CPSC 335 - Algorithm Engineering Project 2: Greedy versus Exhaustive

Malka Ariel Lazerson Fall 2021

Instructor: Doina Bein

Tabl	le	of	Ca	nt	en	ts

Names, CSUF Email, and Intent	2
ReadMe.md Screenshot	3
Code Compilation and Successful Execution	3
Empirical Data	4
Scatter Plots	6
Mathematical Analysis	7
Question Answers	12

Names, CSUF Email, and Intent

Name: Malka Ariel Lazerson

CSUF Email: mlazerson@csu.fullerton.edu

Intent: This document is intended to be one part of a submission for Project 2: Greedy versus Exhaustive. This document contains....

- 1. Names, CSUF-supplied email address(es), and an indication that the submission is for project
- 2.
- 2. Proof of code compilation and successful execution
- 2. Two scatter plots
- 3. Answers to the following questions, using complete sentences.

Questions....

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

Are your empirical analyses consistent with your mathematical analyses? Justify your answer. Is this evidence consistent or inconsistent with hypothesis 1? Justify your answer. Is this evidence consistent or inconsistent with hypothesis 2? Justify your answer.

Code and more files will be placed in the github for the instructor to review.

ReadMe.md Screenshot

```
IN [READHE.md] (md)

Sas-project-2
Container ship weight-maximization
Group members:
Halka A Lazerson mlazersongcsu.fullerton.edu
The Hypothesis

Source
The Problem

Source
Both algorithms will be used to solve an interesting problem. Suppose the following.

We are a trucking company responsible for transporting goods to a seaport, to be loaded in a container ship. Our trucking company has been contracted to transport a list of goods, and we A description of the problem at hand is as follows:

Container ship weight-maximization problem

input: A positive "volume limit" budget V (floating point of TEUs); and a vector G of n "goods" objects, containing one or more goods where each cargo item a=(w, v) has floating point we output: A vector K of goods drawn from G, such that the sum of volumes of goods from K is within the prescribed volume limit Vand the sum of the goods' weight is maximized. In other word
```

Code Compilation and Execution

```
File mayweight hh has been modified. Save it (v n ^G)?
            student@tuffix-vm:~/Desktop/cpsc335-01proj2$ make
            g++ -std=c++17 -Wall maxweight_test.cc -o maxweight_test
            In file included from maxweight_test.cc:13:
            maxweight.hh: In function 'std::unique_ptr<std::vector<std::shared_ptr<CargoItem> > >
            maxweight.hh:511:22: warning: comparison of integer expressions of different signedne
            ·compare]
                     for (int i = 1; i < ( todo->size() -1 ); i++)
             511
            maxweight_test.cc: In lambda function:
            maxweight_test.cc:170:47: warning: comparison of integer expressions of different sig
                      for ( int optimal_index = 0; optimal_index < optimal_weight_totals.size();</pre>
             170
            ./maxweight_test
            load_cargo_database still works: passed, score 2/2
            filter_cargo_vector: passed, score 2/2
            greedy_max_weight trivial cases: passed, score 2/2
            greedy_max_weight correctness: passed, score 4/4
            exhaustive_max_weight trivial cases: passed, score 2/2
            exhaustive max weight correctness: passed, score 4/4
    mainre
         TOTAL SCORE = 16 / 16
            student@tuffix-vm:~/Desktop/cpsc335-01proj2$
∷
```

Empirical Data

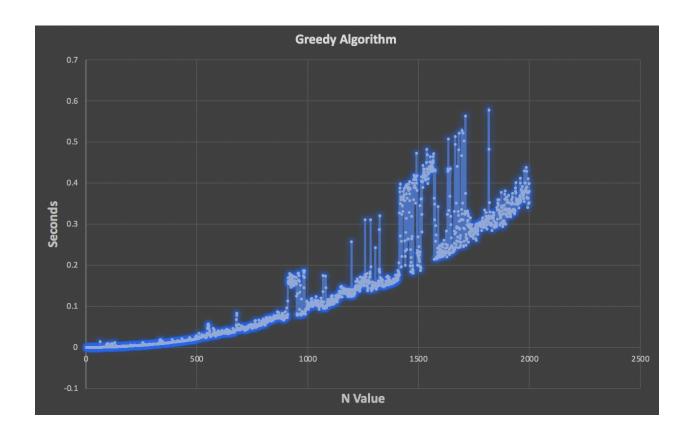
Greedy (smaller sample due to the csv file containing 2000 records)

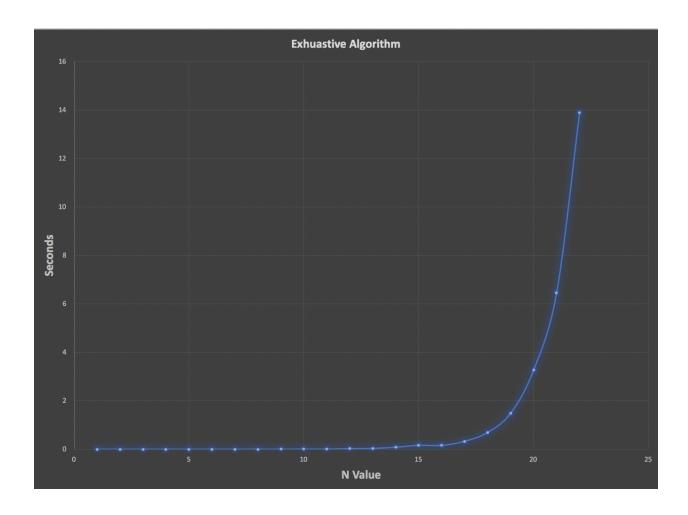
n seconds 1 5.295E 2 2.094E 3 2.219E 4 3.203E 5 3.886E	-06 -06 -06 -06
2 2.094E 3 2.219E 4 3.203E	-06 -06 -06 -06
3 2.219E 4 3.203E	-06 -06 -06
4 3.203E	-06 -06 -06
	-06 -06
5 3.886E	-06
6 4.906E	
7 6.245E	-06
8 7.596E	-06
9 9.311E	-06
10 1.082E	-05
11 1.315E	-05
12 1.514E	-05
13 1.693E	-05
14 1.918E	-05
15 2.158E	-05
16 2.458E	-05
17 2.685E	-05
18 2.993E	-05
19 3.337E	-05
20 3.643E	-05
21 3.959E	-05
22 4.4E	-05
23 4.695E	-05
24 5.028E	-05
25 5.525E	-05
26 5.905E	-05
27 6.343E	-05
28 6.826E	-05
29 0.00010	95
30 7.918E	-05

Exhaustive

n	seconds
1	1.1157E-05
2	3.681E-06
3	7.081E-06
4	1.5916E-05
5	3.4214E-05
6	7.4252E-05
7	0.00016354
8	0.00035618
9	0.00079641
10	0.00165374
11	0.00386692
12	0.02305299
13	0.03190128
14	0.08324177
15	0.16080957
16	0.1602824
17	0.32855243
18	0.6942229
19	1.48594501
20	3.26622518
21	6.45566497
22	13.8857117

Scatter Plots





Mathematical Analysis

This section includes pseudocode, the step count, and Big O Time Complexity

Filter Cargo Vector

```
if itemCount < total_size // 1
        add to ValidCargo // 1
            itemCount++ // 1
       else
        break out of for loop // 1
 End for
return ValidCargo // 1
Calculation....
1 + 1 + n * (2 + 1 + 1 + 1 + 1) + 1 =
2 + n * (6) + 1 =
3 + 6n
Filter Cargo Vector step count is 3 + 6n
Lim as n \rightarrow infinity 3 + 6n
Lim as n \rightarrow infinity 3 + 6n / n
Lim as n \rightarrow infinity 3/n + 6
Apply limit: 3/n = 0, and 0 + 6 = 6
6 \ge 1, so our proof is successful
Filter Cargo Vector is O(n) Time Complexity
Greedy
Step Count.....
```

```
empty result // 1
empty todo = goods // 1
result volume = 0 // 1
v = 0 // 1
  while todo is not empty do: // n (log n)
      // max item index tracker
      maxIndex = 0 // 1
       // Find a good a of w/v ratio
        for i = 1 to todo->size() -1 do: // N - 1
          if item weight/volume ratio at i > item weight/volume ratio at maxIndex // 2
            maxIndex = i // 1
        End for
        v = maxIndex->volume() // 1
        //add to greedy solution if v <= V
        if result volume + v \le V // 2
           add maxIndex to result // 1
           result volume += v // 1
       //remove from todo so as not to compare this item again
        remove maxIndex from todo // 1
  End while
  return result // 1
Calculation.....
4 + n(\log n) * (1 + (n - 1)*(1 + 2) + 1 + 2 + 1 + 1 + 1) + 1 =
4 + n(\log n) * ((3n - 3) + 7) + 1 =
4 + n(\log n) * ((3n - 4) + 1 =
```

```
5 + 3n(log(4) n) =
Greedy Algorithm step count is 5 + 3n(\log(4) n)
Lim as n \rightarrow \inf_{n \rightarrow \infty} 5 + 3n(\log(4) n)
Lim as n -> infinity 5 + 3n(\log(4) n) / n(\log(4) n)
Lim as n -> infinity (1/n) / (1/n) = 1
(1/n) / (1/n) = 1, 1 > 0
Exhaustive algorithm is O( n log(n) )
Exhaustive
Step Count.
sum vector step count.....
total volume = total weight = 0; // 1
       for (auto& item: goods) // N
             total volume += item->volume(); // 1
             total weight += item->weight(); // 1
step count for sum function is 1 + n * (1 + 1) = 1 + 2n = o(n) time
Step count for greedy......
empty bestCargo // 1
candidateTotalWgt = 0 // 1
candidateTotalVol = 0 // 1
bestTotalWgt = 0 // 1
```

```
bestTotalVol = 0 // 1
//size of goods
size t n = size of goods // 1
 //for bitwise method, vector must be <= 64 in size
 if n \ge 64 // 1
  overflow and exit // 1
 for (uint64 t b = 0; b < pow(2, n); b++) do: //(2^n) + 1
    empty candidate vector // 1
   for (uint64 t i = 0; i < n; i++) do: // n + 1
      if (((b >> i) & 1) == 1) // 3
         add i to candidate vector // 1
    End for
   // calculate totals of candidate and best
   sum of candidate // 1 + 2n
   sum of current best // 1 + 2n
   //if within budget and has greater total time than current best
   if candidateTotalVol <= total volume // 1
     if bestCargo->empty() OR candidateTotalWgt > bestTotalWgt // 3
         bestCargo vector = candidate vector // 1
 End for
return bestCargo // 1
8 + ((2^n + 1) * (6 + (2n+1) + (2n+1) + 5)
8 + ((2^n + 1) * (4n + 2 + 11) =
```

$$8 + ((2^n + 1) * (4n + 13) =$$

$$8 + (2^n)(4n) + (2^n)(13) + 4n + 13 =$$

$$(2^n)(4n) + (2^n)(16) + 4n + 21$$

Greedy Algorithm step count is $(2^n)(4n) + (2^n)(16) + 4n + 21$

Proof.....

Let
$$t = (2^n)(4n) + (2^n)(16) + 4n + 21$$
 be in O (2^n)

$$8 + 32 + 4 + 21 = 65$$
, let $c = 65$, let $n = 1$

Let
$$(2^n)(4n) + (2^n)(16) + 4n + 21 > n * 2^n$$

$$65 * (2^n)(4n) + (2^n)(16) + 4n + 21$$

$$130 - 8 - 32 - 4 - 21 > 0,65 > 0$$

Exhaustive algorithm is O(n * 2^n)

Questions

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

Yes, after the N time grows large enough, there is a noticeable difference. The exhaustive algorithm is much slower than the greedy algorithm. This does not surprise me because exhaustive is $O(2^n * n)$, and greedy is $O(n \log(n))$.

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

The mathematical evidence and the empirical analysis align. The exhaustive graph shows a larger jump in time than the greedy algorithm.

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

The evidence is consistent with hypothesis 1, as the exhaustive algorithm will produce the best output in more time. Accuracy is important, even at the expense of time.

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

The evidence also supports hypothesis 2, because the exhaustive algorithm of exponential time is extremely slow. It is definitely impractical to use in a real world setting.