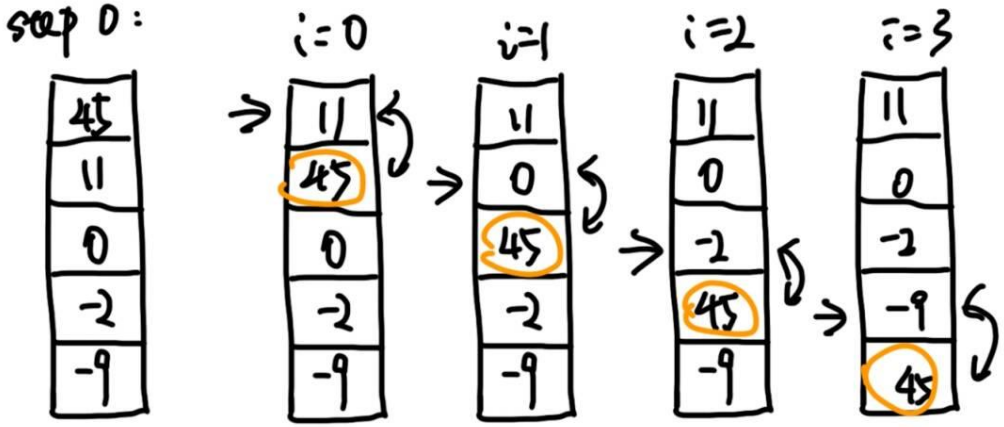
Q1 Bubble Sort

Bubble sort is a simple sorting algorithm that repeatedly steps through the input list element by element, comparing the current element with the one after it, swapping its values if needed. **These pass through the list are repeated until no swaps have to be performed during a pass**, meaning that the list has become fully sorted.

Need a flag

First Iteration (Compare and Swap)

1.Starting from the first index, compare the first and the second elements.
2.If the first element is greater than the second element, they are swapped.
3.Now, compare the second and the third elements. Swap them if they are not in order.
4.The above process goes on until the last element.



How many times to iterate?

N-1 times in the worst case

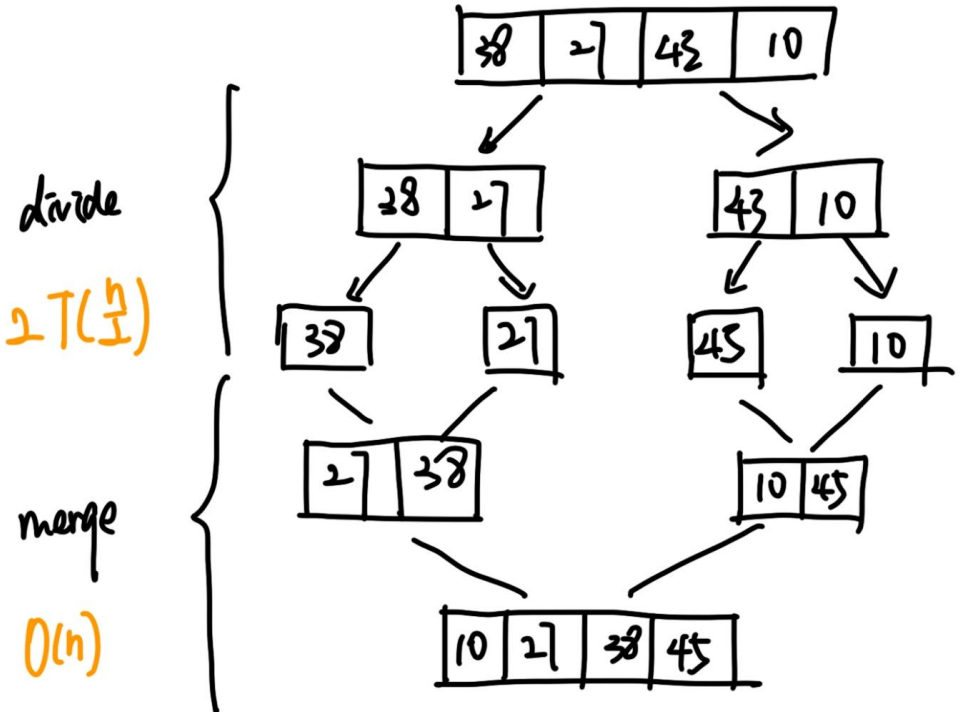
Time complexity: $O(n^2)$

Q1 Merge Sort

Conceptually, a **merge sort** works as follows:

In common practice, n=2

- **Divide** the unsorted list into n sub-lists, each containing one element (a list of one element is considered sorted).
- Repeatedly **merge** sublists to produce new sorted sublists until there is only one sublist remaining. This will be the sorted list.



Time complexity: O(nlogn)

$$T(n) = 2T\left(\frac{n}{2}\right) + n$$
$$= 4T\left(\frac{n}{4}\right) + 2n$$
$$= 8T\left(\frac{n}{8}\right) + 3n$$
$$\dots$$
$$= 2^k T\left(\frac{n}{2^k}\right) + kn \qquad k = \log_2 n$$
$$\dots$$
$$= nT(1) + \log_2 n \, n \in O(nlogn)$$

Q2 Fibonacci Sequence

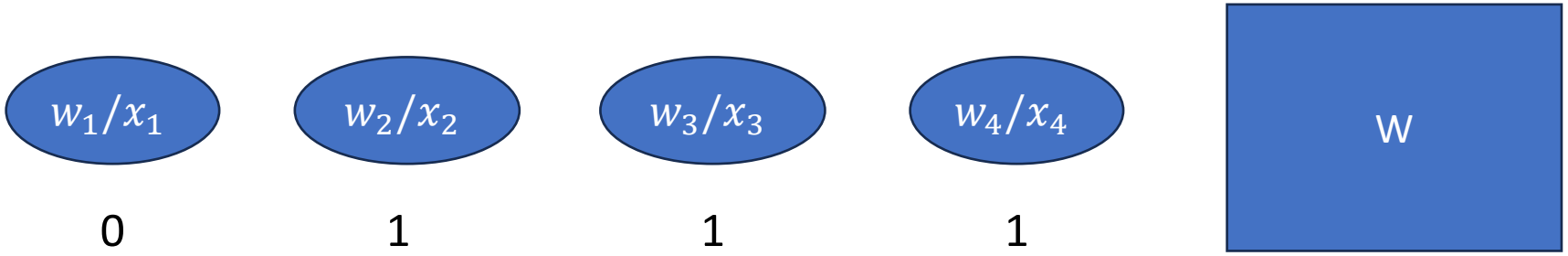
$$F(n) = \begin{cases} 0, & n = 0 \\ 1, & n = 1 \\ F(n-1) + F(n-2), & n \geq 2 \end{cases}$$

Q3 Knapsack problem

$$Maximize\ Z = w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

$$Subject\ to$$

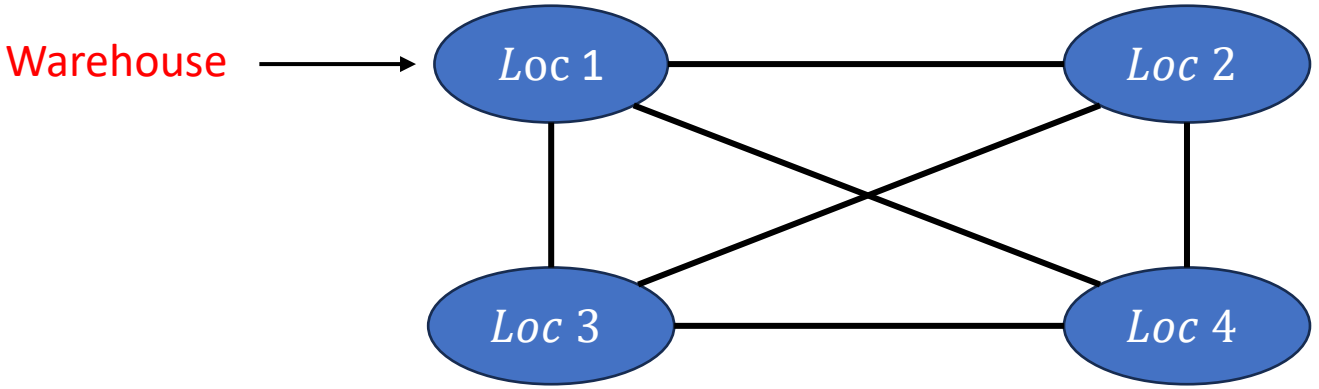
$$\begin{cases} w_1, w_2, \ldots w_n\ are\ integers \\ \sum_{i}^{n} w_i \leq W \end{cases}$$



Steps:

1. Read the input
2. For each scenario:
   (1) Find the best artifacts while calculating the time
      i. Get all combinations of possible artifacts (itertools.product([0, 1], repeat=n))
      ii. Keep track of the best combination
   (2) Add the results to the lists
3. Plot the results

Q4 Travelling Salesman Problem (TSP)



Steps:

1. Read the input
2. For each scenario:
   (1) Find the shortest distance while calculating the time
      i. Get all permutations of locations (itertools.permutations(locations[1:])) and calculate the total distance
      ii. Keep track of the best route
   (2) Add the results to the lists
3. Plot the results