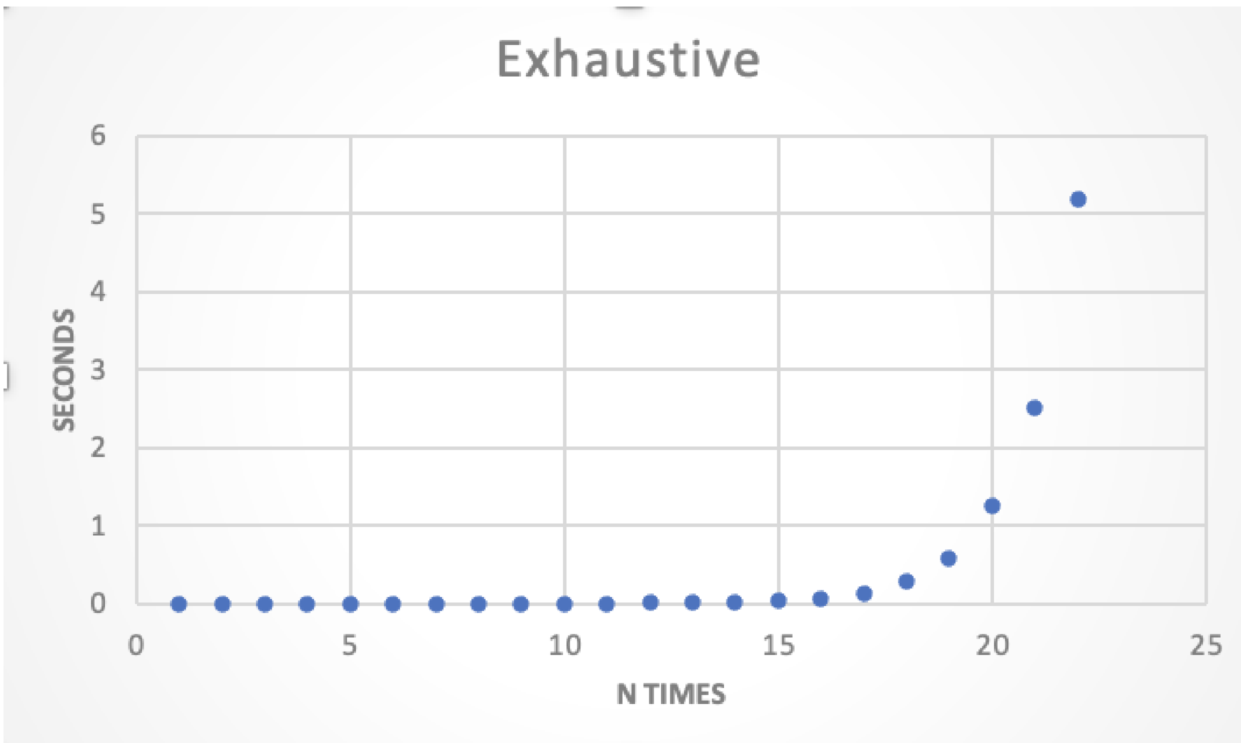
CPSC 335 - Project 2 <u>PDF REPORT</u>

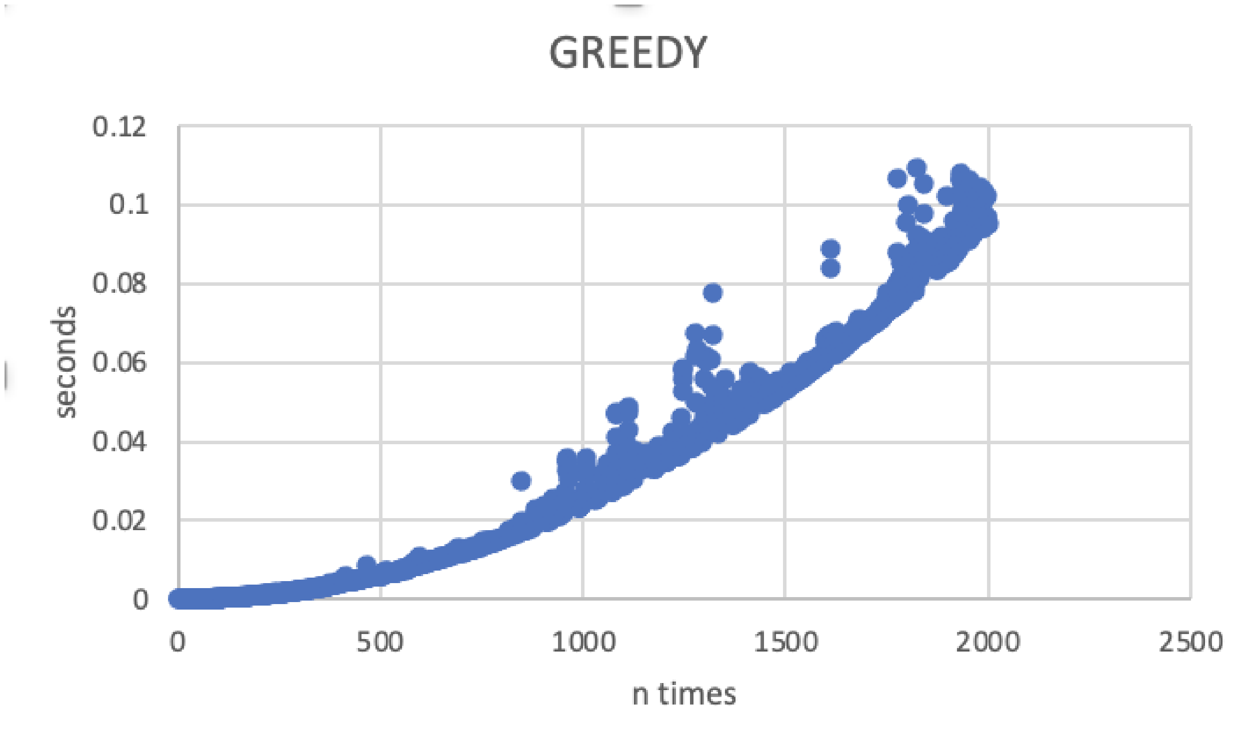
Ali Tahami
atahami3@csu.fullerton.edu

Hamid Suha
hsuha@csu.fullerton.edu

EXHAUSTIVE SCATTER PLOT



GREEDY SCATTER PLOT



GREEDY ALGORITHM STEP COUNT AND PROOF

greedy algorithm step count

```
unique_ptr < cargovector> Todo(new cargovector(goods)); //1
unique_ptr < cargovector> result(new cargovector()); //1

int result_volume=0 //1
Double max=0 //1
int v=0 //1
int index=0 //1

while(!Todo->empty()) //N times
{
   for(int i=0; i< Todo->size(); i++) // n times
   {
      if(max < Todo->at(i)->weight() / Todo->at(i)->volume())
      {
         index=i
         max= Todo->at(i)->weight() / Todo->at(i)->volume()
      }
   }
   v= Todo->at(index)-> Volume() //3
   if(result_Volume+v ≤ Total_Volume)
   {
      result->add_back(Todo->at(index))
      result_volume +=v
   }
   Todo->erase(index) //1
}
return result;
```

$1+1+1+1+1+1=6$

$6+\max(7,0)$

$=\boxed{13}$

$\boxed{13n}$

$2+\max(3,0)$

$\boxed{5}$

$5+3=8$

Proof

$6+n(13n+8)$

$6+13n^2+8n$ (On^2)

$$\lim_{n\to\infty} \frac{13n^2+8n+6}{n^2}$$

$$\lim_{n\to\infty} \frac{26n+8}{2n}$$

$$\frac{26}{2} = 13 \checkmark$$

does belong to $O(n^2)$

EXHAUSTIVE STEP COUNT

exhaustive stepcount

```
for (int j=0; j<n; j++) //n+1
{
    if(((i>>j)&1)==1) 3+max(1,0)=4
    {
        candidate->push_back(goods[j])
    }
}                                                  4n+4
sum_cargo_vector(*candidate, total_volume_candidate, total_weight_candidate) //1
sum_cargo_vector(*best, total_volume_best, total_weight_best) //1
if(total_volume_candidate <= total_volume) // 1+max(5,0) = 6
{
    if(best->size()==0 || total_weight_candidate > total_weight_best) 4+max(1,0)   5
    {
        *best = *candidate //1
    }
}
```

$1+1+6+4n+4 = 4n+12 \quad O(n)$

FILTER CARGO VECTOR STEP COUNT AND PROOF

filter cargovector step count

```
unique_ptr<cargovector> filter(new cargovector) //1
for ( int i=0 i< source() && (*filter).size() < Total; i++) //n times
  {
    if (source[i]->weight()>=min_weight && source[i]->weight()<=max_weight)
      {
        filter->push_back(source[i])
      }
  }
return filter
```

$5+\max(1,0)$

$= \boxed{6}$

$6n+1$

$O(n)$

Proof

$$\lim_{n \to \infty} \frac{6n+1}{n}$$

$= 6$ ✓

does belong to $O(n)$

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 two algorithms, greedy and exhaustive, have a distinct difference. The greedy algorithm is more efficient. The results were unexpected; we did not expect an exhaustive approach to have such a large increase in time complexity. The jump from 20 to 25 resulted in a significant change in time measured in seconds.
b. **Are your empirical analyses consistent with your mathematical analyses? Justify your answer.** Yes, our empirical analysis is consistent with our mathematical analysis. Because we concluded our runtime for the greedy algorithm to be $O(n^2)$ while our exhaustive was $O(2^n \cdot n)$. So exhaustive was much slower. The difference in our empirical analysis was also much slower which makes both algorithms consistent.
c. **Is this evidence consistent or inconsistent with hypothesis 1? Justify your answer.** Yes, it does produce the correct output, however, it is not feasible to implement dues to its exponential increase in runtime.

d. **Is this evidence consistent or inconsistent with hypothesis 2? Justify your answer.** Yes, due to high runtime, it is not feasible to implement especially with a high "n" value. Which makes it far too slow to be of practical use.