William Sutanto
wsutanto@csu.fullerton.edu
Mariah Harris
mharris2021@csu.fullerton.edu

Project 2 CPSC 335: Greedy Versus Exhaustive Search

1. Analyze your greedy algorithm code mathematically to determine its big-O efficiency class, probably $O(n^2)$ or $O(n \log n)$.

```
greedy_max_weight(C, food_items):
     todo = food_items         //1
     result = empty vector          //1
     result_weight = 0         //1
     while todo is not empty: //n times
          Find the food item "a" in todo of maximum weight per
          its calorie    //n times
          Let q be a's calorie in ounces      //1
          if (result_weight + q) <= W:  //2
               result.add_back(a)  //1
               result_weight += q  //1
          Remove "a" from todo                //1
     return result
```

Step count = 3 + n*(n+6)= 3 + n^2 + 6n
From the step count, we can conclude that the greedy algorithm has the time complexity of big-O efficiency class of O(n^2).

2. Analyze your exhaustive optimization algorithm code mathematically to determine its big-O efficiency class, probably $O(2^n \cdot n)$.

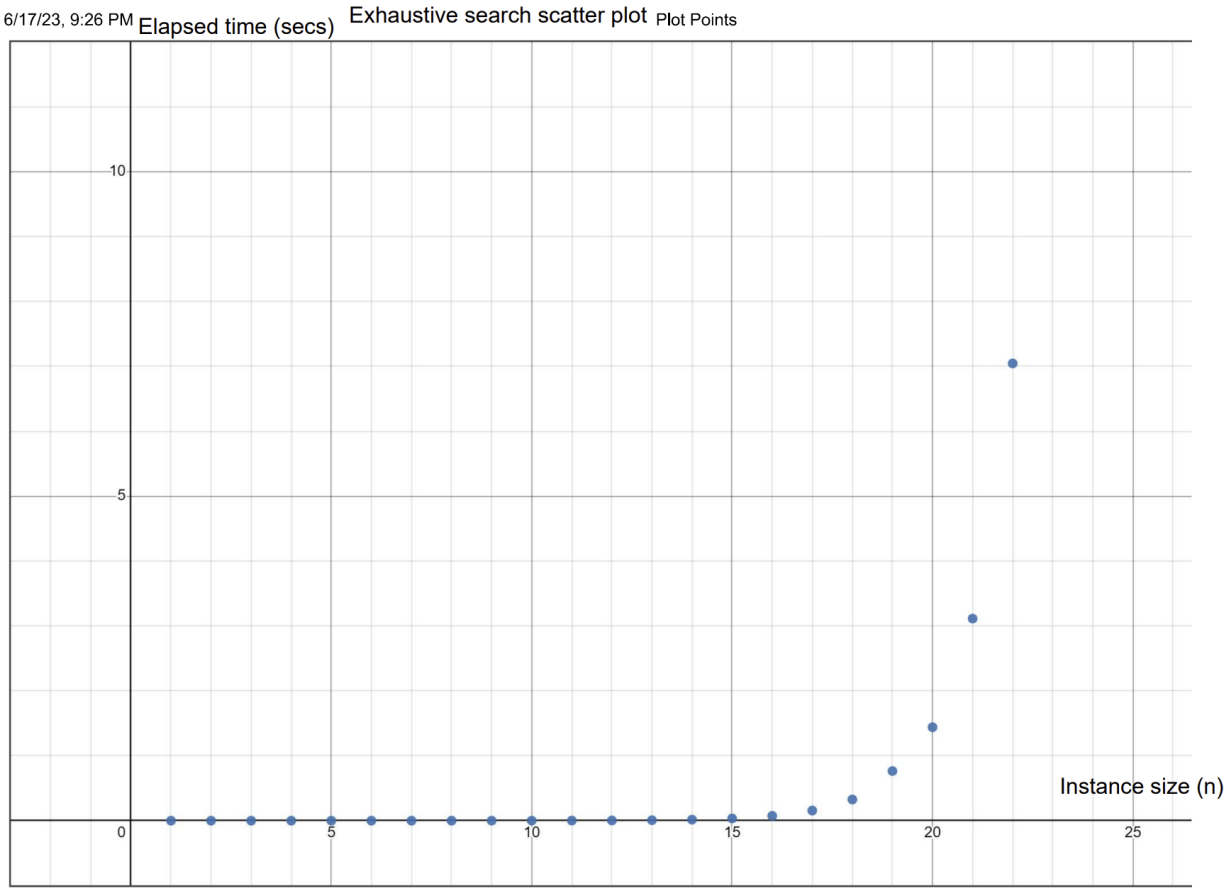
```
exhaustive_max_weight(C, food_items):
     n = |food_items|          //1
     best = None          //1
     best_weight = 0           //1
     for subset from 0 to (2^n -1): //2^n times
          current_subset = empty vector //1
          subset_weight = 0             //1
          subset_calorie = 0            //1
          for j from 0 to n-1:               //n times
```

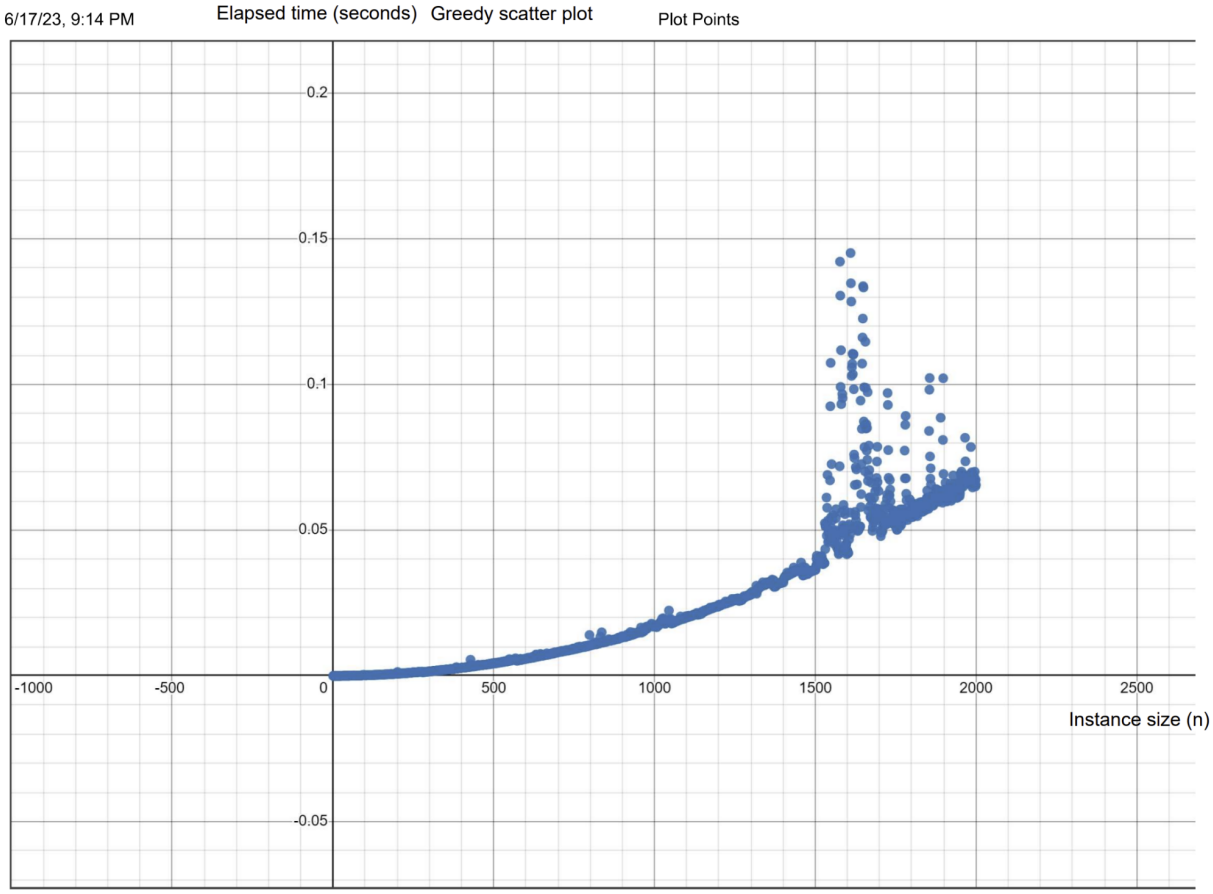
```
            if (subset & (1 << j):   //2
                 subset_weight += food_items[j]->weight  //1
                 subset_calorie += food_items[j]->calorie//1
                 current_subset.add_back(food_items[j]) //1

      if subset_calorie <= C:  //1
           if best is None or  //1
              subset_weight > best_weight:         //1
                best_weight = subset_weight        //1
                best = current_subset              //1
return best
```

Step count = $3 + 2^n(8+n(5)) = 3 + 8(2^n) + (2^n)(5n)$

From the step count, we can conclude that the exhaustive optimization has the time complexity of big-O efficiency class of $O(2^n * n)$.

SCATTER PLOTS





a. **Is there a noticeable difference in the performance of the two algorithms? Which is faster, and by how much? Does this surprise you?** The difference in the performance of the two algorithms is significant. The greedy algorithm is much quicker than the exhaustive optimization. This is not surprising, as the greedy algorithm approach involves making locally optimal choices at each step, with the expectation that this will ultimately lead to a globally optimal solution. In contrast, an exhaustive search considers all possible solutions and then selects the most optimal one.

b. **Are your empirical analyses consistent with your mathematical analyses?** The mathematical analysis of the greedy algorithm in this experiment has a time complexity of O(n^2), indicating its efficiency. The scatterplot for the greedy algorithm shown above confirms this analysis, as it exhibits the characteristics of an exponential graph with continuous and curved growth. The mathematical analysis of the exhaustive optimization algorithm in this experiment has a time complexity of O(2^n * n), indicating its efficiency. The scatterplot for the exhaustive optimization algorithm shown above confirms this analysis, as it exhibits

exponential growth (2^n) and some linear growth (*n) due to the presence of both terms in the time complexity expression. The exhaustive optimization has the time complexity of big-O efficiency class of O(2^n * n). Therefore, the empirical analysis for both the greedy algorithm and exhaustive optimization algorithm align with their respective mathematical analysis.

Hypothesis 1: Exhaustive search algorithms are feasible to implement, and produce correct outputs.
Hypothesis 2: Algorithms with exponential running times are extremely slow, probably too slow to be of practical use.

c. Is this evidence consistent or inconsistent with hypothesis 1? Justify your answer.
Exhaustive search algorithms systematically explore all possible solutions within a given search space, ensuring that no potential solution is missed. Through thorough examination of each available option, exhaustive search algorithms guarantee correct output. They are feasible for smaller search spaces. However, feasibility diminishes for larger input sizes because the search space will grow exponentially and therefore require a large amount of resources and time.

d. Is this evidence consistent or inconsistent with hypothesis 2?
Consistent: In this experiment, the scatter plot showcases the runtime of the exhaustive optimization algorithm. It is evident that for each additional instance exceeding 20 instances, the runtime increases by approximately 5 seconds. When dealing with large-scale datasets in practical scenarios, exponential algorithms prove to be inefficient and unsuitable. Instead, more efficient algorithms with polynomial or sub-exponential time complexities are favored to handle larger input sizes that also offer reasonable execution times.