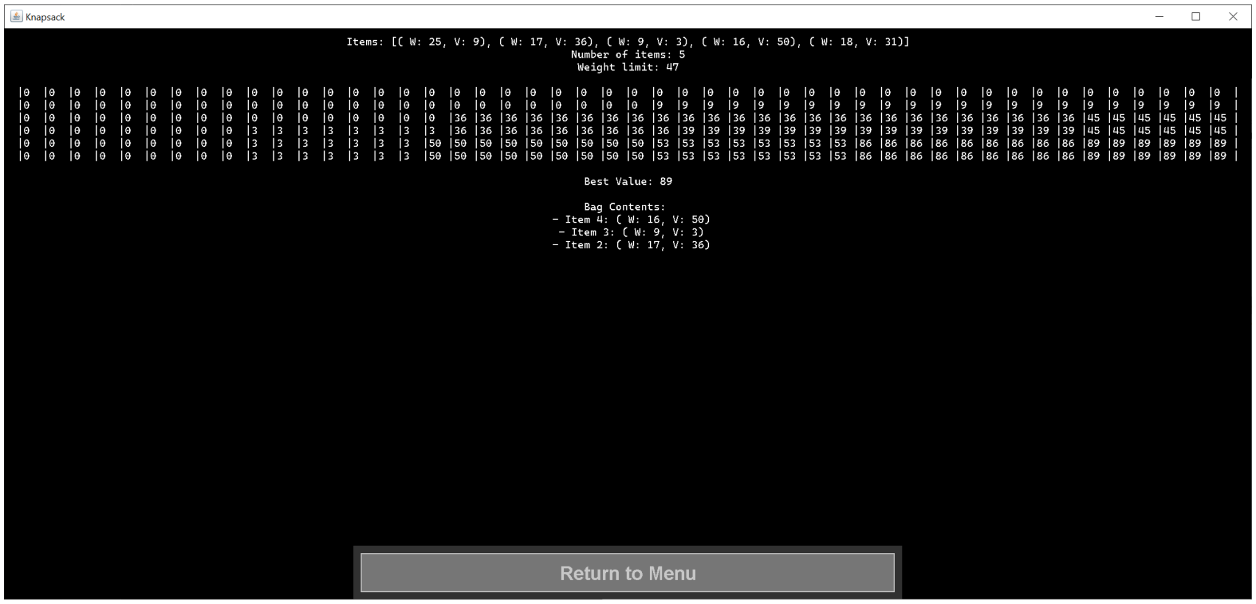
We can then check the barometer results against the inputs to determine complexity.

Our expected complexity should be N*(M+1) where N is the size of the list of items, and M is the weight limit. This translates to Big O complexity of O(m) if we assume that limit>>number of items. (for an interesting problem we want most items to have a weight > 1 and for the bag to be able to hold more than 1 item)

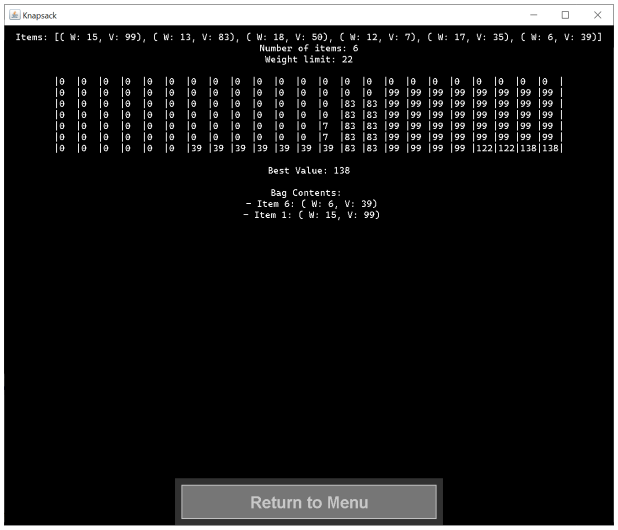
Based on testing;

- Barometer: 371, Num items: 7, Weight Limit: 52
    - 7*(52+1) = 371
- Barometer: 64, Num items: 4, Weight Limit: 15
    - 4*(15+1) = 64
- Barometer: 553, Num items: 7, Weight Limit: 78
    - 7*(78+1) = 553
- Barometer: 305, Num items: 5, Weight Limit: 60
    - 5*(60+1) = 305

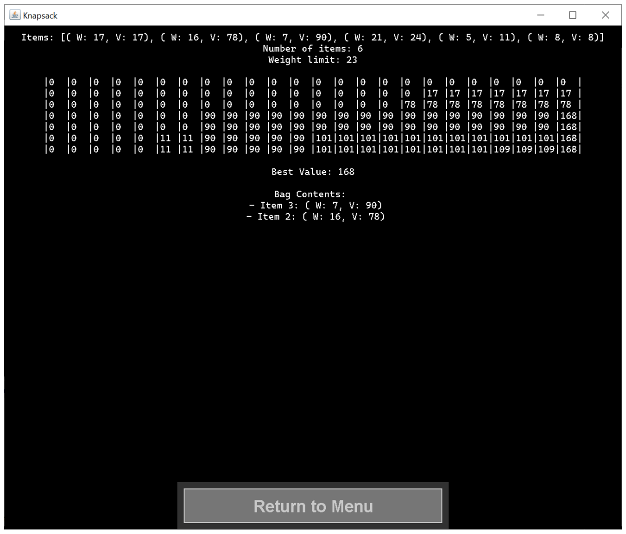
My estimation is spot on. And in each case, limit>>number of items.
Complexity O(m)



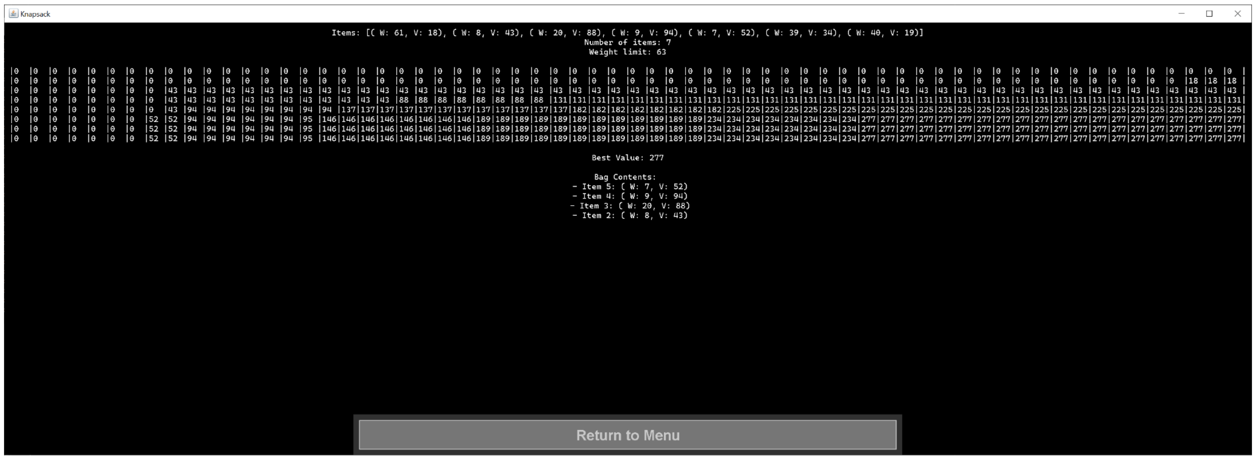
```
                       Items: [( W: 25, V: 9), ( W: 17, V: 36), ( W: 9, V: 3), ( W: 16, V: 50), ( W: 18, V: 31)]
                                                 Number of items: 5
                                                 Weight limit: 47

|0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |
|0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |9  |
|0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |36 |45 |45 |45 |45 |45 |45 |
|0  |0  |0  |0  |0  |0  |0  |0  |0  |3  |3  |3  |3  |3  |3  |3  |3  |36 |36 |36 |36 |36 |36 |36 |36 |36 |39 |39 |39 |39 |39 |39 |39 |39 |39 |39 |39 |39 |39 |39 |39 |39 |45 |45 |45 |45 |45 |45 |
|0  |0  |0  |0  |0  |0  |0  |0  |0  |3  |3  |3  |3  |3  |3  |3  |50 |50 |50 |50 |50 |50 |50 |50 |50 |53 |53 |53 |53 |53 |53 |53 |53 |86 |86 |86 |86 |86 |86 |86 |86 |86 |89 |89 |89 |89 |89 |89 |
|0  |0  |0  |0  |0  |0  |0  |0  |0  |3  |3  |3  |3  |3  |3  |3  |50 |50 |50 |50 |50 |50 |50 |50 |50 |53 |53 |53 |53 |53 |53 |53 |53 |86 |86 |86 |86 |86 |86 |86 |86 |86 |89 |89 |89 |89 |89 |89 |

                                                 Best Value: 89

                                                 Bag Contents:
                                              - Item 4: ( W: 16, V: 50)
                                               - Item 3: ( W: 9, V: 3)
                                              - Item 2: ( W: 17, V: 36)
```



```
Items: [( W: 15, V: 99), ( W: 13, V: 83), ( W: 18, V: 50), ( W: 12, V: 7), ( W: 17, V: 35), ( W: 6, V: 39)]
                              Number of items: 6
                               Weight limit: 22

      |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |
      |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |99 |99 |99 |99 |99 |99 |99 |99 |
      |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |83 |83 |99 |99 |99 |99 |99 |99 |99 |99 |
      |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |83 |83 |99 |99 |99 |99 |99 |99 |99 |99 |
      |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |7  |83 |83 |99 |99 |99 |99 |99 |99 |99 |99 |
      |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |7  |83 |83 |99 |99 |99 |99 |99 |99 |99 |99 |
      |0  |0  |0  |0  |0  |0  |39 |39 |39 |39 |39 |39 |39 |83 |83 |99 |99 |99 |99 |122|122|138|138|

                               Best Value: 138

                                Bag Contents:
                           - Item 6: ( W: 6, V: 39)
                          - Item 1: ( W: 15, V: 99)
```



```
Items: [( W: 17, V: 17), ( W: 16, V: 78), ( W: 7, V: 90), ( W: 21, V: 24), ( W: 5, V: 11), ( W: 8, V: 8)]
                              Number of items: 6
                               Weight limit: 23

    |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |
    |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |17 |17 |17 |17 |17 |17 |17 |
    |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |78 |78 |78 |78 |78 |78 |78 |78 |
    |0  |0  |0  |0  |0  |0  |0  |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |168|
    |0  |0  |0  |0  |0  |0  |0  |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |90 |168|
    |0  |0  |0  |0  |0  |11 |11 |90 |90 |90 |90 |90 |101|101|101|101|101|101|101|101|101|101|101|168|
    |0  |0  |0  |0  |0  |11 |11 |90 |90 |90 |90 |90 |101|101|101|101|101|101|101|101|109|109|109|168|

                               Best Value: 168

                                Bag Contents:
                           - Item 3: ( W: 7, V: 90)
                          - Item 2: ( W: 16, V: 78)
```



```
                                                   Items: [( W: 61, V: 18), ( W: 8, V: 43), ( W: 20, V: 88), ( W: 9, V: 94), ( W: 7, V: 52), ( W: 39, V: 34), ( W: 40, V: 19)]
                                                                               Number of items: 7
                                                                                Weight limit: 63

|0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |
|0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |18 |18 |18 |
|0  |0  |0  |0  |0  |0  |0  |0  |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |
|0  |0  |0  |0  |0  |0  |0  |0  |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |43 |88 |88 |88 |88 |88 |88 |88 |88 |131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|131|
|0  |0  |0  |0  |0  |0  |0  |0  |43 |94 |94 |94 |94 |94 |94 |94 |94 |137|137|137|137|137|137|137|137|137|137|137|137|182|182|182|182|182|182|182|182|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|225|
|0  |0  |0  |0  |0  |0  |0  |52 |52 |94 |94 |94 |94 |94 |94 |95 |146|146|146|146|146|146|146|146|189|189|189|189|189|189|189|189|189|189|189|189|234|234|234|234|234|234|234|234|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|
|0  |0  |0  |0  |0  |0  |0  |52 |52 |94 |94 |94 |94 |94 |94 |95 |146|146|146|146|146|146|146|146|189|189|189|189|189|189|189|189|189|189|189|189|234|234|234|234|234|234|234|234|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|
|0  |0  |0  |0  |0  |0  |0  |52 |52 |94 |94 |94 |94 |94 |94 |95 |146|146|146|146|146|146|146|146|189|189|189|189|189|189|189|189|189|189|189|189|234|234|234|234|234|234|234|234|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|277|

                                                                                Best Value: 277

                                                                                 Bag Contents:
                                                                            - Item 5: ( W: 7, V: 52)
                                                                            - Item 4: ( W: 9, V: 94)
                                                                           - Item 3: ( W: 20, V: 88)
                                                                            - Item 2: ( W: 8, V: 43)
```

3:

```
calculate(Item[] items, int limit) {
      sorted <- items.sort
      weight <- limit
      for (item = 0; < sorted.length; ++) {
            <- Barometer here (each time we try to take an item)
            Item item <- sorted.get(i);
            if (item.weight <= weight) {
                  Item take <- new Item(item.weight, item.value,
                  Math.min(item.number, weight/item.weight));
                  taking.add(take);
                  weight-=take.weight*take.number;
            }
      }
}
```

The meat of the algorithm is the part within the for-loop. This works on the following logic:

if the available weight is enough to hold the new item type we take the smallest of either:

- the number of this item available
- the number of times this item can be taken regardless of how many there actually are

otherwise, we move on

This ensures we always take the most we can of each item, either by taking all there is to take, or by taking as many as we can hold.

The key strategy in this case is utilising a sorted list. This list is sorted by the ratio of values to weights, with the higher the ratio, the better the item is regarded. We assume that taking as many of the best item-type as we can will be better than any other option. We do not check any other options, and so we cannot be sure this is the case.

The greedy algorithm is not optimal. But it will provide the best guess possible given the assumption talked about above.

Once the algorithm has run, we end up with a list of all the items we took, and how many times we took each item. The sum of all the values in this list is the final answer for the problem.

My test case generator works by creating a list of random length containing items with random weights and values and number of instances, and then picking a random weight limit. This is then put through the algorithm and the results are analysed to check correctness.

We can then check the barometer results against the inputs to determine complexity.

Our expected complexity should be N where N is the size of the list of items. This translates to Big O complexity of O(n). However, this is not the full picture… We will go under the assumption of O(n) for now.

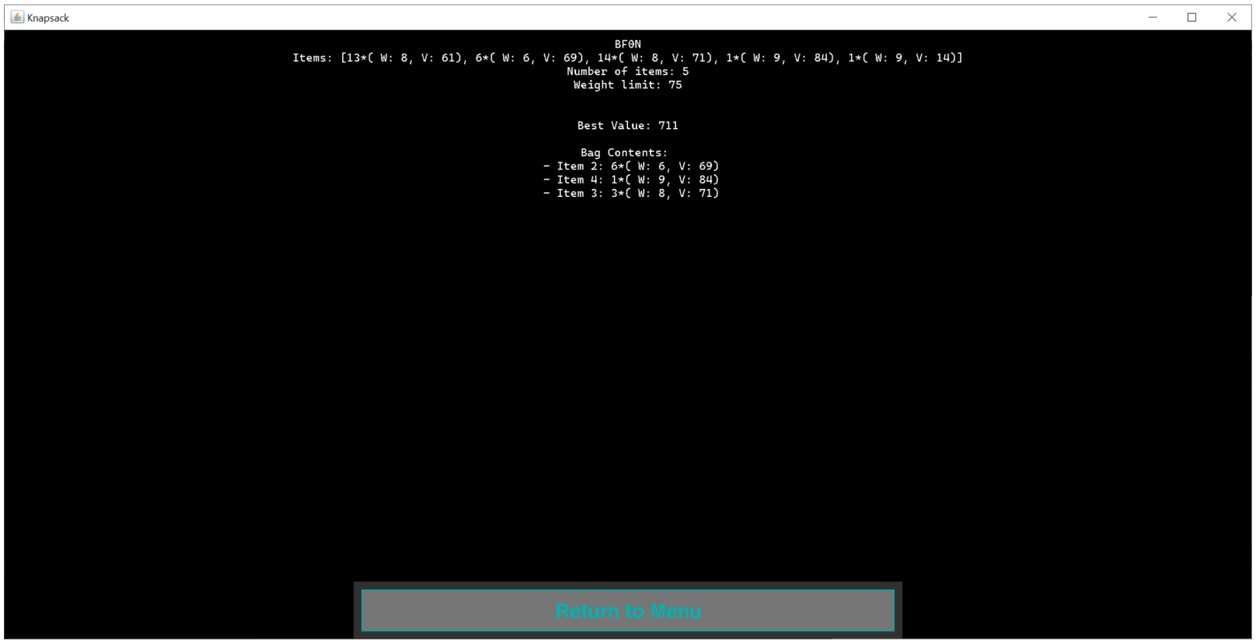
Based on testing;

- Barometer: 7, Num items: 7
- Barometer: 5, Num items: 5
- Barometer: 2, Num items: 2
- Barometer: 4, Num items: 4
- Barometer: 5, Num items: 5

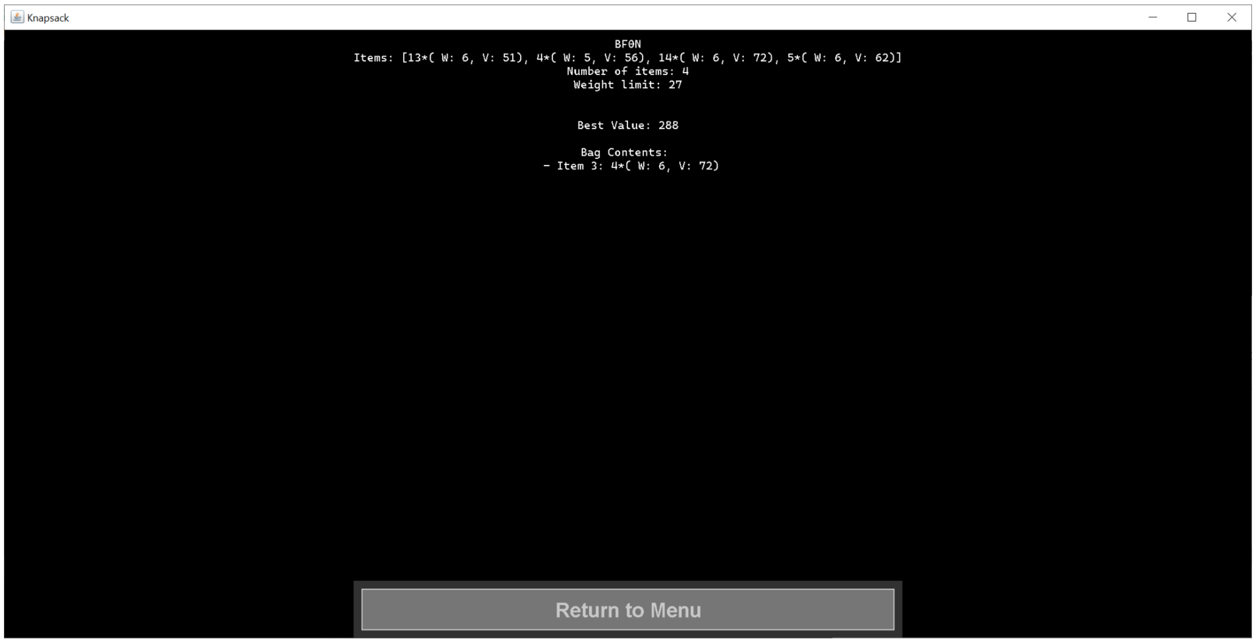
My estimation is spot on. Complexity O(n)...

But this is not true, because there's another feature to the algorithm that has been glossed over. We have to first sort the list of items. The sorting method I used to do this is merge-sort. The complexity of merge sort has been discussed in lectures and previous assignments, so I know it to be O(nlogn).

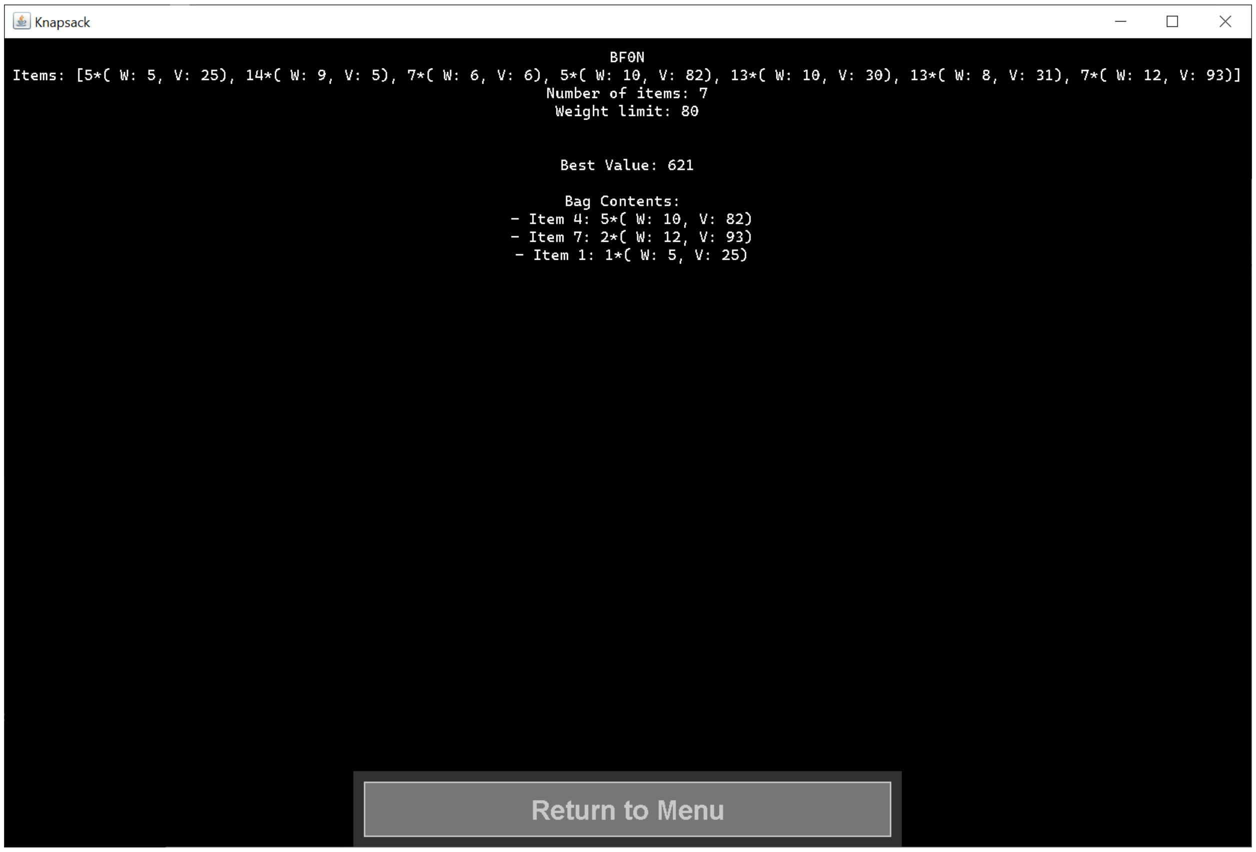
This complexity is greater than O(n), so the complexity of the whole algorithm ends up being O(nlogn).



```
BF0N
Items: [13*( W: 8, V: 61), 6*( W: 6, V: 69), 14*( W: 8, V: 71), 1*( W: 9, V: 84), 1*( W: 9, V: 14)]
Number of items: 5
Weight limit: 75


Best Value: 711

Bag Contents:
- Item 2: 6*( W: 6, V: 69)
- Item 4: 1*( W: 9, V: 84)
- Item 3: 3*( W: 8, V: 71)
```



```
BF0N
Items: [13*( W: 6, V: 51), 4*( W: 5, V: 56), 14*( W: 6, V: 72), 5*( W: 6, V: 62)]
Number of items: 4
Weight limit: 27


Best Value: 288

Bag Contents:
- Item 3: 4*( W: 6, V: 72)
```



```
BF0N
Items: [5*( W: 5, V: 25), 14*( W: 9, V: 5), 7*( W: 6, V: 6), 5*( W: 10, V: 82), 13*( W: 10, V: 30), 13*( W: 8, V: 31), 7*( W: 12, V: 93)]
Number of items: 7
Weight limit: 80


Best Value: 621

Bag Contents:
- Item 4: 5*( W: 10, V: 82)
- Item 7: 2*( W: 12, V: 93)
- Item 1: 1*( W: 5, V: 25)
```

4:

```
calculate(Item[] items, int limit) {
        instances <- (each item : items) item.number // the total collection of items
        values[instances+1][limit+1] <- 0 //initialise everything to zero
        for (instance = 1; < values.length; ++) {
                for (weight = diag; < values[item].length; ++) {
                        <- Barometer here (each time values[x][y] is assigned)
                        values[instance][weight] = weight >= items[item-1].weight ?
                        Math.max(values[instance-1][weight-items[item-1].weight] +
                        items[item-1].value, values[instance-1][weight]) :
                        values[instance-1][weight];
                }
        }
        return values[instances-1][limit]
}
```

The meat of the algorithm is the part within both for-loops. This works on the following logic:

if the available weight is enough to hold the new item we take the largest of either:
- the best case without this item,
- the best case without the weight of this item with this item's value added

otherwise, we take the best case without this item

In other words, we either take the item, or we don't. We choose the option that gives us the biggest result. And since those two options are the only options, the largest of the two is the optimal answer.

So if we're taking the optimal answer at each step, we can be sure that each step beforehand that we're basing this step off of will be optimal too.

If we have a bag filled this way, the final value at the end of the loop will be a bag with the highest value possible - the optimal solution to the problem.

This is essentially identical to part 2. We just include as many copies of each item as we have specified as item.number

My test case generator is the same as in part 3. It works by creating a list of random length, containing items with random weights and values and number of instances, and then picking a random weight limit. This is then put through the algorithm and the results are analysed to check correctness.

We can then check the barometer results against the inputs to determine complexity.

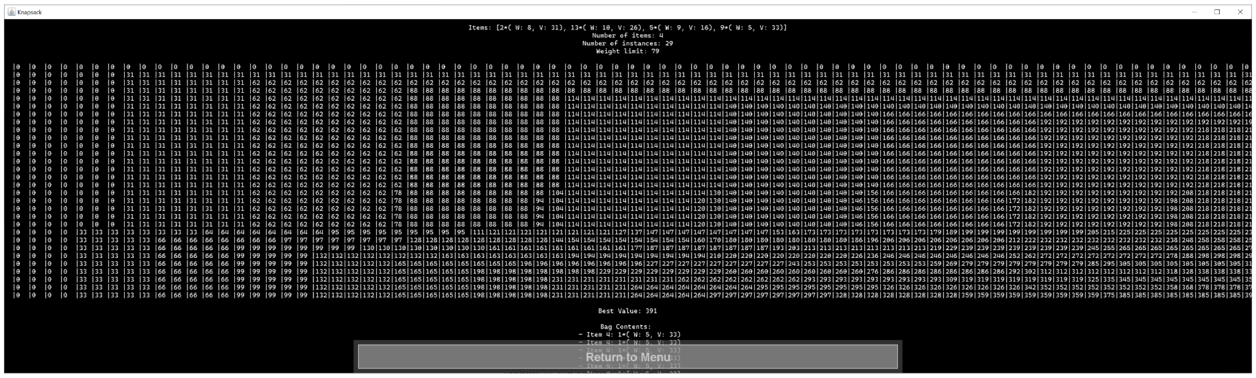
Our expected complexity should be N*(M+1) where N is the size of the list of instances, and M is the weight limit. This translates to Big O complexity of O(nm)

Based on testing;

- Barometer: 3700, Num instances: 50, Weight Limit: 73
    - 50*(73+1) = 3700
- Barometer: 2320, Num instances: 29, Weight Limit: 79
    - 29*(79+1) = 2320
- Barometer: 4324, Num instances: 46, Weight Limit: 93
    - 46*(93+1) = 4324
- Barometer: 1836, Num instances: 34, Weight Limit: 53
    - 34*(53+1) = 1836

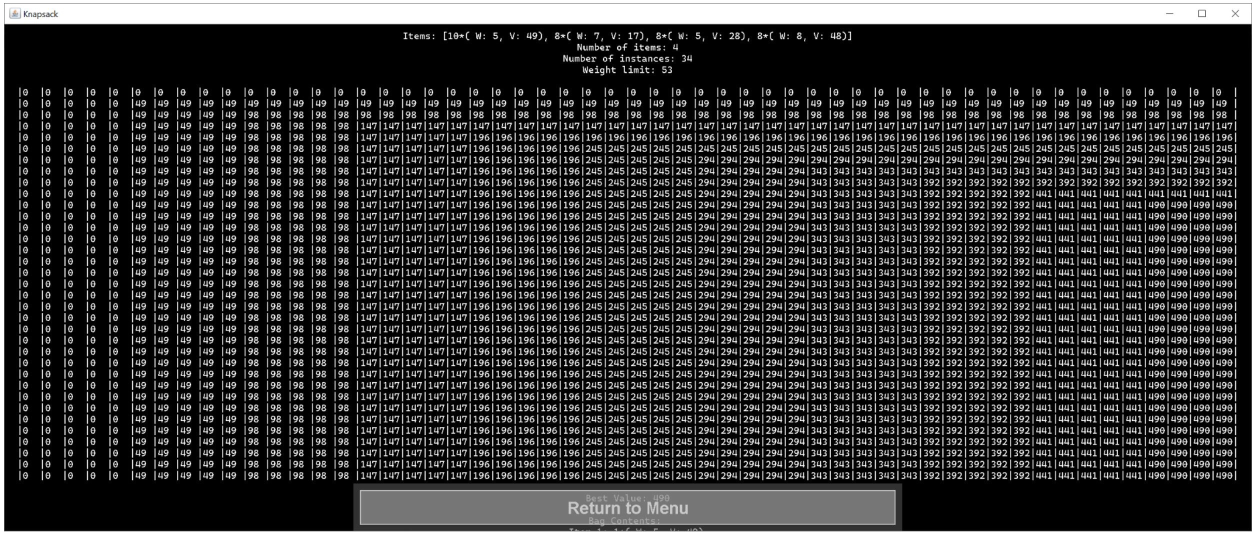
My estimation is spot on.

Complexity O(nm)

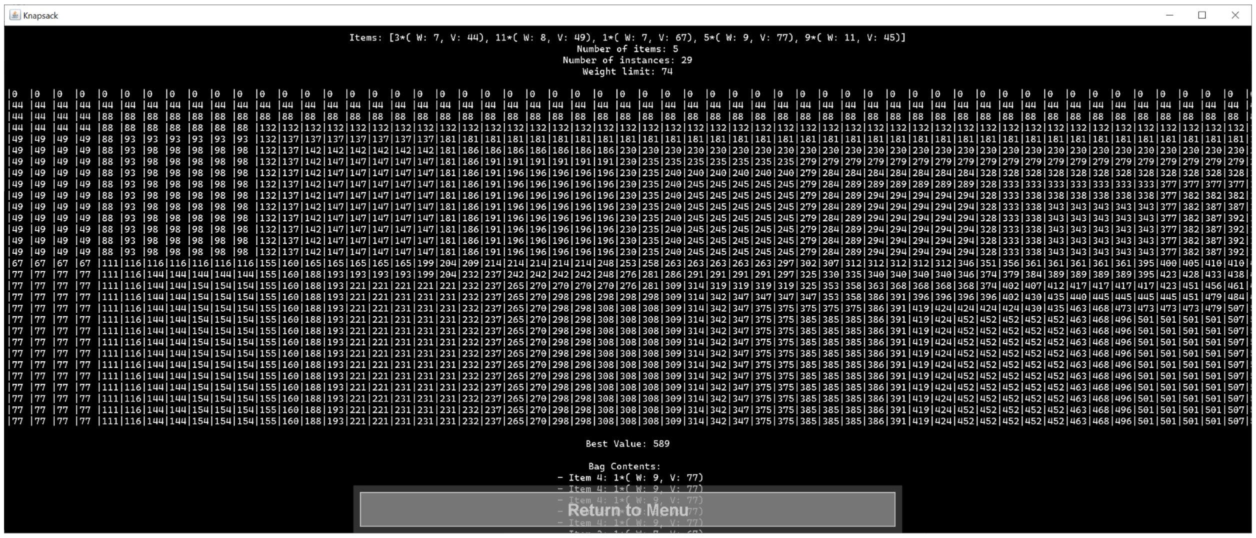


Knapsack

```
Items: [2*( W: 8, V: 31), 13*( W: 10, V: 26), 5*( W: 9, V: 16), 9*( W: 5, V: 33)]
Number of items: 4
Number of instances: 29
Weight limit: 79

|0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31 |31
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|114|11
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|140|14
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|166|166|166|166|166|166|166|166|166|166|166|166|166|16
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|192|192|192|192|192|192|192|192|192|192|192|192|192|19
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|192|192|192|192|192|192|192|192|192|192|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|192|192|192|192|192|192|192|192|192|192|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|192|192|192|192|192|192|192|192|192|192|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|192|192|192|192|192|192|192|192|192|192|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|192|192|192|192|192|192|192|192|192|192|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|192|192|192|192|192|192|192|192|192|192|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|192|192|192|192|192|192|192|192|192|192|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |62 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |114|114|114|114|114|114|114|114|114|114|140|140|140|140|140|140|140|140|140|140|166|166|166|166|166|166|166|166|166|166|192|192|192|192|192|192|192|192|192|192|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |78 |88 |88 |88 |88 |88 |88 |88 |88 |88 |104|114|114|114|114|114|114|114|114|114|130|140|140|140|140|140|140|140|140|140|156|166|166|166|166|166|166|166|166|166|182|192|192|192|192|192|192|192|192|192|208|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |78 |88 |88 |88 |88 |88 |88 |88 |88 |94 |104|114|114|114|114|114|114|114|114|120|130|140|140|140|140|140|140|140|140|146|156|166|166|166|166|166|166|166|166|172|182|192|192|192|192|192|192|192|192|198|208|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |78 |88 |88 |88 |88 |88 |88 |88 |88 |94 |104|114|114|114|114|114|114|114|114|120|130|140|140|140|140|140|140|140|140|146|156|166|166|166|166|166|166|166|166|172|182|192|192|192|192|192|192|192|192|198|208|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |78 |88 |88 |88 |88 |88 |88 |88 |88 |94 |104|114|114|114|114|114|114|114|114|120|130|140|140|140|140|140|140|140|140|146|156|166|166|166|166|166|166|166|166|172|182|192|192|192|192|192|192|192|192|198|208|218|218|218|21
|0  |0  |0  |0  |0  |0  |0  |31 |31 |31 |31 |31 |31 |31 |31 |62 |62 |62 |62 |62 |62 |62 |62 |62 |78 |88 |88 |88 |88 |88 |88 |88 |88 |94 |104|114|114|114|114|114|114|114|114|120|130|140|140|140|140|140|140|140|140|146|156|166|166|166|166|166|166|166|166|172|182|192|192|192|192|192|192|192|192|198|208|218|218|218|21
|0  |0  |0  |0  |33 |33 |33 |33 |33 |33 |33 |33 |64 |64 |64 |64 |64 |64 |64 |64 |95 |95 |95 |95 |95 |95 |95 |95 |95 |111|121|121|121|121|121|121|121|121|127|137|147|147|147|147|147|147|147|147|153|163|173|173|173|173|173|173|173|173|179|189|199|199|199|199|199|199|199|199|205|215|225|225|225|225|225|225|225|225|23
|0  |0  |0  |0  |33 |33 |33 |33 |33 |66 |66 |66 |66 |66 |66 |66 |66 |97 |97 |97 |97 |97 |97 |97 |97 |128|128|128|128|128|128|128|128|128|144|154|154|154|154|154|154|154|154|160|170|180|180|180|180|180|180|180|180|186|196|206|206|206|206|206|206|206|206|212|222|232|232|232|232|232|232|232|232|238|248|258|258|258|25
|0  |0  |0  |0  |33 |33 |33 |33 |33 |66 |66 |66 |66 |66 |99 |99 |99 |99 |99 |99 |99 |99 |130|130|130|130|130|130|130|130|161|161|161|161|161|161|161|161|161|177|187|187|187|187|187|187|187|187|193|203|213|213|213|213|213|213|213|213|219|229|239|239|239|239|239|239|239|239|245|255|265|265|265|265|265|265|265|265|27
|0  |0  |0  |0  |33 |33 |33 |33 |33 |66 |66 |66 |66 |66 |99 |99 |99 |99 |99 |132|132|132|132|132|132|132|132|163|163|163|163|163|163|163|163|194|194|194|194|194|194|194|194|194|210|220|220|220|220|220|220|220|220|226|236|246|246|246|246|246|246|246|246|252|262|272|272|272|272|272|272|272|272|278|288|298|298|298|29
|0  |0  |0  |0  |33 |33 |33 |33 |33 |66 |66 |66 |66 |66 |99 |99 |99 |99 |99 |132|132|132|132|132|165|165|165|165|165|165|165|165|196|196|196|196|196|196|196|196|227|227|227|227|227|227|227|227|227|243|253|253|253|253|253|253|253|253|259|269|279|279|279|279|279|279|279|279|285|295|305|305|305|305|305|305|305|305|31
|0  |0  |0  |0  |33 |33 |33 |33 |33 |66 |66 |66 |66 |66 |99 |99 |99 |99 |99 |132|132|132|132|132|165|165|165|165|165|198|198|198|198|198|198|198|198|229|229|229|229|229|229|229|229|260|260|260|260|260|260|260|260|260|276|286|286|286|286|286|286|286|286|292|302|312|312|312|312|312|312|312|312|318|328|338|338|338|33
|0  |0  |0  |0  |33 |33 |33 |33 |33 |66 |66 |66 |66 |66 |99 |99 |99 |99 |99 |132|132|132|132|132|165|165|165|165|165|198|198|198|198|198|231|231|231|231|231|231|231|231|262|262|262|262|262|262|262|262|293|293|293|293|293|293|293|293|293|309|319|319|319|319|319|319|319|319|325|335|345|345|345|345|345|345|345|345|35
|0  |0  |0  |0  |33 |33 |33 |33 |33 |66 |66 |66 |66 |66 |99 |99 |99 |99 |99 |132|132|132|132|132|165|165|165|165|165|198|198|198|198|198|231|231|231|231|231|264|264|264|264|264|264|264|264|295|295|295|295|295|295|295|295|326|326|326|326|326|326|326|326|326|342|352|352|352|352|352|352|352|352|358|368|378|378|378|37
|0  |0  |0  |0  |33 |33 |33 |33 |33 |66 |66 |66 |66 |66 |99 |99 |99 |99 |99 |132|132|132|132|132|165|165|165|165|165|198|198|198|198|198|231|231|231|231|231|264|264|264|264|264|297|297|297|297|297|297|297|297|328|328|328|328|328|328|328|328|359|359|359|359|359|359|359|359|359|375|385|385|385|385|385|385|385|385|39

Best Value: 391

Bag Contents:
- Item 4: 1*( W: 5, V: 33)
- Item 4: 1*( W: 5, V: 33)
- Item 4: 1*( W: 5, V: 33)
- Item 4: 1*( W: 5, V: 33)
- Item 4: 1*( W: 5, V: 33)
```



```
Items: [10*( W: 5, V: 49), 8*( W: 7, V: 17), 8*( W: 5, V: 28), 8*( W: 8, V: 48)]
Number of items: 4
Number of instances: 34
Weight limit: 53

|0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |49 |
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |98 |
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|147|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|196|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|245|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|294|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|343|343|343|343|343|343|343|343|343|343|343|343|343|343|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|392|392|392|392|392|392|392|392|392|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|441|441|441|441|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|
|0  |0  |0  |0  |0  |49 |49 |49 |49 |49 |98 |98 |98 |98 |98 |147|147|147|147|147|196|196|196|196|196|245|245|245|245|245|294|294|294|294|294|343|343|343|343|343|392|392|392|392|392|441|441|441|441|441|490|490|490|490|

Best Value: 490

Bag Contents:
Item 1: 1*( W: 5, V: 49)
```



```
Items: [3*( W: 7, V: 44), 11*( W: 8, V: 49), 1*( W: 7, V: 67), 5*( W: 9, V: 77), 9*( W: 11, V: 45)]
Number of items: 5
Number of instances: 29
Weight limit: 74

|0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |0  |
|44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |44 |
|44 |44 |44 |44 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |88 |
|44 |44 |44 |44 |88 |88 |88 |88 |88 |88 |88 |132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|132|
|49 |49 |49 |49 |88 |93 |93 |93 |93 |93 |93 |132|137|137|137|137|137|137|137|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|181|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|142|142|142|142|142|181|186|186|186|186|186|186|186|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|230|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|147|147|147|147|147|181|186|191|191|191|191|191|191|230|235|235|235|235|235|235|235|279|279|279|279|279|279|279|279|279|279|279|279|279|279|279|279|279|279|279|279|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|147|147|147|147|147|181|186|191|196|196|196|196|196|230|235|240|240|240|240|240|240|279|284|284|284|284|284|284|284|328|328|328|328|328|328|328|328|328|328|328|328|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|147|147|147|147|147|181|186|191|196|196|196|196|196|230|235|240|245|245|245|245|245|279|284|289|289|289|289|289|289|328|333|333|333|333|333|333|333|377|377|377|377|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|147|147|147|147|147|181|186|191|196|196|196|196|196|230|235|240|245|245|245|245|245|279|284|289|294|294|294|294|294|328|333|338|338|338|338|338|338|377|382|382|382|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|147|147|147|147|147|181|186|191|196|196|196|196|196|230|235|240|245|245|245|245|245|279|284|289|294|294|294|294|294|328|333|338|343|343|343|343|343|377|382|387|387|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|147|147|147|147|147|181|186|191|196|196|196|196|196|230|235|240|245|245|245|245|245|279|284|289|294|294|294|294|294|328|333|338|343|343|343|343|343|377|382|387|392|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|147|147|147|147|147|181|186|191|196|196|196|196|196|230|235|240|245|245|245|245|245|279|284|289|294|294|294|294|294|328|333|338|343|343|343|343|343|377|382|387|392|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|147|147|147|147|147|181|186|191|196|196|196|196|196|230|235|240|245|245|245|245|245|279|284|289|294|294|294|294|294|328|333|338|343|343|343|343|343|377|382|387|392|
|49 |49 |49 |49 |88 |93 |98 |98 |98 |98 |98 |132|137|142|147|147|147|147|147|181|186|191|196|196|196|196|196|230|235|240|245|245|245|245|245|279|284|289|294|294|294|294|294|328|333|338|343|343|343|343|343|377|382|387|392|
|67 |67 |67 |67 |111|116|116|116|116|116|116|155|160|165|165|165|165|165|199|204|209|214|214|214|214|214|248|253|258|263|263|263|263|263|297|302|307|312|312|312|312|312|346|351|356|361|361|361|361|361|395|400|405|410|410|
|77 |77 |77 |77 |111|116|144|144|144|144|144|155|160|188|193|193|193|193|199|204|232|237|242|242|242|242|248|276|281|286|291|291|291|291|297|325|330|335|340|340|340|340|346|374|379|384|389|389|389|389|395|423|428|433|438|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|221|221|221|232|237|265|270|270|270|270|276|281|309|314|319|319|319|319|325|353|358|363|368|368|368|368|374|402|407|412|417|417|417|417|423|451|456|461|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|298|298|298|309|314|342|347|347|347|347|353|358|386|391|396|396|396|396|402|430|435|440|445|445|445|445|451|479|484|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|375|375|375|386|391|419|424|424|424|424|430|435|463|468|473|473|473|473|479|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|
|77 |77 |77 |77 |111|116|144|144|154|154|154|155|160|188|193|221|221|231|231|231|232|237|265|270|298|298|308|308|308|309|314|342|347|375|375|385|385|385|386|391|419|424|452|452|452|452|452|463|468|496|501|501|501|501|507|

Best Value: 589

Bag Contents:
- Item 4: 1*( W: 9, V: 77)
- Item 4: 1*( W: 9, V: 77)
- Item 4: 1*( W: 9, V: 77)
- Item 4: 1*( W: 9, V: 77)
- Item 4: 1*( W: 9, V: 77)
- Item 3: 1*( W: 7, V: 67)
```

5:

```
calculate(Item[] items, int limit) {
    items.sort //sort the items so we can do a best-first-search
    node[items.length+2] <- 0 //initialise everything to zero
    //node stores how many of each item we take, followed by the running total of
    //weight and value
    node = traverse(node)
    taking[i] <- node[i]*items[i] //take the number of each item as given in node
    return node[node.length-1] //return the best value from the traversal
}
traverse(int[] node) {
    <- Barometer here (each time we traverse the graph)
    If (visited(node)) return node
    bestNode = node;
    for (i = 0; < node.length-2; ++) {
        newNode = node (with node[i]+=1)
        newV = newNode[length-2]+items[i].value
        newW = newNode[length-1]+items[i].weight

        If (pruning) continue

        bestNode = best of (traverse(newNode)) or (bestNode)
    }
    return bestNode
}
```

We sort the items, so we can do some simple heuristic work by default. Then we recursively traverse through the graph until we've found the node with the highest value in the graph, with a weight below the limit.

We prune based on whether or not the next node will break the weight limit or if we've run out of items to add in that dimension. This ensures we never break the rules.

We also prune based on if the path started by this traversal can possibly return a better result than what we already have. If not there's no point looking

Then we simply look at every node left, and take the one with the best value. The only way this would give us a sub-optimal answer is if we accidentally prune the best answer. However, we only prune branches we know do not contain the best answer. So our algorithm will always return an optimal answer.

My test case generator is the same as in part 3. It works by creating a list of random length, containing items with random weights and values and number of instances, and then picking a random weight limit. This is then put through the algorithm and the results are analysed to check correctness.

We can then check the barometer results against the inputs to determine complexity.

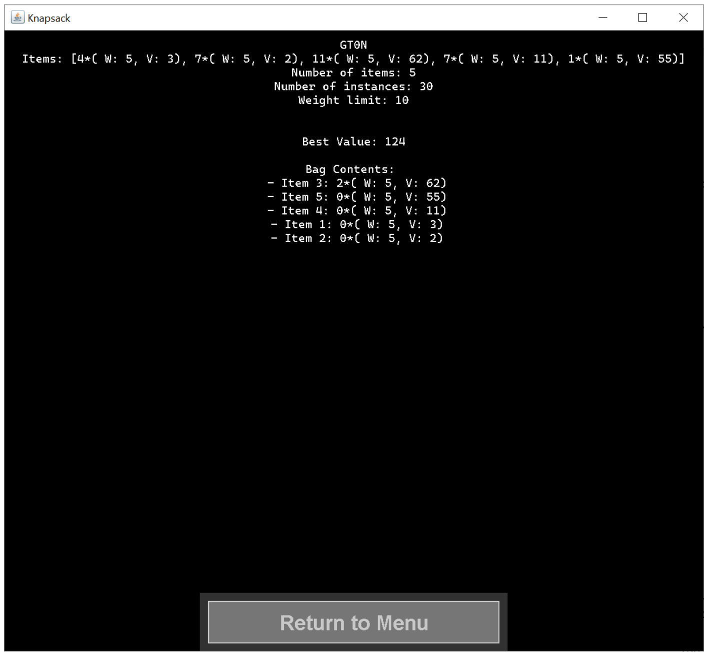
Our expected worst-case complexity should be every combination of instances of each item I^N where I is the number of instances of each item, and N is the number of items. This translates to Big O complexity of O(i^n) In practice, it should be a lot lower than this as we do a lot of pruning.

Based on testing;

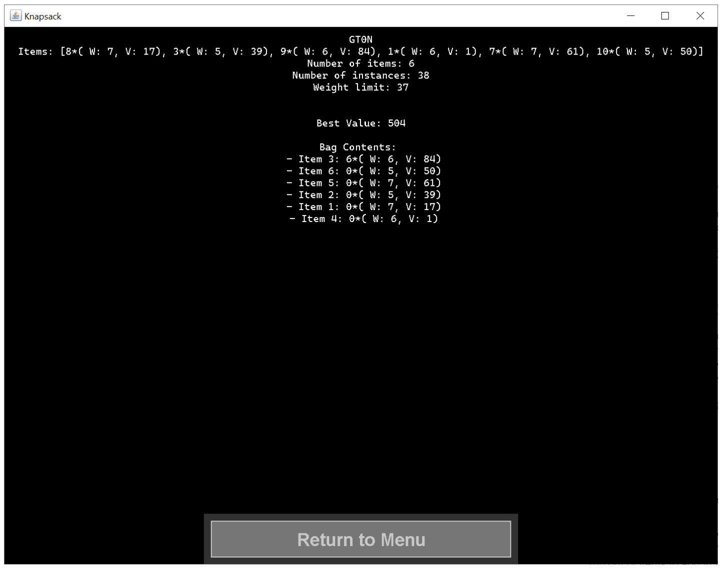
- Barometer: 9656, Num instances: 43, Num Items: 6
  - $(43/6 \approx 7)^6 \approx 117,649 > 9656$
- Barometer: 1037, Num instances: 28, Num Items: 4
  - $(28/4 = 7)^4 = 2401 > 1037$
- Barometer: 10625, Num instances: 41, Num Items: 5
  - $(41/5 \approx 8)^5 \approx 37074 > 10625$
- Barometer: 30, Num instances: 30, Num Items: 5
  - $(30/5 = 6)^5 = 7776 > 30$

My estimation is correct.

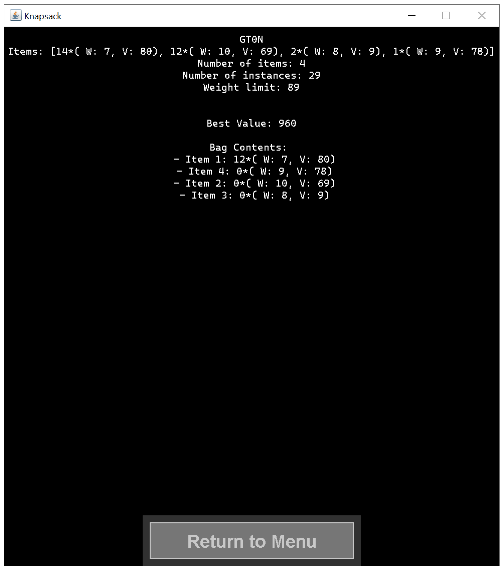
Complexity is somewhere lower than O(i^n)



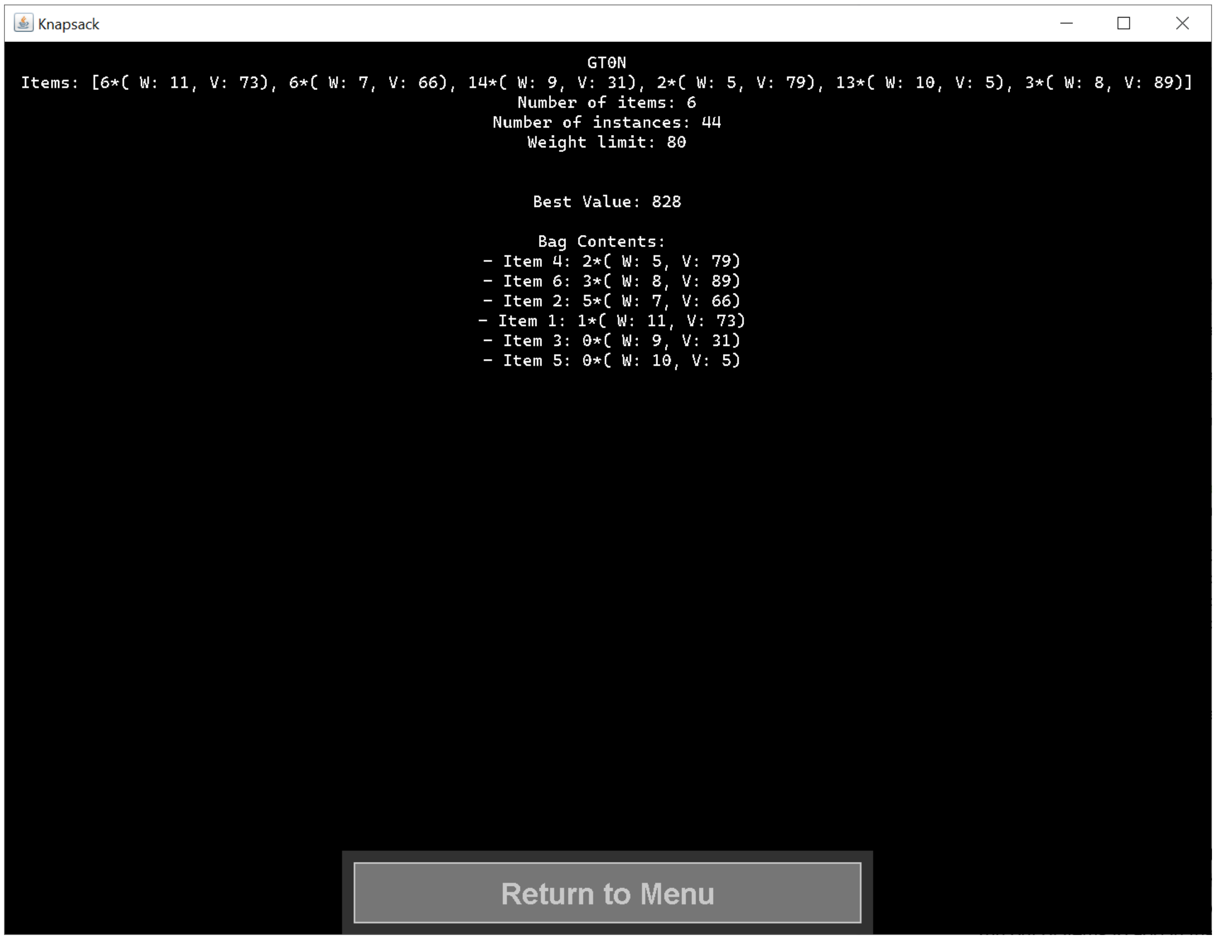
```
GT0N
Items: [4*( W: 5, V: 3), 7*( W: 5, V: 2), 11*( W: 5, V: 62), 7*( W: 5, V: 11), 1*( W: 5, V: 55)]
Number of items: 5
Number of instances: 30
Weight limit: 10


Best Value: 124

Bag Contents:
- Item 3: 2*( W: 5, V: 62)
- Item 5: 0*( W: 5, V: 55)
- Item 4: 0*( W: 5, V: 11)
- Item 1: 0*( W: 5, V: 3)
- Item 2: 0*( W: 5, V: 2)
```



```
GT0N
Items: [8*( W: 7, V: 17), 3*( W: 5, V: 39), 9*( W: 6, V: 84), 1*( W: 6, V: 1), 7*( W: 7, V: 61), 10*( W: 5, V: 50)]
Number of items: 6
Number of instances: 38
Weight limit: 37


Best Value: 504

Bag Contents:
- Item 3: 6*( W: 6, V: 84)
- Item 6: 0*( W: 5, V: 50)
- Item 5: 0*( W: 7, V: 61)
- Item 2: 0*( W: 5, V: 39)
- Item 1: 0*( W: 7, V: 17)
- Item 4: 0*( W: 6, V: 1)
```



```
GT0N
Items: [14*( W: 7, V: 80), 12*( W: 10, V: 69), 2*( W: 8, V: 9), 1*( W: 9, V: 78)]
Number of items: 4
Number of instances: 29
Weight limit: 89


Best Value: 960

Bag Contents:
- Item 1: 12*( W: 7, V: 80)
- Item 4: 0*( W: 9, V: 78)
- Item 2: 0*( W: 10, V: 69)
- Item 3: 0*( W: 8, V: 9)
```



```
GT0N
Items: [6*( W: 11, V: 73), 6*( W: 7, V: 66), 14*( W: 9, V: 31), 2*( W: 5, V: 79), 13*( W: 10, V: 5), 3*( W: 8, V: 89)]
Number of items: 6
Number of instances: 44
Weight limit: 80


Best Value: 828

Bag Contents:
- Item 4: 2*( W: 5, V: 79)
- Item 6: 3*( W: 8, V: 89)
- Item 2: 5*( W: 7, V: 66)
- Item 1: 1*( W: 11, V: 73)
- Item 3: 0*( W: 9, V: 31)
- Item 5: 0*( W: 10, V: 5)
```

6:

1.
    a.
```
explore(Node) {
        If (Node.isExplored) return List.of(Node);
        Node.explore;
        For (Edge : Node.neighbours) {
              If (Edge.isExplored) continue;
              Edge.explore;
              List l = explore(Edge.node);
              If (l != null) {
                    If (l.contains(Node)) print(l);//cycle
                    l.add(Node);
                    return l;
              }
        }
        return null;
}
```
    b. This algorithm has a time complexity of O(m+n).
    This is because in the case of no cycles, the worst case scenario, we will explore every node connected in the graph. This is a complexity of n.
    But for each node, we explore each of its neighbouring edges too - exactly once, for a complexity of O(m)
    This all leads to a complexity of O(n+m)

2. Base case: one leaf, no nodes.
   Each step, take all the leaf nodes, and turn them into nodes with two new leaf nodes for children.
   Assuming step k follows the rules, with n nodes, and e leaves, step k+1 now has:
    - Nodes, N: n+e
    - Leaves, E: 2*e

   We have, by assumption, e=n+1.
   N = 2n+1
   E = 2n+2

   E=N+1

3. Assume G does contain an edge not present in T. By theorem in pg 19 of graphs-algorithms lecture slides, this edge, (x, y) for some x,y ∍ V, must hold that either x or y is an ancestor of the other in depth first search.
   If we now look at BFS, we have two cases; x is an ancestor of y; or y is an ancestor of x in DFS.
   Case 1 (x is ancestor of y in DFS):
      When we start calculating x in BFS, we look at each of x's children, which includes y's ancestor (from DFS), and y itself (from edge (x,y)). These are both added to the queue.
      y will now be expanded at the same time as y's ancestor. In other words, y's ancestor will no longer be y's ancestor in BFS.

      DFS does not equal BFS. Contradiction.
   Case 2 (y is ancestor of x in DFS):
      This case is identical by symmetry.
      DFS does not equal BFS. Contradiction
   G cannot contain any edges not found in T.
   G=T

4. To show NPC, we need to translate from one problem to the other. And show both are NP.
   Start with 0-1 Knapsack:
      Showing an answer is correct means adding all the weights of selected items. If the sum is $\leq$ W, then the answer is correct. Summing is in P, so 0-1 knapsack is NP.
      Now, translate from Subset Problem to 0-1 knapsack.
      Let each value in the input set for Subset Problem be an Item, where weight = value = the value of the number in the input set.
      Let the weight limit = the target number for the subset problem.
      Now, running the problem as a 0-1 knapsack problem will return the correct answer if one exists - it will return the largest value. It will never overestimate, as it cannot exceed the weight limit, so the largest available answer is the correct one. If there is an answer, knapsack can find it.
      0-1 knapsack is NPC.
   Now, 0-N Knapsack:
      Showing an answer is correct means adding all the weights of selected items. If the sum is $\leq$ W, then the answer is correct. Summing is in P, so 0-1 knapsack is NP.
      Now, translate 0-1 to 0-N. If we select N=1 for every item, 0-N = 0-1.
      0-N knapsack is NPC.

5. To show NP-Hard, we just need to translate from one to the other.
   Translating 0-N knapsack to integer linear programming:

   > Integer linear programming problem is to maximise linear function subject to constraints.
   > 0-N knapsack is to maximise linear function subject to one constraint.
   >
   > ∑Item profits(Ap,Bp,Cp,...) subject to ∑Item weights(Aw,Bw,Cw,...) $\leq$ W.
   >
   > This is a linear programming problem.

   Integer linear programming is NP-Hard.