# 1. Design Patterns:

### ①Creational patterns – Singleton:

Class Finder has only one instance to find the profit and path. Singleton reduce the cost of memory and resources.

Using Singleton is hard to extend the original code which violates the Open Closed Principle(OCP).

### ②Creational patterns – Builder:

Builder has good encapsulation and separate the construct with show and it is more flexible to extend.

Builder is more expensive to maintain.

# 2. Analysis of algorithms:

## ①Brute Force:

$T(n) = 3T(n-1)$

$= 3^2T(n-2)$

…

$= 3^nT(1)$

So that the time complexity is: **$O(3^n)$**

## ②Approximate:

Find min cost town cost: O(n)

**Worst case:**

The cost of town is sorted from small to large and value of each town is larger than the cost of last town.

Example: A[1:0] B[3:2] C[5:4] D[7:6] E[9:8]

Time complexity: **$O(n^2)$**

**Best case:**

Pick up load at the second last town and drop off at the last town.

Time complexity: **O(n)**

**Average case:**

**$T(n) = n + T(n / 2)$**

According to master theorem:

Time complexity: **O(n)**

## ③Exact:

Using dynamic programming:

Calculating the 3xn matrix costs 3n

Finding path cost 3n

Total: 3n + 3n = 6n

Time complexity: **O(n)**