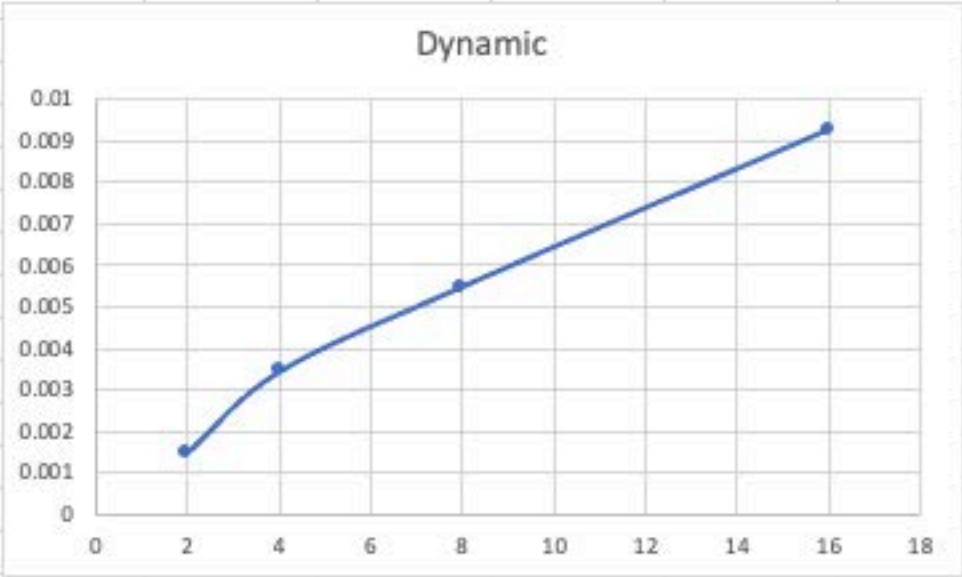
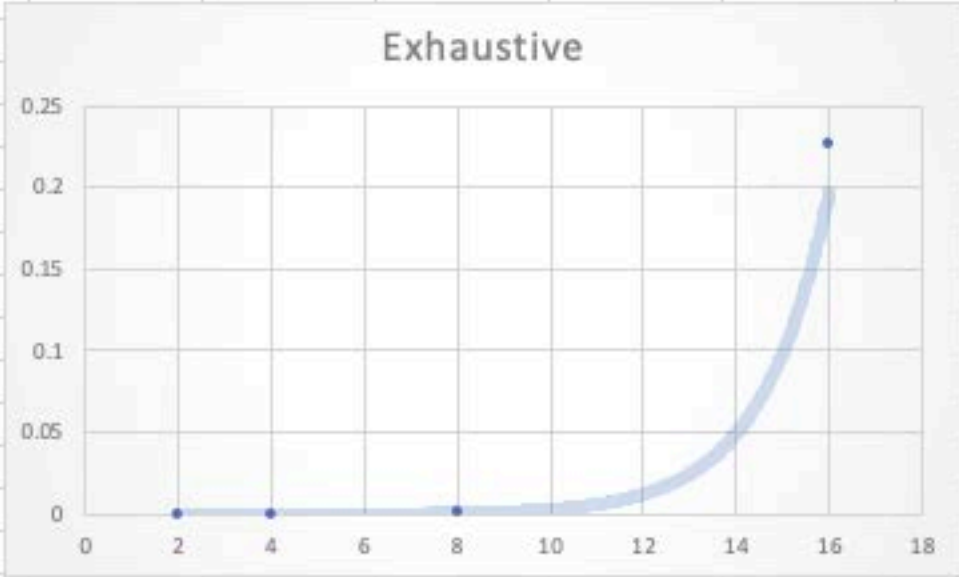
armors	dyamic time	exhaustive time	Jordhy Tapia
2	0.00149421	0.0000226	
4	0.00343149	0.0000249	
8	0.00547488	0.0006994	
16	0.00925247	0.226062	





Jordhy Tapia

```
Dynamic max defense ( Armors, total cost)
{
<Vector <vector>>double  map                        1 s.c    ]  1+n(1+n(1))+1 -> [n^2]
for  i=0 to armors.size+1 do                         n s.c    ]
    vector<double> v;                                1 sc     ]
    for j=0 to total cost +1 do                      n s.c    ]
        v. pushback (0)   +                          1 sc     ]
    map. push back (v)       +1 sc                   1 sc     ]
    end for
end for
for  i=1 to map.size() do                            n sc     ]  n(2+n(5)) -> [n^2]
    double d = armors [i-1] -> defense()             +1 sc    ]
    int  c = armors [i-1] -> cost ()                 +1 sc    ]
    for int j=1 to  map[i].size() do                 n sc     ]
        map[i][j] = map [i-1] [j]                    +1sc     ]
        if ( j >= c && map [i-1][j-c] + d > map [i][j]    3+max(1,0) -> 4 sc ]
            map [i][j] = map[i-1][j-c]+d
        endif
    end for
end for
int cap = map[0].size                                +1 sc    ]  2+3n = [3n]
std:: unique_ptr <ArmorVector> selected (new ArmorVector)   +1 sc  ]   = n
for  map.size-1  to 1                                n sc     ]
    if map [i][ cap ] != map [i-1][cap]              1+max(3,0) -> 4 sc ]
        size_t  itemIndex = i-1
        selected -> push back [armors [ itemIndex]]
        cap -= armors [ itemIndex ] -> cost          n^2+n^2+3n = [O(n^2)]
    end if
end for

return  selected
}
```

```
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

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

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

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

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