Algorithm Design & Analysis
TCP2101
Fractional Knapsack Algorithm Analysis
TC02/TT05

Prepared By:

Ahmad Hamdan	1131121467
Mahmoud Abdelazim	1132702480

February 5, 2017

Contents

List of Figures

1 Fractional Knapsack Algorithm

1.1 Introduction

This report briefly discusses the fractional knapsack algorithm and shows an example of its implementation using c++. Then the output will be analysed through its output speed to measure efficiency. The analysis will provide examples of best case, worst case, and average case scenarios. Finally, the report will present a comparison of running time with input size to show how the algorithm deals with large inputs of data.

1.2 Fractional Knapsack Algorithm

The fractional knapsack algorithm was implemented in an attempt to solve the knapsack problem using a heap-based priority queue. Given a set of items, each with a weight,binifit and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible. Also you are allowed to break any item into parts or fractions to optimize the use of the knapsack.

1.3 Best, Average and Worst case scenarios

We have implemented the fractional knapsack algorithm using C++. The program prompts the user to input the number of items, and the weight of the knapsack to be simulated. Then a table of results is displayed to the user showing the sequence of items being inserted into the knapsack. We will show the best, worst and average case scenarios of the algorithm.

1.3.1 Best case:

The best case scenario describes the priority queue when it doesn't need to pop anything from it to place it in the knapsack.
in the best case T = O(1) case the size of knapsack is 0 and no items is going to be inserted to it that is going to take less time than the other .

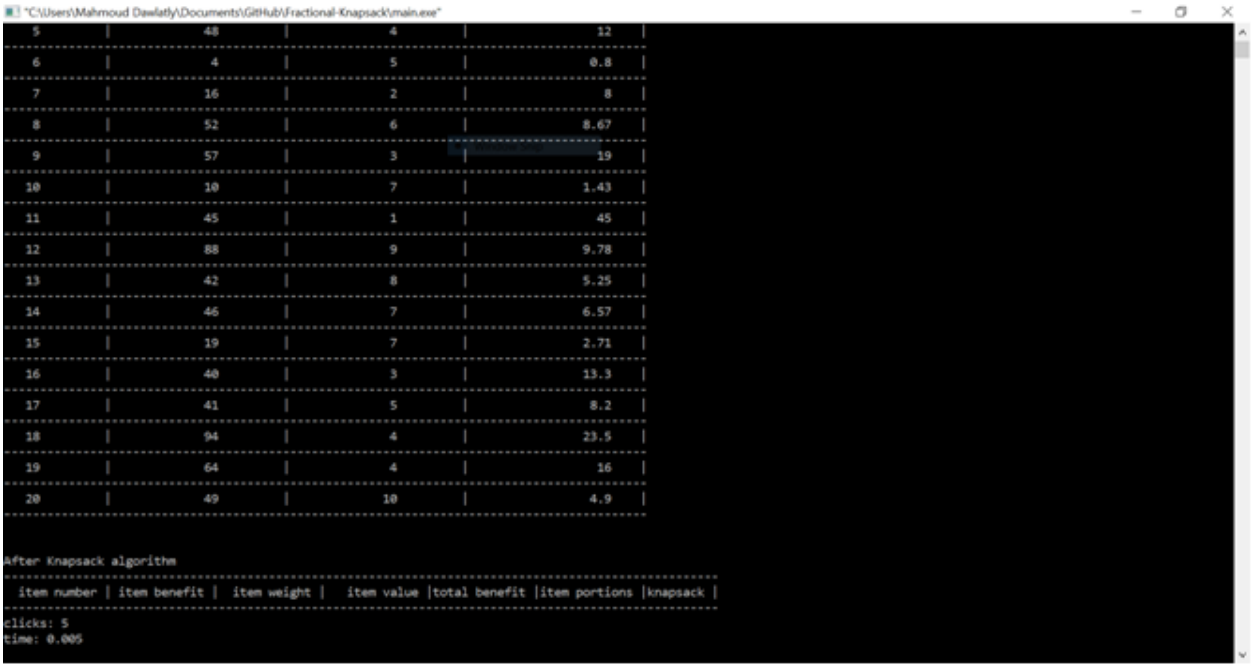


Figure 1.1: Best case scenario

1.3.2 Worst case:

On the other hand, the worst case scenario describes when all the items in the priority queue will be removed and placed in the knapsack. This is because it will take the longest time.
in the worst case the Knapsack size is very big to content the all items inside it and T = O(n). it is going the larget amount of time case the Knapsack is going to visit all the items and insert it .

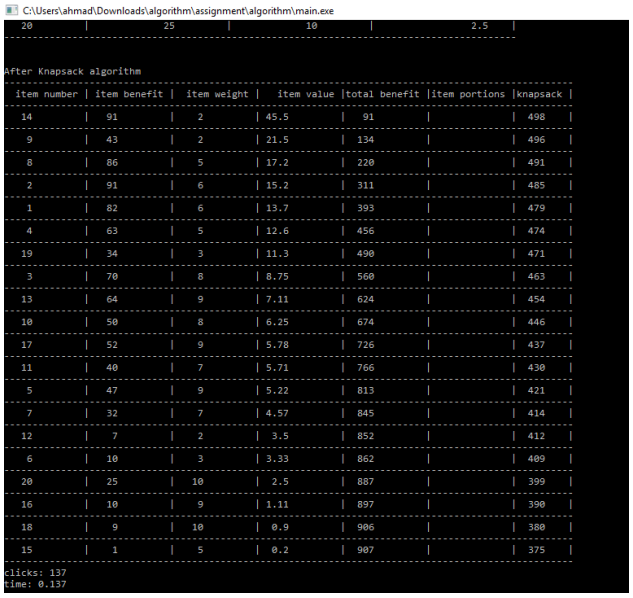


Figure 1.2: worst case scenario

1.3.3 Average case:

The average case scenario shows the cases in between the two extremes.
in the average case the knapsack is going to insert some items and get the portion of the items .

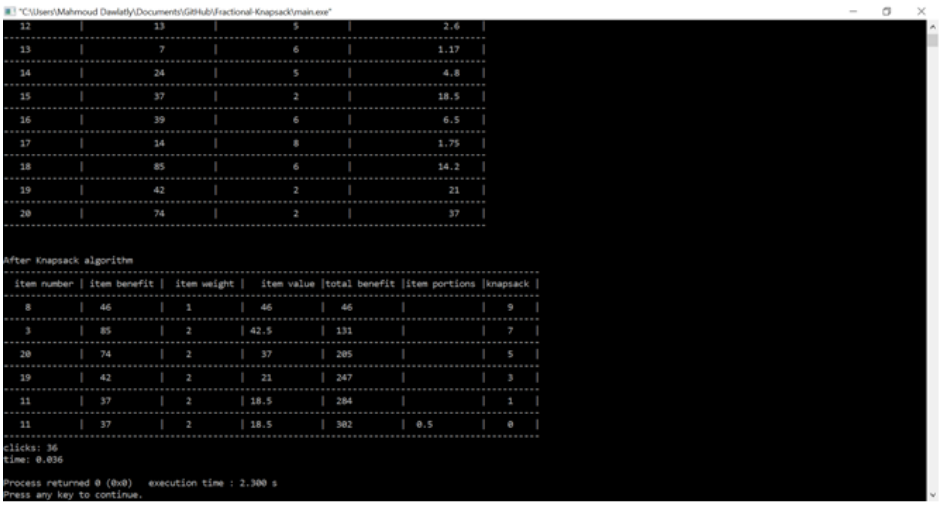
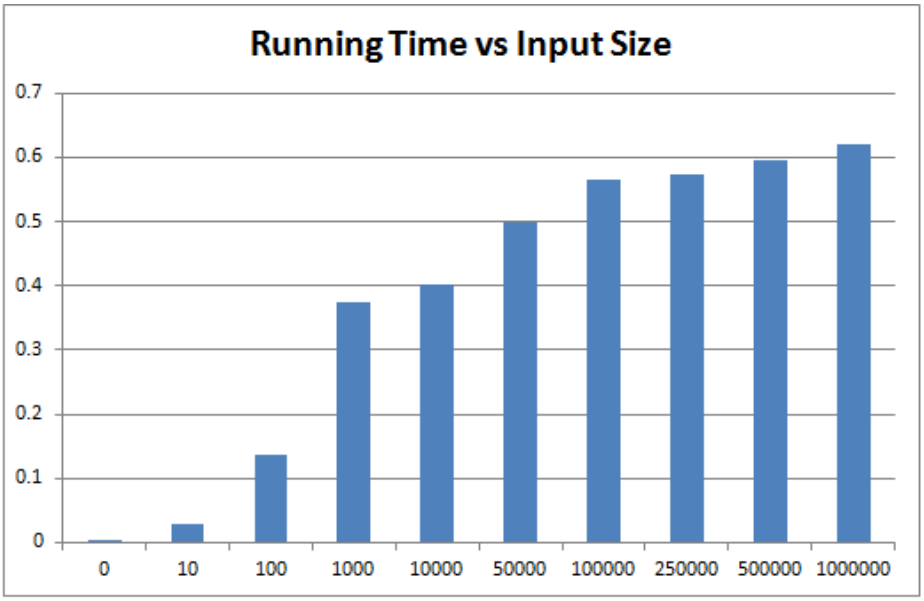


Figure 1.3: average case scenario

1.4 Experiment Results

This section will show the experiment results on the Knapsack code by running different items size and see the time as shown in the graph below.
and the table below explain clearly the number of items inserted and the time in seconds:



Input Size	Running Time
0	0.002505
10	0.028616
100	0.135732
1000	0.375037
10000	0.400939
50000	0.499864
100000	0.564289
250000	0.573452
500000	0.596718
1000000	0.619287

Figure 1.4: time and input size graph

the result approves that there is a relation between the input size and the time if the input size getting larger the time also is going to be larger.

1.5 conclusion

the code shows the 3 case senatrions in the Knapsack algorithm :Best, Worst and average and sgows the run time complixity for each one .
and shows also the time amount according to the input size, when the input size is getting larger the time also is getting lager.