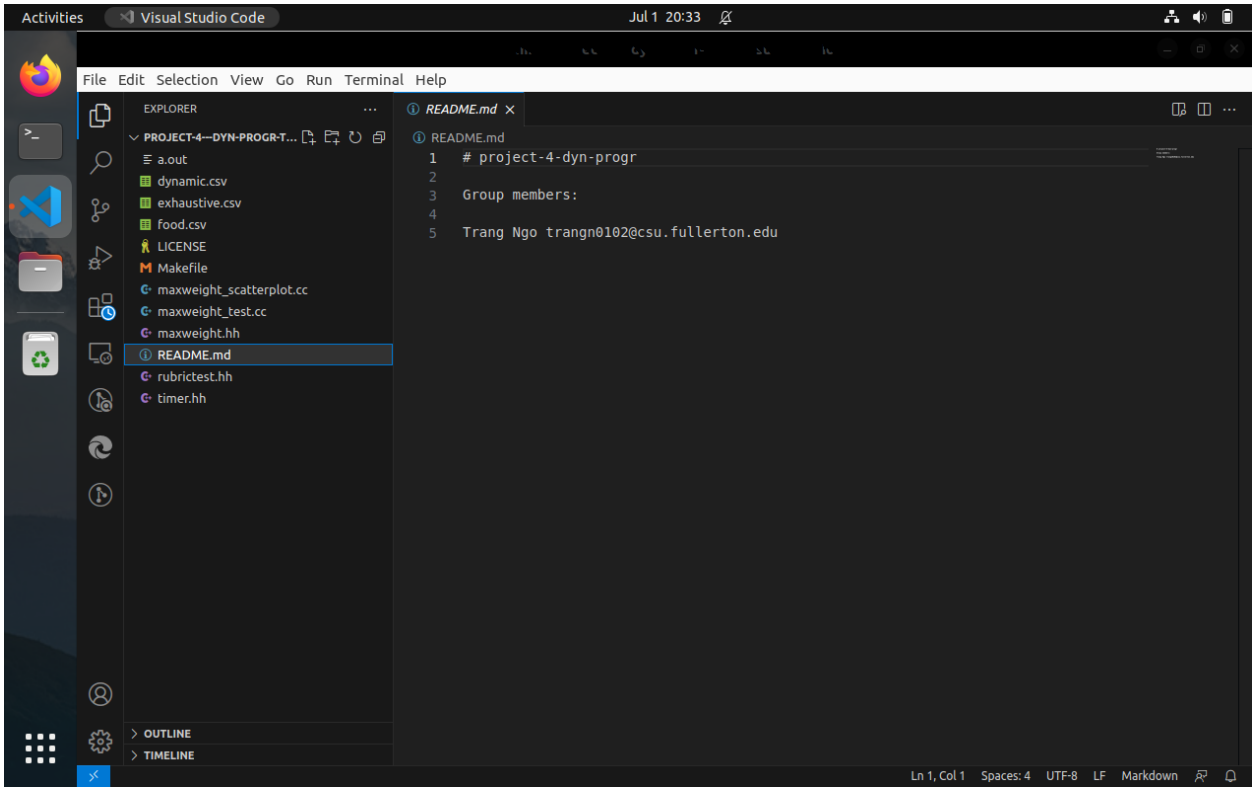
CPSC 335 Project 4 Report

1. Name, CSUF Email, Indication for submission

Trang Ngo trangn0102@csu.fullerton.edu
This is my submission for Project 4: Dynamic vs. Exhaustive

Tuffix Screenshot:



2. Mathematically Analysis

a. Dynamic

```
std::unique_ptr<FoodVector> dynamic_max_weight
(
        const FoodVector& foods,
        double total_calories
)
{
std::vector<std::vector<double>> T((foods.size()+1), std::vector<double> (total_calories+1, 0)); //
2
std::unique_ptr<FoodVector> source(new FoodVector(foods)); //1
std::unique_ptr<FoodVector> best(new FoodVector); //1
for (size_t i = 1; i <= foods.size(); i++ ) { //n times
```

```
        for (size_t j = 1; j <= total_calories; j++){ //n times
                if (i == 0 || j == 0) { //3
                        T[i][j] = 0; //1
                }
                else if (j - foods[i-1]->calorie() < 0){ //3
                        T[i][j] = T[i-1][j]; //2
                } else {
                        T[i][j] = std::max(T[i-1][j], (T[i-1][j-foods[i-1]->calorie()] +
foods[i-1]->weight())); //6
                }
        }
}
size_t i = foods.size(); //1
size_t c = total_calories; //1
while (i != 0 && c != 0) { // n + 2 times
        if (T[i][c] != T[i - 1][c]) { //2
                best->push_back(foods[i - 1]); //1
                c = c - foods[i - 1]->calorie(); //3
        }
        i--; //1
   }
   return best; //0
}
```

Step Count:

$SC = 2+1+1+n*n[3+\max(1,3+\max(2,6)]+1+1+(n+2)*(2+\max(4,0))+1$

$= 4 + n*n[3+\max(1,9)]+2+(n+2)*(6)+1$

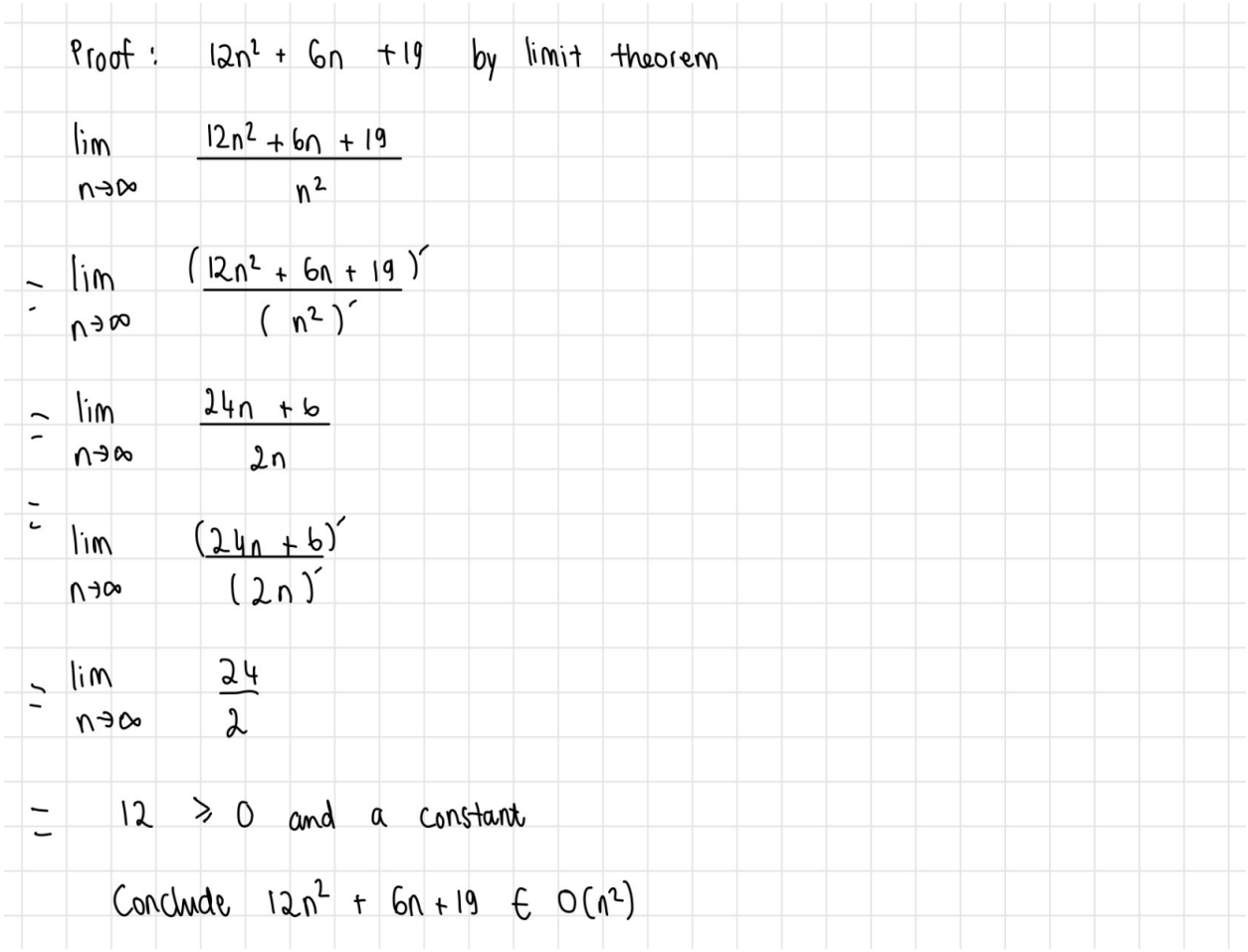
$= 4 + n^2[12] + 2 + 6n + 12 + 1$

$= 4 + 12n^2 + 15 + 6n$

$= 12n^2 + 6n + 19$

Time complexity : $O(n^2)$

Proof:



b. Exhaustive

```
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:

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

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

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

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

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

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

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

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

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

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

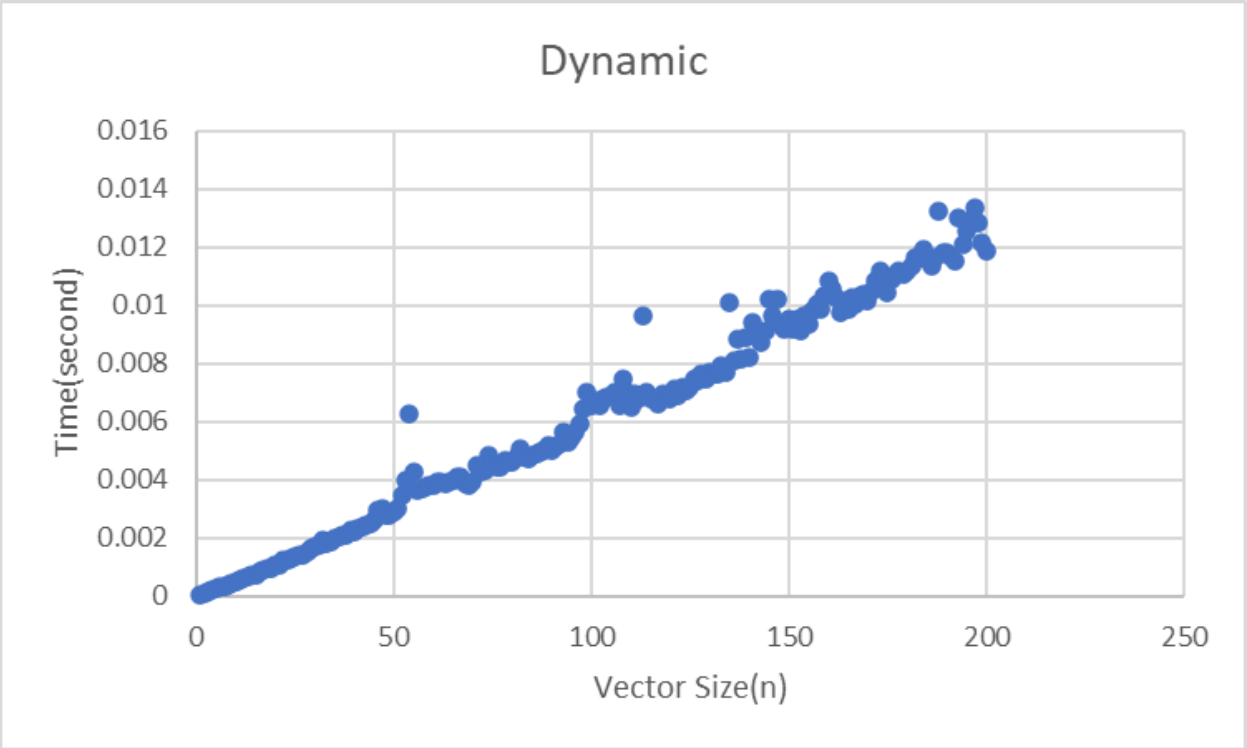
$$= \lim_{n\to\infty} \frac{4}{1} + 0$$

$= 4 \geq 0$ and a constant

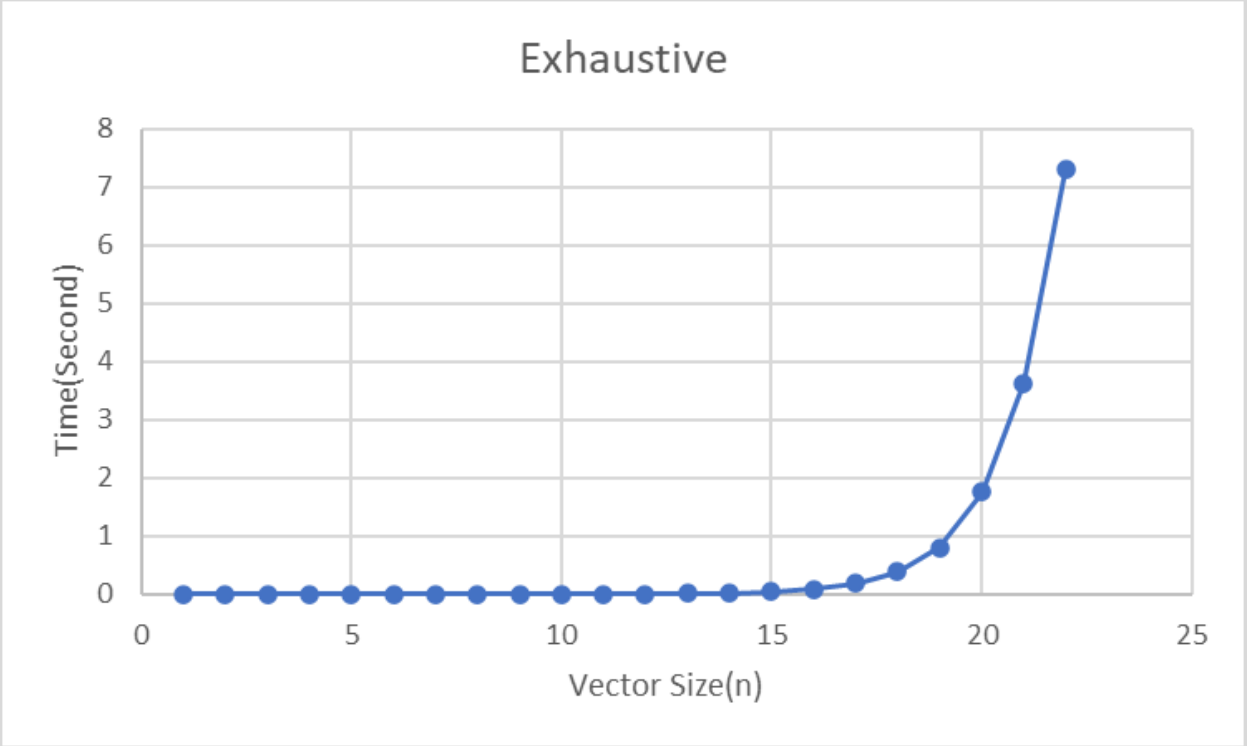
Conclude $2^n(4n) + 2^n(9) + 4 \in O(2^n * n)$

3. **Two scatter plots meeting the requirements stated above.**

a. Dynamic



b. Exhaustive



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 dynamic algorithm is much faster than the proper exhaustive algorithm. Based on the graph of the dynamic algorithm, when the vector size (n) is 20, the time is approximately less than 0 second, while for the exhaustive algorithm, it takes almost 2 seconds. This does not surprise me because the time complexity, the exhaustive time complexity is $O(2^n * n)$ would be slower than the dynamic time complexity is $O(n^2)$.

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

Yes, my empirical analyses are consistent with my mathematical analyses. According to the empirical analyses, the scatterplot for dynamic algorithms takes faster process time than the scatterplot for proper exhaustive search. When the vector size (n) is 20, the dynamic algorithm takes less than 0.1 second while the proper exhaustive algorithm takes about 2 seconds. For the mathematical analyses, the dynamic algorithm has the step count and the proof concluded that the time complexity belongs to $O(n^2)$, on the other hand, the proper exhaustive algorithm has the step count and the proof 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 2 seconds to process. The processing time would grow exponentially ($O(2^n *n)$) as the size increases.