CPSC 335-01 project 4.
Jenni Wu (Chih-Yi Wu)
Email: chihyi1126@csu.fullerton.edu

1. Analyze each algorithm mathematically to determine big-O efficiency class.

- **Dynamice_max_defense**

```
heap sort(todo,n)          nlogn
m = total_cost             1tu
dp[0][0]= NONE             1tu
for j = 1 to m do          m·1=m
        dp[0][j] = NONE
endfor
for i = 1 to 2 do          2·1=2
        dp[i][0] = NONE
endfor
while i <= n do            n
        for int j = 1 to m do   m
                if cost[i] > j do        1+max(1,2) =3
                        dp[n][j] = dp[p][j]
                else
                        dp[n][j]= max(dp[p][j].dpdefense, defense[i] + dp[p][j -itemCost].dpdefense)
                endif
                if (i == n) && (j= = m) do   3+max(1,0)=4.
                        add arr2(item) in the beginning of finalresult
                endif
        endfor
endwhile
return finalresult         1tu.
```

m is total_cost.

SC= nlogn+1+1+ m+2 +n(m (3+ 4))+1
= nlogn + 5 + m + 7mn+1
= nlogn + 7mn + m+ 6
O (nlogn + mn) if m < logn then O(mn).

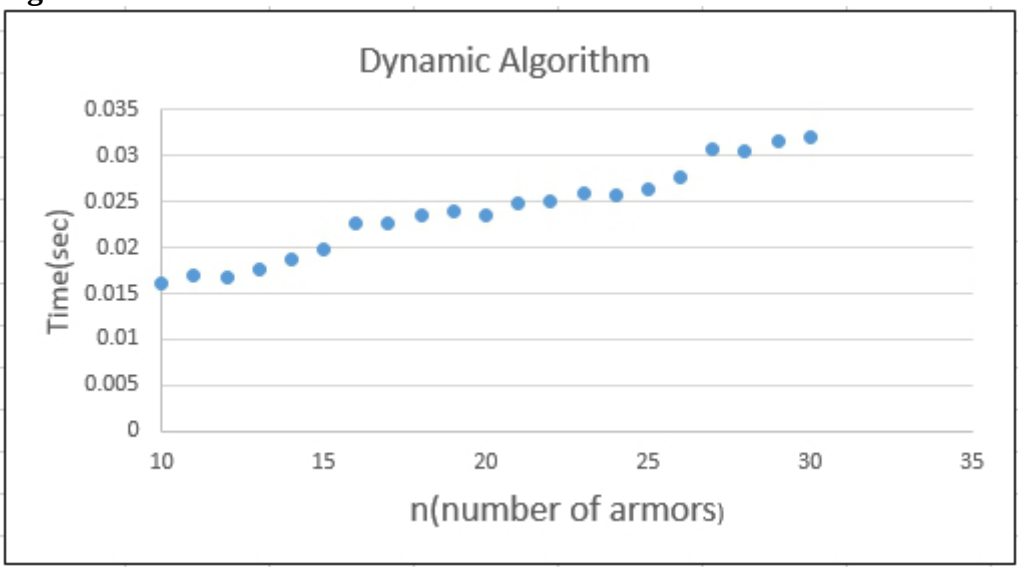
- **Exhaustive_max_defense**

```
for i from 0 to (2^n -1) do                                          //2^n tu
        for j from 0 to n-1:                                         //n tu
                if (bits >> j) & 1 == 1:                             //3 tu
                        candidate.add_back(armor_items[j])           //1 tu
        if total_gold_cost(candidate) <= G:                          //1 tu
                if best.size()==0 || candidate_total_defense> best.defense   //3 tu
                        best = candidate;                            //1 tu
return best                                                          //1 tu
```

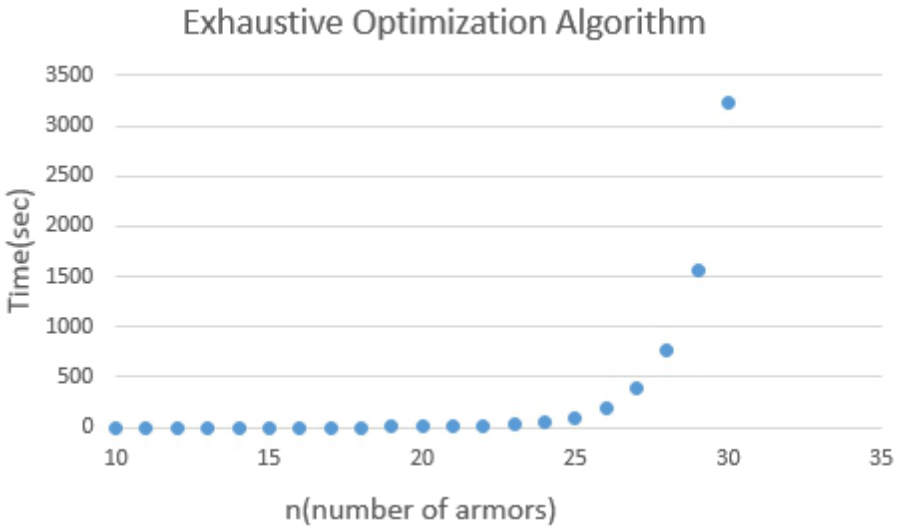
SC= 2^n * (n*(3 + max(1,0)) + (1 + max(3 + max(1,0)))) + 1
=2^n * (4n + 5) +1
O(n^2 *n)

2. Scatter plots

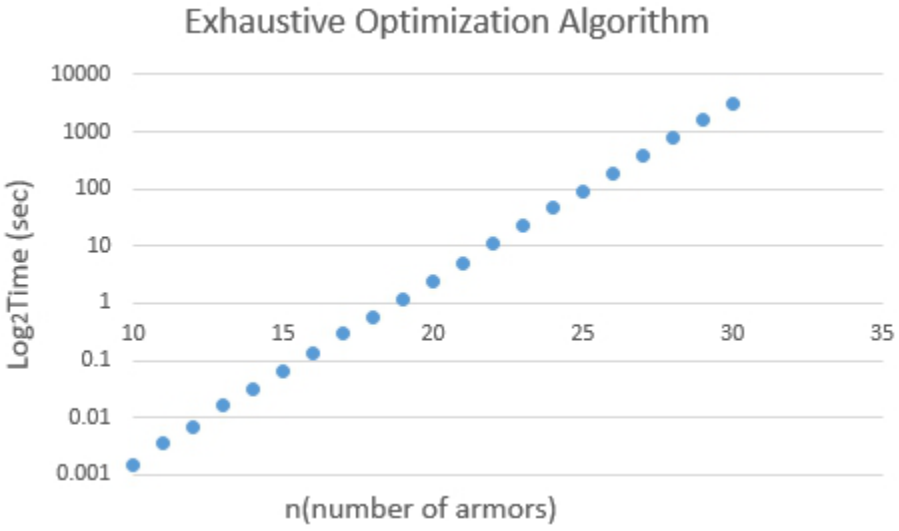
- **Dtnamic Algorithm**



- **Exhaustive Optimization Algorithm**



If take Log2 on the execution time, the results will appear to be close to a liner line, whcih is also expected.



3. Answer following questions:

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

Yes, there is a noticeable in the performance ot these two algorithms. Dynamic algorithm is much faster compared to exhaustive optimization algorithm. Considering the Time complexities of these two algorithms, this is not a surprise result at all. The time complexity of Exhaustive Optimization Algorithm is 2^n/m+log2n times of the time compexity of Dynamic algorithm. I already knew that Exhaustive Optimization Algorithm needs to take time from project2, so it is not surprise for me again.

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

Yes, the big-O of dynamic algorithm is O(m + n*$\log_2 n$), so I expectect the result is close to a line, and the result consist with my expectation.

the big-O of exhaustive optimization algorithm is O(n*2^n), so I expectect the result is close to a exponential line, and the result consist with my expectation.

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

Hypothesis1: Exhaustive search algorithms are feasible to implement, and produce correct outputs.

Yes, exhaustive optimization algorithm will give correct answer, but it takes a long time to execute.

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

Hypothesis2: Algorithms with exponential running times are extremely slow, probably too slow to be of practical use.

Yes, Algorithms with exponentential runnings times are taking too much time to execute. There are other ways that can save running time and also produce acceptable outputs such as dynamic algrithm with logarithmic running times.