2. Ordenação por inserção (Insertion Sort)

2.1. Ideia Básica

O método de ordenação por inserção é o mais rápido entre os métodos básicos(método das bolhas, método de selecção directa e método de ordenação por inserção).

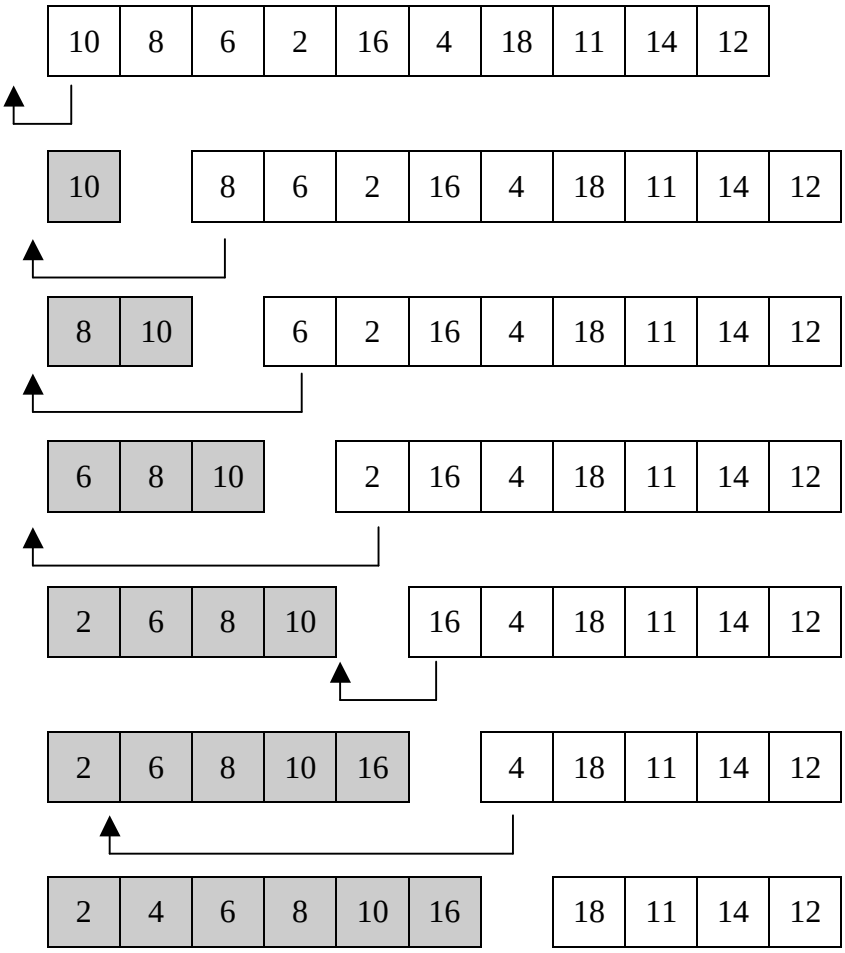
Um exemplo deste tipo de ordenação, ocorre todos os dias, por exemplo enquant se joga cartas. Para ordenar as cartas da mão, va-se retirando uma carta de cada vez, junta-se as outras e então insere-se a carta extraída no lugar correcto. Este processo é repetido até todas as cartas se encontrem na ordem correcta.

A principal característica deste método consiste em ordenar um conjunto de elementos, utilizando um subconjunto ordenado localizado em seu inicio, e em cada iteração, acrescentamos a este subconjunto mais um elemento, até que atingimos o último elemento do conjunto assim com que ele se torne ordenado.

2.2. Método

Este método, considera-se o array a ordenar como um array dividido em dois sub-arrays (esquerdo e direito), com o da esquerda ordenado e o da direit desordenado. Os elementos são retirados um de cada vez do sub-array da esquerda (não ordenado), e move-se esse elemento para o sub-array da esquerda, inserindo-o na posição correcta por forma a manter o sub-array da esquerda ordenado, terminando o processo quando o sub-array da d reita ficar vazio.

Para ordenar os seguintes dados por ordem crescente, processa-se do seguint modo:



2.3. Algoritmo

"Este algoritmo considera o array como contendo uma**parte ordenada** (sub-array da esquerda) e uma **parte não ordenada** (sub-arrya da direita)".

1. Iniciar a "**parte ordenada**" com o primeiro elemento do vector;
2. Retirar um elemento do sub-array da esquerda (não ordenado);
3. inserir o elemento na posição correcta (ordenado) no sub-array da esquerda;
4. Voltar ao passo 2.

"Pode-se terminar o processo quando o sub-array da direita ficar vazio".

2.4. Eficiência

Tempos de Ordenação (sendo **N** o número de elementos a ordenar)

- Melhor caso = 1 + 1 + ... + 1 (*n-1 vezes*) = n-1 ≈ **n** (dados já ordenados), dado que o ciclo interno nunca será executado, pois a sua condição falha sempre.
- Média = metade do anterior, isto é, aproximadament **$n^2/4$**
- Pior caso = 1 + 2 + ... + n-1 = (n-1)(1 + -1)/2 = n(n-1)/2 ≈ **$n^2/2$** (dados ordenados em ordem inversa), sendo N o número de elementos do array, o ciclo mais abrangente deverá ser executado N vezes, e o ciclo in erno será executado N vezes por cada ciclo mais abrangente...

2.5. Implementação em C++

```
void insertionSort(vector<int>& v)
{
  for(int i = 1; i < int(v.size()); i++)
    for(int j = i; j > 0 && v[j - 1] > v[j]; j--)
      swap(v[j], v[j - 1]);  // puxar os valores até encontrar a posição correcta
}
```

2.6. Resumo

Método de ordenação, na qual são procurados sucessivos elementos que se encontram fora de ordem, retira o elemento da lista e depois insere o elemento de forma ordenada. Este tipo de ordenação em pequenas listas é rápido, sendo extremamente lento para grandes listas.