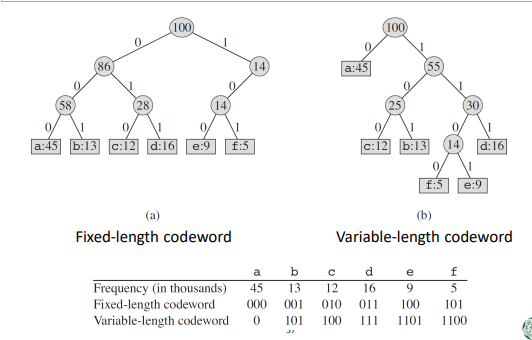
**Lecture 8: Priority queue**

**Huffman Code**

****

: a Huffman code is a particular type of optimal [prefix code](https://en.wikipedia.org/wiki/Prefix_code) that is commonly used for [lossless data compression](https://en.wikipedia.org/wiki/Lossless_data_compression).

– A binary tree can be used to compress the data for which the frequency of each element is known. – This kind of binary tree is called the Huffman coding tree.

**Lecture 9: Sort**

**Sorting**

• Sorting lists a set of data in ascending or descending order

• Sorting is essential when searching data

**Sorting Algorithms**

• Simple but inefficient algorithms

– Insert sort, selection sort, bubble sort

• Complex but efficient algorithms

– Quick sort, heap sort, merge sort, radix sort

• Internal sort

– Sort a set of data that is stored in main memory

• External sort

– Sort a set of data, when most of the data is stored in the external storage device and only a part of the data is stored in the main memory

**Time complexity of Sorting Algorithms**

**테이블이(가) 표시된 사진

자동 생성된 설명**

**Selection Sort**

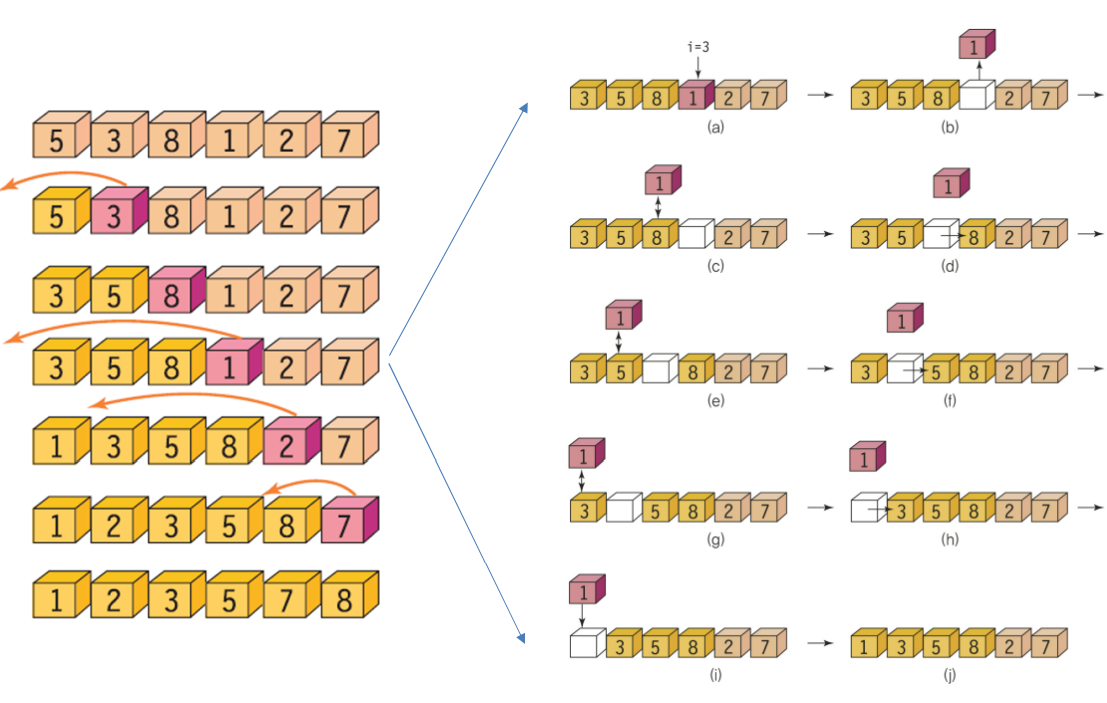
**텍스트이(가) 표시된 사진

자동 생성된 설명**selection sort is an [in-place](https://en.wikipedia.org/wiki/In-place_algorithm) [comparison](https://en.wikipedia.org/wiki/Comparison_sort) [sorting algorithm](https://en.wikipedia.org/wiki/Sorting_algorithm). It has an [O](https://en.wikipedia.org/wiki/Big_O_notation)(*n*2) [time complexity](https://en.wikipedia.org/wiki/Time_complexity), which makes it inefficient on large lists, and generally performs worse than the similar [insertion sort](https://en.wikipedia.org/wiki/Insertion_sort). Selection sort is noted for its simplicity and has performance advantages over more complicated algorithms in certain situations, particularly where [auxiliary memory](https://en.wikipedia.org/wiki/Auxiliary_memory) is limited.

The algorithm divides the input list into two parts: a sorted sublist of items which is built up from left to right at the front (left) of the list and a sublist of the remaining unsorted items that occupy the rest of the list. Initially, the sorted sublist is empty and the unsorted sublist is the entire input list. The algorithm proceeds by finding the smallest (or largest, depending on sorting order) element in the unsorted sublist, exchanging (swapping) it with the leftmost unsorted element (putting it in sorted order), and moving the sublist boundaries one element to the right.

**Insertion Sort**

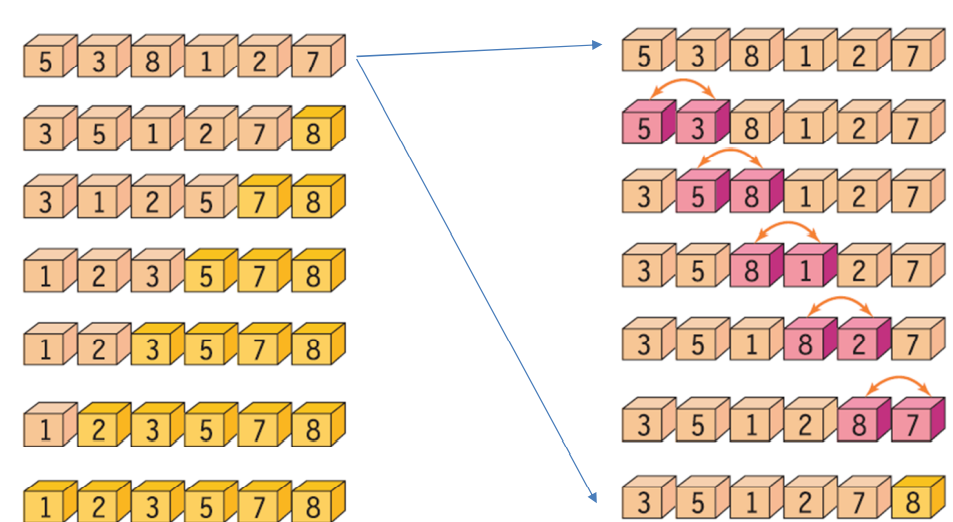
: Insertion sort is a simple [sorting algorithm](https://en.wikipedia.org/wiki/Sorting_algorithm) that builds the final [sorted array](https://en.wikipedia.org/wiki/Sorted_array) (or list) one item at a time. It is much less efficient on large lists than more advanced algorithms such as [quicksort](https://en.wikipedia.org/wiki/Quicksort), [heapsort](https://en.wikipedia.org/wiki/Heapsort), or [merge sort](https://en.wikipedia.org/wiki/Merge_sort).



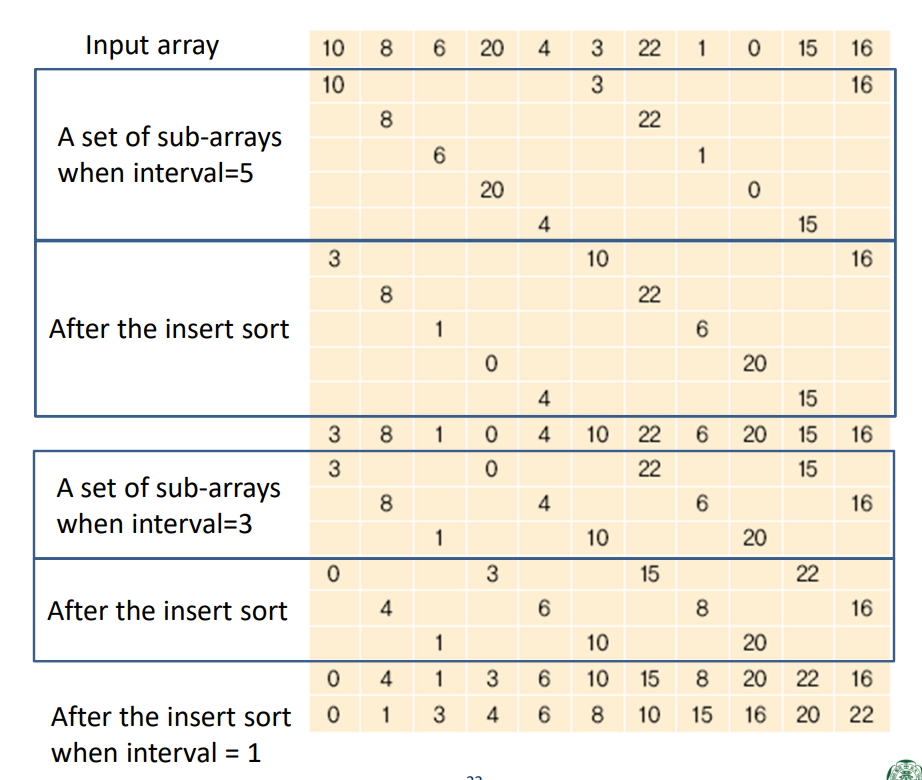
**Bubble Sort**

– Swap two adjacent data, when they are not in order

– This comparison-exchange process is repeated from the left to the right.

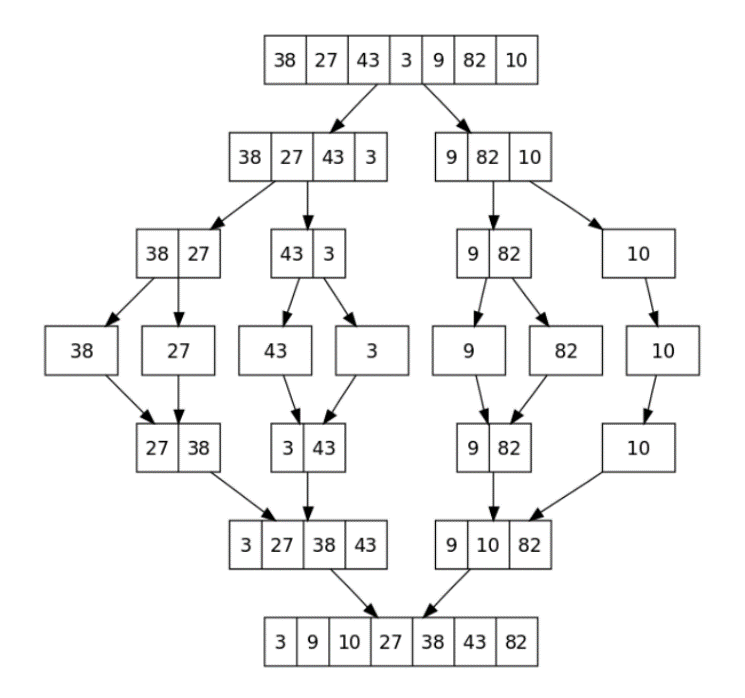
****

**Shell Sort**

****

: Shellsort, also known as Shell sort or Shell's method, is an [in-place](https://en.wikipedia.org/wiki/In-place_algorithm) [comparison sort](https://en.wikipedia.org/wiki/Comparison_sort). It can be seen as either a generalization of sorting by exchange ([bubble sort](https://en.wikipedia.org/wiki/Bubble_sort)) or sorting by insertion ([insertion sort](https://en.wikipedia.org/wiki/Insertion_sort)). The method starts by sorting pairs of elements far apart from each other, then progressively reducing the gap between elements to be compared. By starting with far apart elements, it can move some out-of-place elements into position faster than a simple nearest neighbor exchange.

**Merge Sort**

****divide and conquer

First divide the list into the smallest unit (1 element), then compare each element with the adjacent list to sort and merge the two adjacent lists. Finally all the elements are sorted and merged.