Maximizing Warung Tegal Profit with Branch and Bound Algorithm

Wenny Yustalim (13515002)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13515002@std.stei.itb.ac.id

***Abstract*—Choosing the raw ingredients in order to satisfy customers' demand in a Warung Tegal needs several points of consideration. In business, the most important principle is extracting the maximum revenue out of the minimum expenses. But in order to maintain the sustainability of a warteg, one should never neglect the basic requirements that the customers should receive. We should not close our eyes towards the fullfilment of nutrition, especially coming from Warung Tegal, one of the most commonly found food station in our countries. This paper discusses the algorithmic strategy implemented in choosing the most profitable yet nutritionally balanced ingredients for Warung Tegal all around our beloved country.**

Keywords—Branch and Bound; Knapsack; maximum profit; warung Tegal.

I. INTRODUCTION

Warteg or *warung tegal* is generally a small shop that sells Javanese food and provides spots for customers to sit and dine. It was established by Javanese people from the town Tegal in Central Java. They sell favourite Javanese dishes and rice, the wide variety of pre-cooked dishes are arranged in a glass case cupboard. They are well known on selling modestly-priced meals in large portion, hence making them popular among all layers of society.

Warung Tegal has been the simplest solution anybody can find when hunger strikes. It is typically seen as a small family-owned business in Indonesia and can be found in almost any narrow streets in any city. It is now inseparable to the daily lives of millions of people in Indonesia, regardless of what background the person comes from. Anybody ranging from a school kid, a university student, police officer, and even white-collar workers can be easily seen blending in a warteg, eating *tempe* with his or her bare hands. In Sumatra and Malay Peninsula, warteg may also be referred as *kedai.*

As its name suggests, warteg has its roots in the city of Tegal in Central Java. Initially run by the citizens of the villages of Sidapurna, Sidakaton and Krandon, its management would rotate once every three to four months and would come from the same family. This custom is still followed even in large cities such as Jakarta or Bandung, and is actually done to avoid boredom by staying at the same place. Those whose turn is not yet up would fill their time farming. [1]

People are addicted to the food in warteg, mainly because the food provided usually tastes like home. Especially in campus area in Indonesia, where most of the students have limited monthly budget to survive, they tend to choose warteg for breakfast, lunch, and even dinner, without even considering the cleanliness, the risks and the nutrition that they need to fulfill every single day. Most of the dishes served by warteg come in large portion.

Figure 1. Variety of people can be seen eating in a warteg
Source: http://4.bp.blogspot.com/-J84DpJHGOs0/U-nZV6DT6PI/AAAAAAAAEv0/dC_gKTN5r1I/s1600/warteg.png

In order for any business to sustain, they have to carefully calculate their expenses and their incomes. They should also consider the most strategic spot where many people pass by. There are a lot of things that they should consider when they go to the traditional market to buy ingredients for the day. Some goes to the market everyday, some every two days, and some weekly. Nevertheless, everytime they compare one ingredient from another, they always have to think which ingredients would give them the maximum profit in return, without neglecting the nutritional balance of the food they serve.

The daily nutrition intake one must have in order to be categorized as 'eating well' consists of carbohydrates, such as bread, rice, potatoes, or pasta, protein that may come from meat, fish, eggs or beans, fruits, vegetables, milk or other dairy products, healthy fat and sugar.

It is not easy to combine all of the ingredients together with a limited budget. The resource that they have mostly relies on the previous day's earnings.

This is where technology has to step in. Sometimes we always think about how to make the world a better place using the rise of technology, but we fail to look left and right, to try and solve the problems in our own local community.

Many of the owners of warteg, especially those who come from low-educated background, may have difficulties to cherrypick the ingredients they want to present the next day. They buy their food mainly based on instincts and feelings. Some owners who want to gain more out of their budget bring calculator to the traditional market to calculate how much profit he or she will gain from buying a particular ingredient.

This problem strongly resembles the Knapsack Problem. There are many approaches to solve the Knapsack Problem, which will be discussed further on this paper.

II. Basic Concepts and Theories

A. *Tree*

A tree is a directed graph that is widely used in computer science as an abstract data type (ADT). It has a root value and subtrees of children with a parent node that may be represented by a linked node.

This type of data structure is usually defined recursively as a set of nodes that starts from a root node. Each node contains a value or state. A tree is most commonly represented as an adjacency list of edges between nodes, hence a directed graph. In general, the data structure of a given node only contains the list of its children, without any reference to the parent.

It is made up of nodes/vertices and edges, without any cycle. An empty tree is a tree with no nodes. Sibling is a group of nodes with the same parent. Leaf is a node with no children. Degree is the number of sub trees of a node. Edge is the connection between one node and another. Depth is the number of edges from the root node to the node itself.

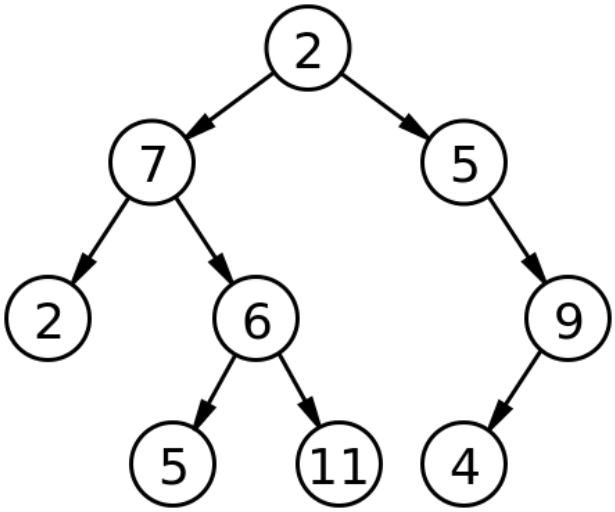


Figure 2. A tree represented by a directed graph with no cycle. Node 6 has 2 children, labelled 5 and 11. The root node has no parent.
Source: https://en.wikipedia.org/wiki/File:Binary_tree.svg

B. *Knapsack problem*

The knapsack problem or rucksack problem is a problem where there exists a rucksack with a certain weight capacity, along with several objects with different weights and values. The objects should be fit inside the bag, without the bag being overweight; the total weight should be less than or equal to the weight limit (capacity), but the value should be as much as possible. This is an example of a combinatorial optimization. Just like the name of the problem itself, it depicts the situation where someone is constrained by a fixed-size knapsack and must fill it with the most valuable items.

This problem often arises in resource allocation where there are financial constraints and is studied in fields such as combinatorics, computer science, complexity theory, cryptography, applied mathematics, and daily fantasy sports.

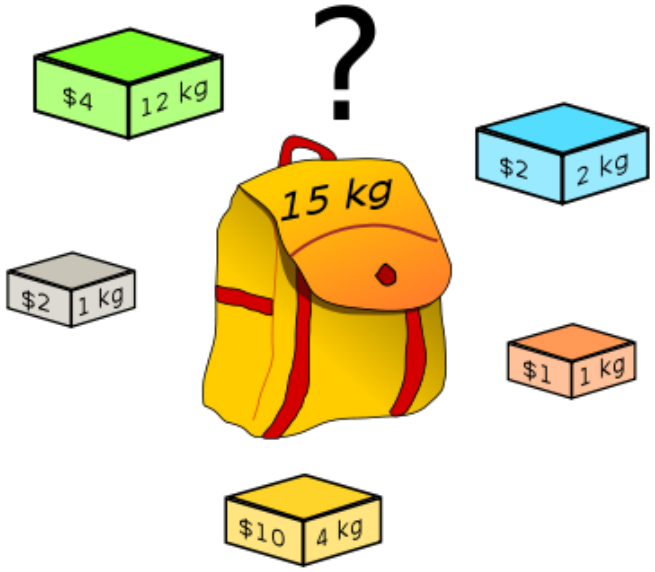


Figure 3. Rucksack with a capacity of 15kg to be filled with 5 boxes of different weight and values
Source: https://motherboard-images.vice.com/content-images/contentimage/no-id/1431232724509610.png

The knapsack problem has been studied for more than a century, with early works dating as far back as 1897.[3] The name "knapsack problem" dates back to the early works of mathematician Tobias Dantzig (1884–1956),[2] and refers to the commonplace problem of packing your most valuable or useful items without overloading your luggage.

The most common problem being solved in the field of informatics is the 0-1 (integer) knapsack problem, which restricts the number x_i of copies of each kind of item to zero or one. Given a set of n items numbered from 1 up to n, each with a weight w_i and a value v_i, along with a maximum weight capacity W,[3]

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \in \{0, 1\}.$$

Here xi represents the number of instances of item i to include in the knapsack. The main objection of this solution is to maximize the profit by inserting the objects into the knapsack without exceeding the capacity of the bag.

The bounded knapsack problem (BKP) removes the restriction that there is only one of each item, but restricts the number x_i of copies of each kind of item to a maximum non-negative integer value c:

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } 0 \leq x_i \leq c$$

The unbounded knapsack problem (UKP) places no upper bound on the number of copies of each kind of item and can be formulated as above except for that the only restriction on x_i is that it is a non-negative integer. [3]

$$\text{maximize} \sum_{i=1}^{n} v_i x_i$$

$$\text{subject to} \sum_{i=1}^{n} w_i x_i \leq W \text{ and } x_i \geq 0$$

C. *Branch and Bound Algorithm*

Branch and bound is an algorithm design paradigm which is generally used for solving combinatorial optimization problems. These kinds of problems typically has exponential time complexity which may also require exploring all possible permutations in worst case. Branch and Bound algorithm can solve these problems relatively very quickly.

The following is the skeleton of a generic branch and bound algorithm for minimizing an arbitrary objective function *f*. [4] In order to obtain an actual algorithm from this, one requires a bounding function (*g* for example), that computes lower bounds of *f* on nodes of the search tree, as well as a problem-specific branching rule.

1) Using a heuristic, find a solution x_h to the optimization problem. Store its value, $B = f(x_h)$. (If no heuristic is available, set *B* to infinity.) *B* will denote the best solution found so far, and will be used as an upper bound on candidate solutions.
2) Initialize a queue to hold a partial solution with none of the variables of the problem assigned.
3) Loop until the queue is empty:
    1. Take a node *N* off the queue.
    2. If *N* represents a single candidate solution *x* and $f(x) < B$, then *x* is the best solution so far. Record it and set $B \leftarrow f(x)$.
    3. Else, *branch* on *N* to produce new nodes N_i. For each of these:
        1. If $g(N_i) > B$, do nothing; since the lower bound on this node is greater than the upper bound of the problem, it will never lead to the optimal solution, and can be discarded.
        2. Else, store N_i on the queue.

There are several different queue data structures that can be utilized. A stack represented by a LIFO (Last-In-First-Out) queue will yield a depth-first algorithm. A best-first branch and bound algorithm can be obtained by using a priority queue that sorts nodes on their *g*-value (bound).

The depth-first variant is recommended when no good heuristic is available for producing an initial solution, because it quickly produces full solutions, and therefore upper bounds. [5]

III. Problem Identification

A. *Case Study*

The cost of commodities varies each year, some experiences an extreme escalation, and some declines. Business owners have to put more concerns towards deciding which goods they need to buy.

TABLE I. Costs of Common Ingredients Provided by Warteg

No.	Ingredient	Price (in Rupiah)		
		2015	2016	2017
1.	Rice	8200/kg	9500/kg	10000/kg
2.	Chicken	29000/kg	33900/kg	34000/kg
3.	Soy	7000/kg	10000/kg	11000/kg
4.	Salt	600/kg	750/kg	1200/kg
5.	Egg	22000/kg	22000/kg	20333/kg
6.	Flour	7500/kg	7500/kg	7500/kg

7.	Cooking Oil	11000/ltr	11700/ltr	13800/ltr
8.	Shallot (*Bawang Putih*)	23000/kg	28000/kg	22000/kg

Data acquired from:

[a.] https://ews.kemendag.go.id/

[b.] http://kabarsembako.blogspot.co.id/2014/12/daftar-harga-sembako-akhir-tahun.html

[c.] http://www.kemenperin.go.id/artikel/4045/Harga-Garam-Anjlok

The above table is a compilation from various sources regarding the average price of commodities in Indonesia throughout the year 2015, 2016, and 2017.

The ingredients above are the most commonly found raw materials which almost every *warteg* needs to have in stock. It is rather difficult to calculate how much profit each raw material will give out. Rather than calculating the cost needed to cook each product, it would be easier to count the ingredients used in one session and calculate the price. However, this kind of lazy calculation would be inaccurate. To make it much more precise, the cost of each ingredient used in cooking up one type of food should be weighed in detail.

For example, the ingredients needed for cooking 40 servings of corn fritter (*bakwan jagung*) are:

- 3 ears fresh corn
- 150 gram all purpose flour
- 75 gram rice flour
- 2 stalks celery, thinly sliced
- 2 stalks scallion, thinly sliced
- 3 eggs
- 10 shallots, minced
- 5 cloves garlic, minced
- 1½ teaspoon salt
- 1 teaspoon ground pepper
- 300 ml water
- enough oil for deep frying[6]

The estimated total price on 2017 is:

```
(0.189kg * Rp. 4.500 = Rp. 850.5) +
(0.15kg * Rp. 7.500 = Rp. 1.125) +
(0.075kg * Rp. 14.000 = Rp. 1.050) +
Rp. 150 + Rp. 200 +
(0.21 * Rp. 20.333 = Rp. 4.269) +
(0.1 * Rp. 22.000 = Rp. 2.200) +
(0.015 * Rp. 17.000 = Rp. 255) +
(8.535gr * Rp. 1,2 = Rp 10,4) +
Rp 20,7 + Rp. 1.031 + Rp. 13.800 =
Rp. 21.758,80 for 40 servings.
```

For each piece of corn fritter, it needs approximately (Rp. 21.758.80 / 40 servings) = Rp. 543.97 worth of ingredients. A typical warteg sells corn fritter for Rp. 2.000 per serving. The profit for one piece of corn fritter is (Rp. 2.000 – Rp. 543.97) Rp. 1.456.03.

B. *Comparison of Approaches*

The knapsack problem can be solved by different kinds of algorithm. Not all algorithm can reach the optimal solution.

1) Greedy Algorithm

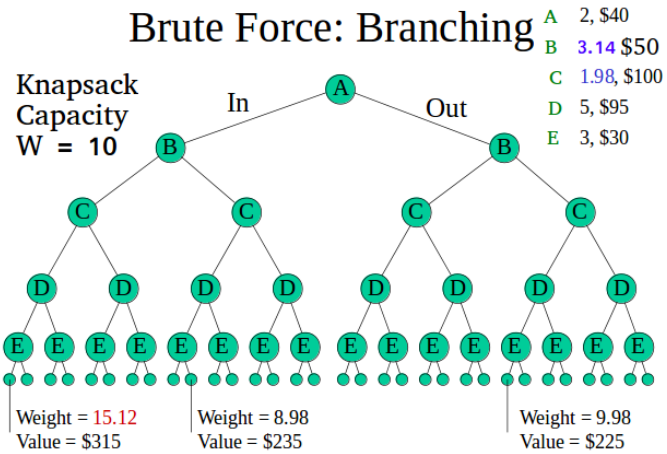
The Greedy approach is to pick the items starting from the biggest value per unit weight (from the maximum individual profit), decreasing until the items are all fit inside the rucksack or the weight capacity has been exceeded. This type of algorithm only works optimally for fractional knapsack problem and may not produce correct result for 0/1 (integer) knapsack.

2) Dynamic Programming

We can also use Dynamic Programming (DP) for 0/1 Knapsack problem. This method requires a 2D (2 dimensional) table of size *n x W*. The DP solution doesn't work if item weights are not integers.

3) Brute Force

Since DP solution doesn't always work, another feasible solution is to use Brute Force. If there exists *n* items, there are 2^n solutions to be generated. The algorithm will check each one to see if they satisfy the constraint, save maximum solution that satisfies constraint. This solution can be expressed as tree. This is a guaranteed optimal solution, but would require exponentially longer time as the number of items increases.



holy

Figure 4. The Brute Force Approach
Source:
http://www.cse.msu.edu/~torng/Classes/Archives/cse830.03fall/Lectures/Lecture11.ppt

4) Backtrack

Another alternative is using backtracking to optimize the Brute Force solution. If it is represented by a tree, we can do DFS (Depth First Search). If we reach a point where a solution no longer is feasible, we stop exploring. In the given example, backtracking would be much more effective if we had even more items or a smaller knapsack capacity.

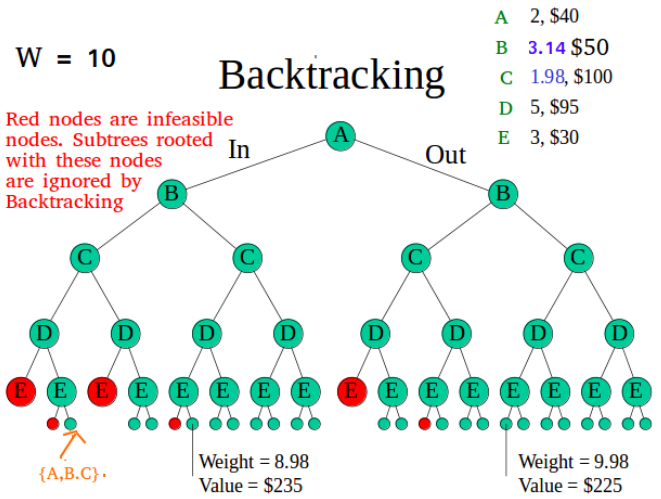


Figure 5. The Backtrack Approach
Source:
http://www.cse.msu.edu/~torng/Classes/Archives/cse830.03fall/Lectures/Lecture11.ppt

5) Branch and Bound

The backtracking based solution works better than brute force by ignoring infeasible solutions. We can do better (than backtracking) if we know a bound on best possible solution subtree rooted with every node. If the best in subtree is worse than current best, we can simply ignore this node and its subtrees. So we compute bound (best solution) for every node and compare the bound with current best solution before exploring the node. [7]

Example bounds used in below diagram are, A down can give $315, B down can $275, C down can $225, D down can $125 and E down can $30. In the next article, we have discussed the process to get these bounds.

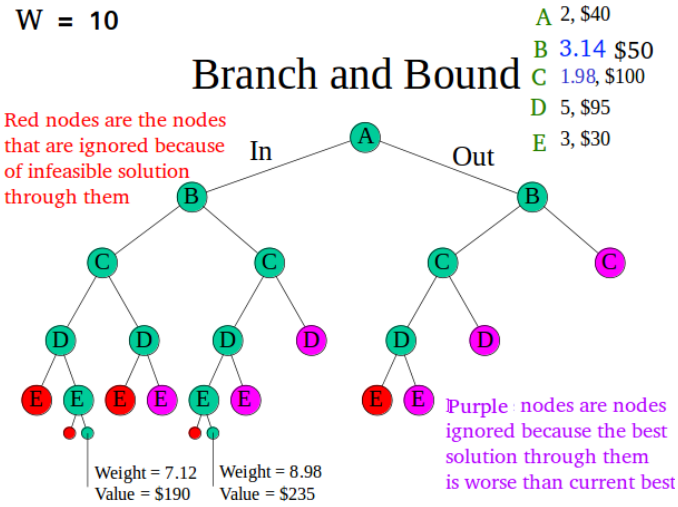


Figure 6. The Branch and Bound Approach
http://www.cse.msu.edu/~torng/Classes/Archives/cse830.03fall/Lectures/Lecture11.ppt

Branch and bound is very useful technique for searching a solution but in worst case, we need to fully calculate the entire tree. At best, we only need to fully calculate one path through the tree and prune the rest of it.

IV. Implementation

A. Code Overview

The implementation of this projet is done in C++ by the author. The interface is made in Indonesian language because majority of warteg owners can speak only either Bahasa Indonesia or their own respective regional language.

The application will ask the user for input of the current buying and selling price of each item, which is then saved into a statically allocated array of items.

The bound function of this application is calculated mainly to find the upper bound on maximum profit. The function returns bound of profit in subtree rooted with root node. As the level gets deeper, the total weight and total value is added if the application decides that the weight won't exceed the total capacity. Weight here represents the buying price.

The knapsack algorithm itself uses a queue of nodes that will be traversed. One by one, an item is extracted from the decision tree to be computed, and the maximum profit and current weight will continuously be updated and saved.

The tree is traversed per level. One level represents one item, whether it is wise to be bought, or should just be left behind. If the current cumulated weight is still less than the capacity, and the profit is greater than the previous profit, then the maximum profit will be updated. On each node, the bound will be calculated by the bound function.

B. Execution

The author has written a short program for calculating the maximum profit a warteg can earn when they are benchmarking their ingredients. The bold characters indicate user input. Warteg owners would be required to specify their budget, the number of items they want to calculate, the names of the ingredients, the buying cost of the item, and the selling cost of the item. The execution is as follows.

```
Masukkan budget anda (Rupiah): 5000
Masukkan jumlah jenis bahan: 10
Bahan ke-1: nasi
Masukkan harga beli dan harga jual: 1000 3000
Bahan ke-2: bakwan
Masukkan harga beli dan harga jual: 500 2000
Bahan ke-3: perkedel
Masukkan harga beli dan harga jual: 850 1500
Bahan ke-4: sayuran
Masukkan harga beli dan harga jual: 500 1000
Bahan ke-5: jamur
Masukkan harga beli dan harga jual: 1000 2500
Bahan ke-6: tempe
Masukkan harga beli dan harga jual: 1500 2500
Bahan ke-7: tahu
Masukkan harga beli dan harga jual: 1000 2300
Bahan ke-8: sambal
Masukkan harga beli dan harga jual: 200 700
Bahan ke-9: kentang
Masukkan harga beli dan harga jual: 900 1700
Bahan ke-10: telur
Masukkan harga beli dan harga jual: 1000 3500

Bahan yang disarankan:
```

```
Bahan ke-1: nasi,
harga beli: 1000, harga jual: 3000
Bahan ke-2: bakwan,
harga beli: 500, harga jual: 2000
Bahan ke-4: sayuran,
harga beli: 500, harga jual: 1000
Bahan ke-5: jamur,
harga beli: 1000, harga jual: 2500
Bahan ke-7: tahu,
harga beli: 1000, harga jual: 2300
Bahan ke-8: sambal,
harga beli: 200, harga jual: 700
Bahan ke-10: telur,
harga beli: 1000, harga jual: 3500
Keuntungan bersih terbesar: 9800
```

C. Result Analysis

Based on the results from the application itself, the best solution is to choose 7 (seven) ingredients as shown on the execution part of this paper. The maximum profit gained is Rp. 9.800, which is obtained after subtracting the total cost from total revenue.

1) This result cannot be easily predicted without the help of detailed calculation. Mere eyes and basic computing skills would not be able to identify which ingredients should be picked to gain the most profit. If one blindly subtracts the cost from the revenue of one product, he would not be aware of how it affects the next potential ingredients that is most profitable.

2) Even though it calculates the maximum profit, it does not ensure the nutrition balance in the menu. This application is limited to satisfy the economic view of a warteg, without giving any constraints towards which type of nutrition is an absolute necessity for the consumers' daily needs.

3) The Branch and Bound Algorithm will still have exponential complexity in the worst case scenario. It will generate all intermediate stages and all leaves at worst, and will have $2^{n-1} - 1$ nodes.

4) However, the Branch and Bound approach is still better than the brute force algorithm. On average cases, it will not generate all possible nodes/solution. The required memory depends on the length of the priority queue.

ACKNOWLEDGMENT

First and foremost, the author want to offer this endeavor to our God almighty for the wisdom He bestowed upon her, the motivation, strength, and good health needed to finish this research.

The author would also like to express her sincerest sense of gratitude and admiration towards all the lecturers of IF2211 Strategi Algoritma, namely Drs. Nur Ulfa Maulidevi, Dr. Ir. Rinaldi Munir, M.T., and Dr. Masayu Leylia Khodra ST., MT. for all the guidance and support they have continuosly delivered to their students whole-heartedly. They have enlightened and broadened the knowledge of their students in a way that cannot be found anywhere else.

Another heap of thanks to Warteg Hipster, Jl. Bahureksa No.5, Citarum, Bandung Wetan, Kota Bandung, Jawa Barat 40115, Indonesia for serving the author not only with delicious, affordable dishes that tasted like home, but also their very own personal experience in developing a Warteg, even in total darkness due to the regional blackout.

Lastly, the author would like to give a cordial, virtual hug to all of her family and friends for their endless moral support to finish this project.

REFERENCES

[1] https://www.goodindonesianfood.com/en/the-origin-of-warung-tegal/ retrieved on 16 May 2017.

[2] Dantzig, Tobias. 1930. *Numbers: The Language of Science*. Macmiillan.

[3] Mathews, G. B. 1897. *On the Partition of Numbers: Proceedings of the London Mathematical Society*, p. 486–490. Unpublished.

[4] Clausen, Jens. 1999. *Branch and Bound Algorithms – Principles and Examples.* Copenhagen: University of Copenhagen.

[5] Mehlhorn, Kurt and Sanders, Peter. 2008. *Algorithms and Data Structures: The Basic Toolbox*, p. 249. Karlsruhe: Springer.

[6] http://dailycookingquest.com/by-category/side-dish/bakwan-jagung retrieved on 17 May 2017.

[7] http://www.geeksforgeeks.org/branch-and-bound-set-1-introduction-with-01-knapsack/ retrieved on 17 May 2017.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Mei 2017

Wenny Yustalim / 13515002