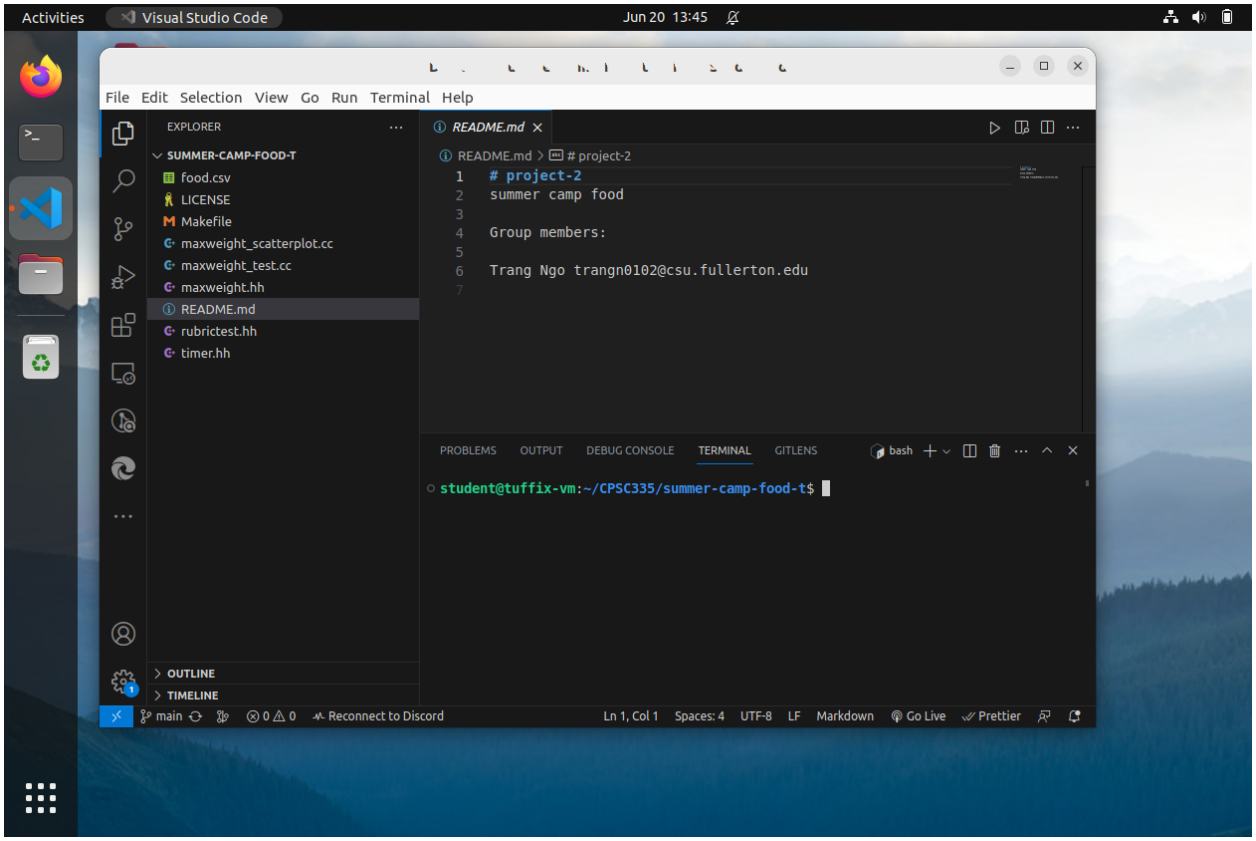
CPSC 335 Project 2 Report

1. Name, CSUF Email, Indication for submission

Trang Ngo trangn0102@csu.fullerton.edu
This is my submission for Project 2: Greedy versus Exhaustive

Tuffix Screenshot:



2. Mathematically Analysis

a. Greedy Pattern

```
auto todo = foods; //1
std::unique_ptr<FoodVector> result (new FoodVector); //1
double total = 0; //1
auto result_weight = todo.begin(); //1
while (!todo.empty()) { //n times
        double best_food = 0; //1
        for (auto i = todo.begin(); i != todo.end(); i++) { //n times
                auto temp_weight = i->get()->weight()/i->get()->calorie(); //2
                if(temp_weight >= best_food) { //1
                        result_weight = i; //1
```

```
                    best_food = temp_weight; //1
                }
            }
            if ((total + result_weight->get()->calorie()) <= total_calorie) { //2
                result->push_back(*result_weight); //1
                total += result_weight->get()->calorie(); //1
            }
            todo.erase(result_weight); //1
        }
        return result; //0
```

Step Count:

$$SC = 1+1+1+1+n*\{1+[(n)*[2+[1+\max(2,0)]]] + (2+\max(2,0))+1\}$$

$$= 4+n*\{1+[n*5]+4+1\}$$

$$= 4+n*\{[n*5]+6\}$$

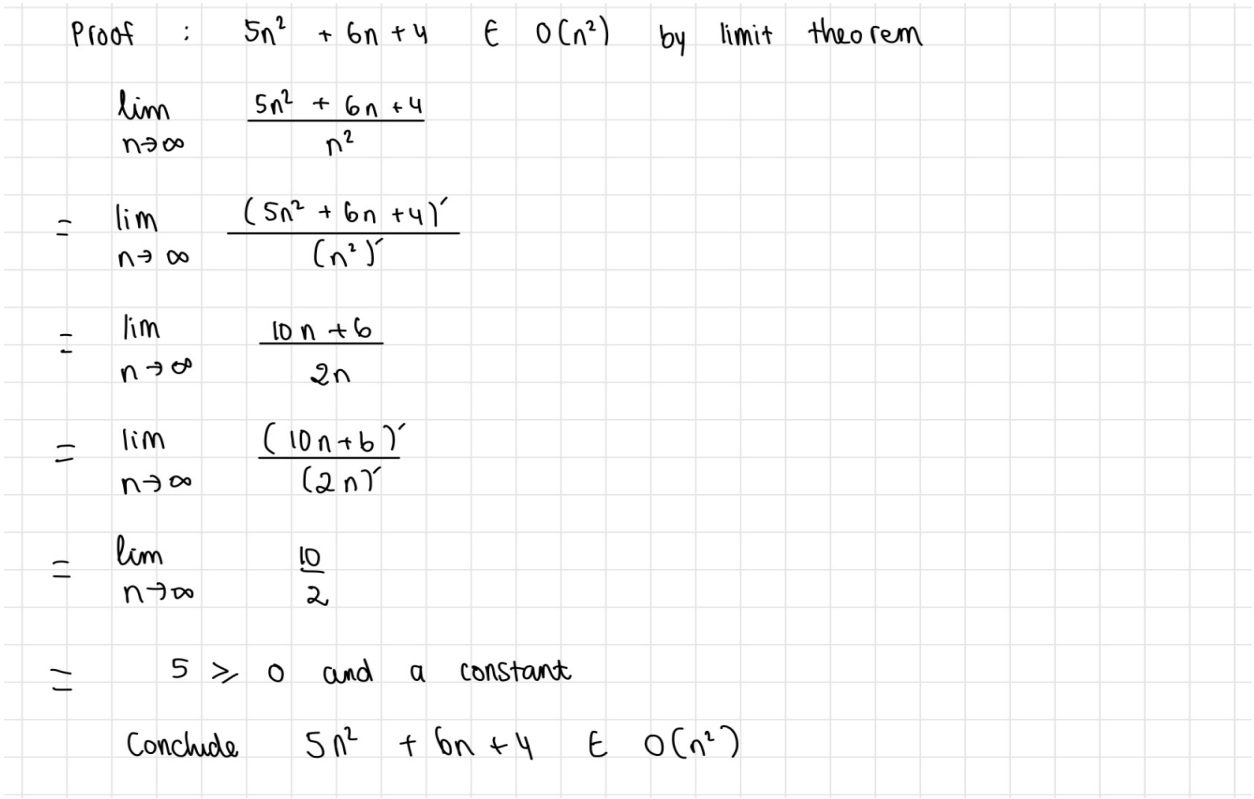
$$= 4+n*\{5n+6\}$$

$$= 4+5n^2+6n$$

$$= 5n^2+6n+4$$

Time Complexity $= O(n^2)$

Proof:



b. Proper Exhaustive Search

```
std::unique_ptr<FoodVector> best(new FoodVector); //1
int n = foods.size();  //1
double best_weight = 0.0; //1
double best_calories = 0.0; //1
for (uint64_t i = 0; i <= std::pow(2, n)-1; i++) { //((2^n-1)-0)+1 = 2^n times
        std::unique_ptr<FoodVector> candidate(new FoodVector); //1
        double candidate_weight = 0.0; //1
        double candidate_calories = 0.0; //1
        for (int j = 0; j <= n-1; j++){ //n-1+0+1 = n times
                if (((i >> j) & 1) == 1) { //3
                        candidate->push_back(foods[j]); //1
                }
        }
        sum_food_vector(*best, best_calories, best_weight); //1
        sum_food_vector(*candidate, candidate_calories, candidate_weight); //1
        if (candidate_calories <= total_calorie) { //1
                if (best->empty() || candidate_weight > best_weight) { //2
                        best = std::move(candidate); //1
                }
```

```
            }
        }
        return best; //0
```

Step Count:

$$SC = 1+1+1+1+1+(2^n)*[1+1+1+(n)*(3+\max(1,0))+1+1+1+\max([2+\max(1,0)],0)]$$

$$= 4 + 2^n [3 + [n*4] + 3 + \max(3,0)]$$

$$= 4 + 2^n [3 + 4n + 6]$$

$$= 4 + 2^n [4n + 9]$$

$$= 4 + 2^n(4n) + 2^n(9)$$

$$= 2^n(4n) + 2^n(9) + 4$$

Time complexity : $O(2^n * n)$

Proof:

Proof : $2^n(4n) + 2^n(9) + 4 \in O(2^n * n)$ by limit theorem

$$\lim_{n\to\infty} \frac{2^n(4n) + 2^n(9) + 4}{(2^n * n)}$$

$$= \lim_{n\to\infty} \frac{(2^n(4n) + 2^n(9))}{(2^n * n)} + \lim_{n\to\infty} \frac{4}{2^n * n}$$

Common factor

$$= \lim_{n\to\infty} \frac{(2^n * (4n + 9))}{2^n * n} + \lim_{n\to\infty} \frac{4}{2^n * n}$$

$$= \lim_{n\to\infty} \frac{(4n+9)}{n} + \lim_{n\to\infty} \frac{4}{2^n * n}$$

$$= \lim_{n\to\infty} \frac{(4n+9)'}{(n)'} + \lim_{n\to\infty} \frac{(4)'}{(2^n * n)'}$$

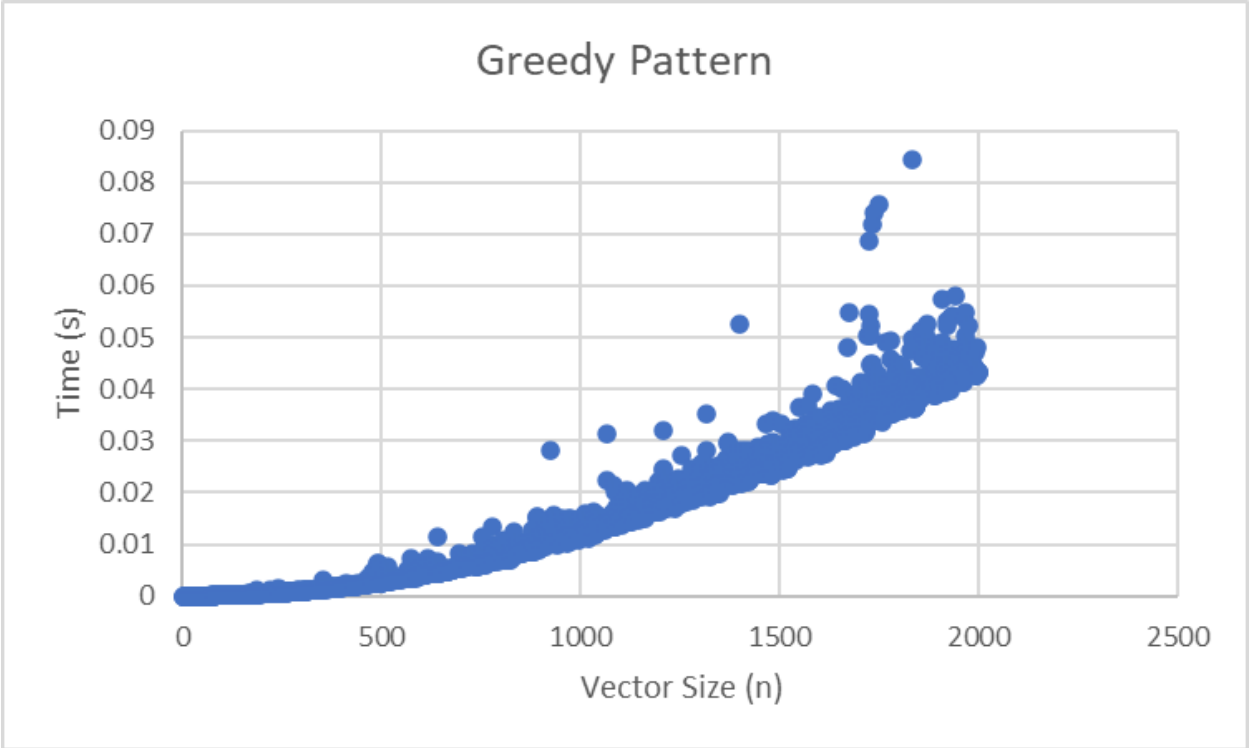
$$= \lim_{n\to\infty} \frac{4}{1} + 0$$

$= 4 \geq 0$ and a constant

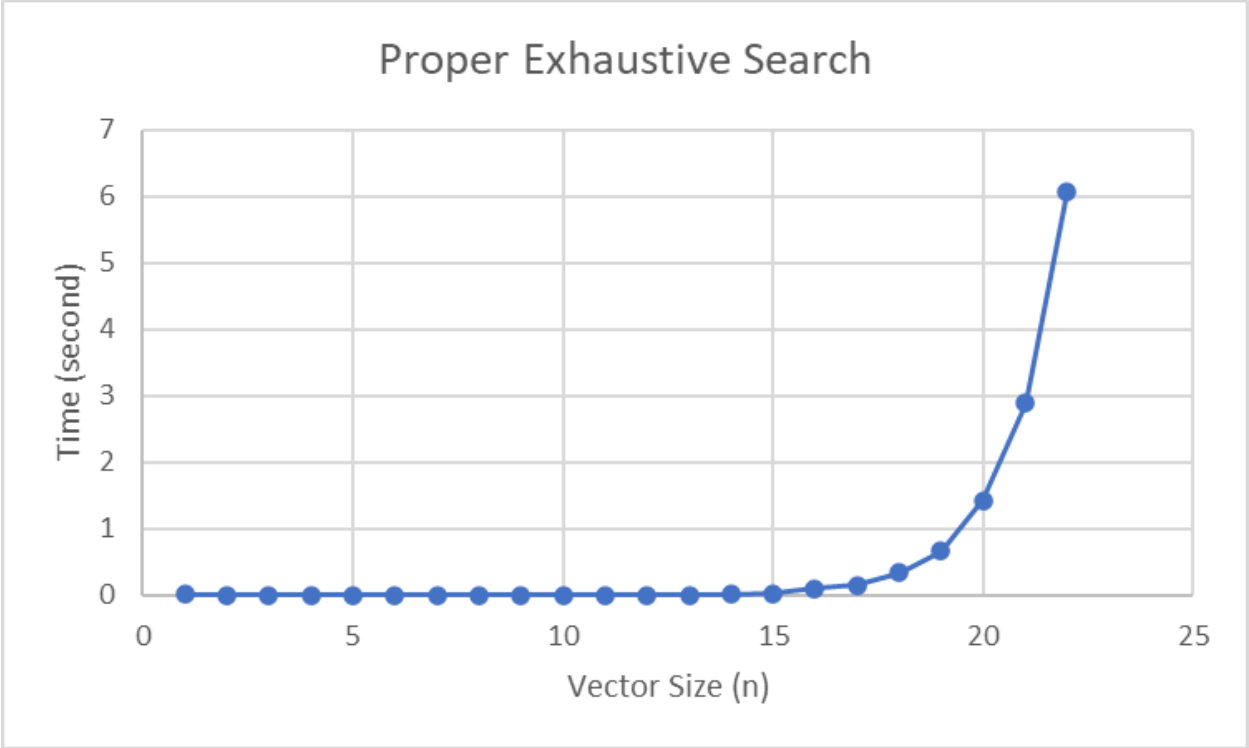
Conclude $2^n(4n) + 2^n(9) + 4 \in O(2^n * n)$

3. Two scatter plots meeting the requirements stated above.

a. Greedy Pattern



b. Proper Exhaustive Search



4. Answers to the following questions, using complete sentences.

a. Is there a noticeable difference in the performance of the two algorithms? Which is faster, and by how much? Does this surprise you?

There is a noticeable difference in the performance of the two algorithms. The greedy algorithm is much faster than the proper exhaustive algorithm. Based on the graph of the exhaustive algorithm, when the vector size(n) is 20, the time is approximately 1.5 second, while for the greedy algorithm, it takes very less than 1 second (approximately 0.00000828 second). This does not surprise me because proper exhaustive search would search for all possible results while greedy search for the best result. I also noticed that when compiling the two algorithms, proper exhaustive search takes much longer to process than the greedy algorithm. Additionally, with the time complexity, the exponential (exhaustive) time complexity would be slower than the quadratic (greedy) time complexity.

b. Are your empirical analyses consistent with your mathematical analyses? Justify your answer.

My empirical analyses are consistent with my mathematical analyses. According to the empirical analyses, the scatterplot for greedy algorithms takes faster process time than the scatterplot for proper exhaustive search. When the vector size (n) is 20, the greedy algorithm takes less than 0.1 second while the proper exhaustive algorithm takes about 1.5 second. For the mathematical analyses, the greedy algorithm concluded that the time complexity belongs to $O(n^2)$, on the other hand, the proper exhaustive algorithm concluded the time complexity belongs to $O(2^n *n)$. So the empirical analyses align with the mathematical analyses with the observation of time complexity and execution time.

c. Is this evidence consistent or inconsistent with hypothesis 1? Justify your answer.

The first hypothesis claimed that exhaustive search algorithms are feasible to implement, and produce correct outputs. This evidence is consistent because the algorithm considers all possible results, also with the implementation of subsets, it would ensure an optimal solution.

d. Is this evidence consistent or inconsistent with hypothesis 2? Justify your answer.

The second hypothesis claimed that algorithms with exponential running times are extremely slow, probably too slow to be of practical use. This evidence is consistent because based on the empirical analyses, when the vector size(n) of exhaustive search is 20, the algorithms take around 1.5 second to process. The processing time would grow exponentially ($O(2^n *n)$) as the size increases.