Knapsack Runtime and Optimization

Collin Blanchard, Holly Buff
School of Information, Technology,
and Computing
Abilene Christian University
Abilene, Texas
{cab13e, hab13a}@acu.edu

ABSTRACT

This is going to be our abstract of the paper and we will wait to complete until after the entire paper is finished, as it will contain a summary of the entire paper.

CCS Concepts

Mathematics of computing→Combinatorics • Theory of computation→Sorting and searching.

Keywords

Knapsack; Combinatorial Optimization

1. INTRODUCTION

The knapsack problem (KP) is a common problem that is discussed in classroom settings. KP is also known as the rucksack problem but for this paper, the terminology that we will use will either be "knapsack problem" or "KP" (for short). KP is a combinatorial problem in which the result required is the most optimal result from a finite set of items. Many combinatorial problems are NP-complete meaning that there are currently no known algorithms that are both correct and fast on all possible cases. KP being a combinatorial problem is also NP-complete with a typical runtime of $O(n*W)$, where W is a restriction on weight, but in our code and in the rest of the paper we will refer to the weight as the cost. Despite the KP problem being NP-complete, the optimization for KP is actually NP-hard, meaning that there is no known polynomial time solution, and at least as difficult as the original, decision, part of KP. The purpose of this paper is to outline an experiment and test the runtime of different solutions for KP, as well as, introduce other possible optimizations for KP.

The rest of the paper is organized as follows. Section 2 is a description of the phases within the experiment. Sections 3-6 discuss the different phases within the experiment. Section 3 being Phase 1, Section 4 being Phase 2, Section 5 being Phase 3, and finally Section 6 being Phase 4. Section 7 focuses on the final results from all Phases of the experiment. The paper concludes with an overall summary and discussion on future works in Section 8. It should be noted that there were 5 phases to the experiment but the 5th and final phase was to complete this paper so we shall say that the experiment only has 4 actual phases.

DOI: http://dx.doi.org/10.1145/12345.67890

2. Description of Experiment

This experiment is comparing runtimes for different solutions to KP and consists of 4 phases. All programming and algorithms used in this experiment are implemented in C#. All test data is read in with a .csv file and the data consists of the maximum capacity on one line and every line following consists a name, the cost, and then the value of a specified item. We tested a variety of sample data such as: a knapsack containing 40 items with each item being more than half of the capacity such that only one item can be accepted into the optimal solution. Phase 1 consists of writing a greedy solution for knapsack and running it. Phase 2 consisted of adding an exhaustive binary search and running it. Phase 3 was tree pruning and optimizing Phase 2. Phase 4 consisted of adding a timer and then exporting the optimal solutions of the knapsacks into a text file.

3. Phase 1

3.1 Description of Phase 1

This phase of the experiment consists of creating 4 greedy solution to approximate the bounds for KP that we know we cannot do better or worse than. The first greedy solution is to first sort the knapsack with highest value first. The second greedy solution consists of sorting the knapsack with the lowest cost first. The third greedy solution consists of sorting the knapsack with the highest ratio first. The final greedy solution consists of a partial knapsack where once the capacity is nearly full, the algorithm will take a piece of the next item.

3.2 Implementation of Phase 1

In our phase 1 we implemented everything exactly to the specifications of the experiment instructions. It consisted of reading in the .csv files and applying the greedy solutions. We implemented in this phase and throughout all other phases the return type of a tuple so that we can return both a knapsack and a capacity separate from each other. We also used an item struct that contained the name cost and value to avoid the use of multiple parallel arrays.

4. Phase 2

4.1 Description of Phase 2

In this phase, the goal is to create an exhaustive search that guarantees an optimal answer for any given knapsack. However due to time constraints, we stopped any knapsack that had too many items such that the runtime took 10 minutes or longer. The plan for this phase was to create a binary tree that contained all possible sacks with all possible item combinations shown in Figure 1. The binary tree will consist of a first row that will be the root node and then the second and proceeding rows will consist of an item. The left child of and item or node will represent that the current item being looked at is not included in the sack and the right child will represent that the current item is included in the sack. We will then perform a depth-first search through the binary tree to add up the total value, costs, and final knapsack lists.

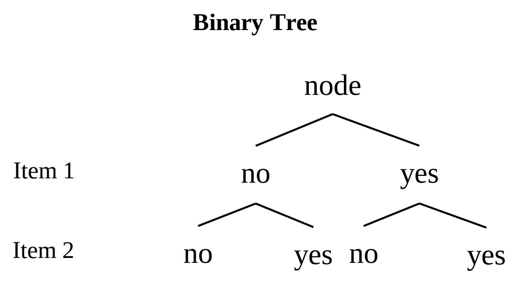

Figure 1. This is a visual example of the binary tree. Yes and no correspond to either accepting or not accepting the current item.

4.2 Implementation of Phase 2

It should be noted that within Phase 2, we deviated from the original plan of creating a binary tree for our exhaustive search algorithm.

Rather than using a binary tree we simply searched iteratively through the list of items using for loops. For testing purposes, during this phase, we would terminate the program whenever it ran past 2 minutes and we still had not received any output.

5. Phase 3

5.1 Original Plan for Phase 3

This phase is an addition to Phase 2, in which we add extra optimization (tree pruning) to the binary tree. Tree pruning that is planned to be implemented is that if a branch is over capacity then we drop that subtree. Also, if there is not enough value left in the subtree compared to the greedy solutions or best solutions we will also drop the subtree.

5.2 Implementation of Phase 3

Being that we deviated from the binary tree of Phase 2, there was not any tree pruning to do but there were a few optimizations that we could still make. Within our optimizations we pass in our greedy minimum and then we calculate the total value remaining for the rest of the knapsack within the current iteration. We then make 2 comparisons, one condition checks to see if we are over capacity at any point and then breaks out of that for loop and continues with the next item in the list. Within the same condition, we check to see if the remaining value is less than the greedy minimum and will also break out of the loop and continue to the next item. Another optimization that we implemented was that we ordered the list in a value over cost ratio in ascending order.

6. Phase 4

6.1 Original Plan for Phase 4

Phase 4 is also an addition to the previous phases. In this phase we were required to add a timer into the program in order to calculate the time that all solutions would take individually. We also are required to export to a file all of our times and solutions for KP.

6.2 Implementation for Phase 4

In this phase we implemented a timer into the code so that we can record the time that the exhaustive search takes. Not only did we implement a timer but we made a cut off so that the program is terminated after 10 minutes of searching. We also implemented a file output. Here we take the knapsack list and convert it into a string for easy file output.

7. Result

The heading of subsections should be in Times New Roman 12-point bold with only the initial letters capitalized. (Note: For subsections and subsubsections, a word like *the* or *a* is not capitalized unless it is the first word of the header.)

8. Concluding Remarks and Further Works

Implementation

9. ACKNOWLEDGMENTS

Our thanks to ACM SIGCHI for allowing us to modify templates they had developed. In addition, we would like to formally thank Dr. Ray Pettit for being an excellent professor and imparting his knowledge and wisdom onto us.

10. REFERENCES

[1] Stuart J. Russell and Peter Norvig. 2016. *Artificial Intelligence A Modern Approach* 3rd ed., Boston: Pearson.