Billy Bonka’s Chocolate Factory  
Szymon Janusz 20792986

In the context of Billy Bonka’s Chocolate Factory, an optimal solution is a gift box with the highest retail value whilst meeting the defined constraints.

To achieve this optimal solution, the program calculates all possible permutations of all items (chocolates). The gift box that meets the constraints is then added to a list, which can be iterated through or sorted, to find the gift box with the highest retail value. This brute-force approach is guaranteed to find the optimal solution at the cost of resources.

The program created for Billy Bonka’s Chocolate Factory has an asymptotic efficiency of O(2^n). This results in an exponential growth of possible permutation with each possible item (chocolate) added. The explanation for this is that with each new item added, the bit vector that represents all possible permutations gains an additional bit, e.g. with 5 items, the bit vector is “11111”, however with 6 items, it results in the bit vector “111111”. As this is a binary representation of the solution space, with each additional bit added, the total number that can be represented is doubled.

The limitations of the provided solution are memory and CPU processing time. Dealing with an exponential growth of permutations results in the brute-force approach not being feasible for large-scale calculations with more items present. The asymptotic complexity suggests that it would use up too many resources (RAM and CPU processing time) to provide a solution to a larger problem.

To overcome this, an alternative approach would be the “branch and bound” algorithm. The algorithm looks at promising solutions only, that have a chance to outperform the current best and would allow the program to quickly disregard any permutations that are guaranteed to be worse than the current best solution, thus resulting in a reduction of resources and processing time. The items are enumerated in descending order of their profit, like the greedy approach. The algorithm could use a best first search, where the node which has the best bound is traversed into next.

Finally, to maximise profit of the gift box, one would simply have to change the linear search (O(n)) function present, “GetOptimalSolution(ref List<GiftBox> GiftBoxes)” to search for the gift box with the highest profit instead of the retail value. Doing so will not impact the efficiency of the program at all and is more efficient than sorting (O(nlogn)) the list in descending order and returning the first element.