JORDHY TAPIA
JORDHYT@CSU.FULLERTON.EDU
PROJECT 2 SUBMISSION

RESPONSES

1. 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 definitely a noticeable difference in the performance of the two algorithms. The greedy algorithm is a lot faster compared to the exhaustive search. When I tested the speed of the algorithms with 16 armors, the greedy algorithm was about 28,000 times faster with a speed of 0.000007 compared to the exhaustive algorithm of .2 seconds. This does not surprise me as we have studied in class that the exhaustive algorithm takes a considerable amount of time.

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

   My empirical analyses is consistent with the mathematical analysis. The greedy algorithm had a n^2 time complexity. This does not show well in the graph because the test cases were very small, it only looks like a straight line. However, if larger armor vectors were used I am certain a curve would appear as it heads to infinity. The graph for the exhaustive search is consistent with the mathematical analysis of n*2^n even with only 16 armors in the armor vector. It has an exponential curve.
3. Is this evidence consistent or inconsistent with hypothesis 1? Justify your answer.

   Exhaustive searches are feasible to implement and produce correct outputs. This is true because this project required the implementation of an exhaustive search and it in fact produced the correct output.

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

   Although I was able to implement the exhaustive search, using it on an ArmorVector with more than 16 armors resulted time that was too long for practical use. No practical computer program uses any function that requires minutes or even hours to produce a result.

```
std::unique_ptr<ArmorVector> greedy_max_defense
(
      const ArmorVector& armors,
      double total_cost
)
{
    ArmorVector copy = armors;
    std::unique_ptr<ArmorVector> result( new ArmorVector);
      double resultCost = 0;
      double defPoints = 0;
     int defMax = 0;

    while(!copy.empty())
    {
        for ( int i = 0; i < copy.size(); ++i)
        {
            if(copy[i]->defense() > defPoints)
            {
                defPoints = copy[i]->defense();
                defMax = i;
            }
        }
        if(resultCost + copy[defMax]->cost() <= total_cost)
        {
            result->push_back(copy[defMax]);
            resultCost += copy[defMax]->cost();
        }
        swap(copy.at(defMax),copy.back());
        copy.pop_back();
        defPoints = 0;
        defMax = 0;

    }

    return result;
}
```

+4 S.C

n S.C

n S.C

1+(2,0) S.C → 3 S.C

2+(2,0) S.C

+4 S.C

$4 + n(n(3)+4+4)$

$\downarrow$

$4 + n(3n+8)$

$\downarrow$

$4 + 3n^2 + 8n$

$\downarrow$

$3n^2 + 8n + 4 \rightarrow \boxed{O(n^2)}$

```
std::unique_ptr<ArmorVector>
(
    const ArmorVector& armors,
    double total_cost
)
{
    const int n = armors.size();
    assert(n < 64);

    std::unique_ptr<ArmorVector> best(new ArmorVector);
    double candidateCost;
    double candidateDefense;
    double bestCost;
    double bestDefense;

    for ( int i = 0; i < pow(2,n); ++i)
    {
        std::unique_ptr<ArmorVector> candidate(new ArmorVector);

        for ( int j = 0; j < n; ++j)
        {
            if ((i>>j) & 1)
            {
                candidate->push_back(armors[j]);
            }
        }

        sum_armor_vector(*best, bestCost, bestDefense);
        sum_armor_vector(*candidate, candidateCost , candidateDefense);

        if (candidateCost <= total_cost)
        {
            if(best->empty() || (candidateDefense > bestDefense))
               *best = *candidate;
        }

    }

    return best;
}
```

$]+2$ s.c

$+1$ s.c $+$ $+1$ s.c $= 2$ s.c

$] = 6$ s.c $] = 8$ s.c

2^n s.c

1 s.c $+$ 1 s.c $= 2$ s.c

$+2^n(2+n(3)+n+n$

$3+n)$

$+2$ s.c

n s.c

2 s.c $= 3$ s.c

$+1$ s.c

n s.c.

n s.c

$1+(2+(n,0),0) = 3+nc$

2 s.c

n s.c

$8+2^n(2+3n+n+n+3+n)$

$2^n(6n+5)+8$

$2^n(6n) + 2^n(5) + 8$

$O(n) = 2^n \cdot n$

Armors	Greedy Time	Armors	Exhaustive Time
2	0.000002564	2	0.000053782
4	0.000003609	4	0.000072225
8	0.000005142	8	0.00090743
16	0.000007277	16	0.230091