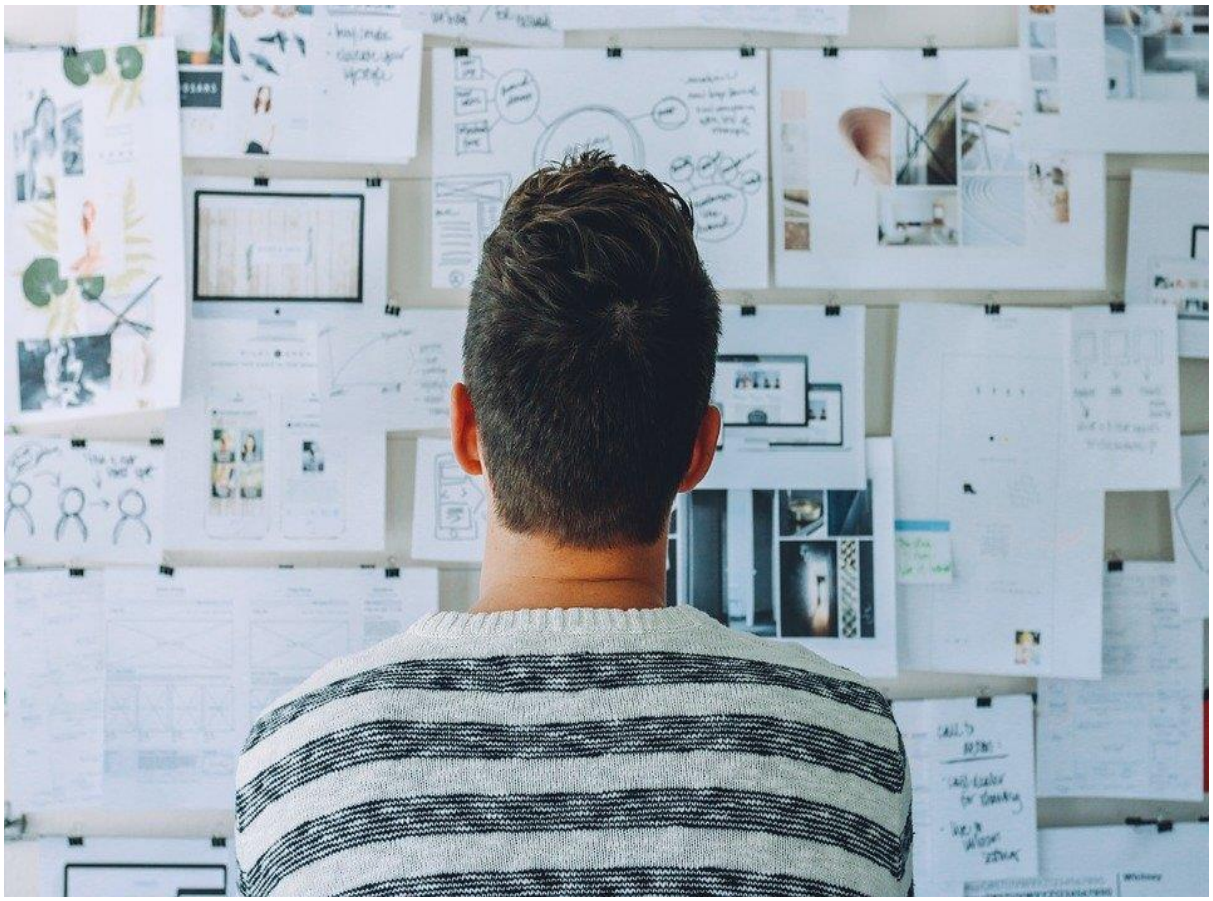
AULA 6: ALGORTIMOS RECURSIVOS (INSERTION SORT)



Fonte: https://pixabay.com/pt/photos/conceito-homem-pap%c3%a9is-pessoa-plano-1868728/

Caro aluno, seja bem-vindo (a) ao nosso capítulo que descreve o comportamento de um tipo de algoritmo para ordenamento de dados chamado de ***insertion sort***. Este algoritmo pertence a categorias dos algoritmos de ordenação amplamente utilizados em desenvolvimento de algoritmos para resolução de problemas. Durante esse capítulo vamos entender o comportamento do algoritmo, prezando pelos seus fatores mais básicos até pontos estruturantes.

Vamos juntos!!

6.1 O que é o algoritmo *insertion sort*?

Um algoritmo *insertion sort* é um algoritmo de ordenação que tem como finalidade percorrer as posições do vetor começando pelo primeiro índice. Com novas posições desse vetor (ou seja, a cada novo atributo que for sendo recebido), será necessário fazer a inserção desse novo elemento no conjunto de vetor (por isso se chama ordenação por inserção). Neste caso, esse algoritmo constrói, no fim, uma matriz considerando um elemento por vez, bem como uma inserção por vez. Essa inserção precisa ser colocada no lugar certo, ou seja, a partir da ordenação crescente (Figura 6.1).

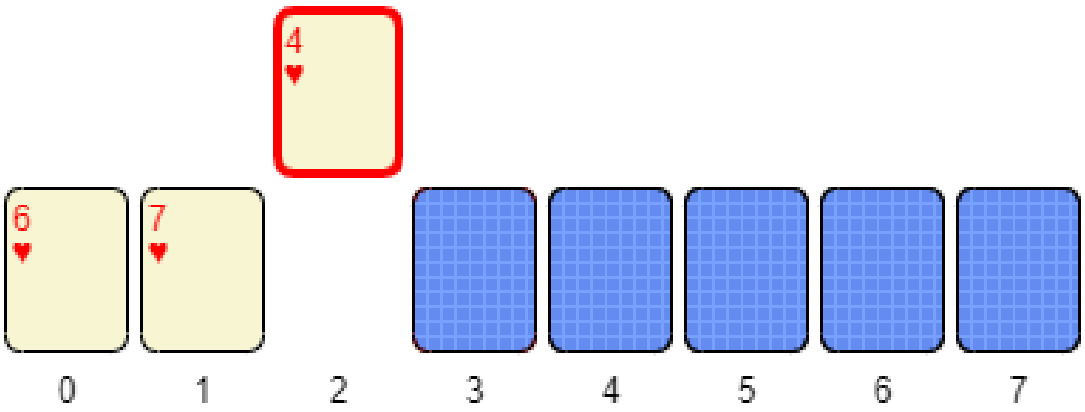


Figura 6.1: Inserção em Algoritmos Insertion Sort.

Fonte: https://pt.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/insertion-sort

Percebemos na figura 6.1 acima que temos uma carta com o valor número 4 na segunda posição. Logo, o algoritmo realizará o processo de ordenamento deste valor. Esse ordenamento levará em consideração os valores anteriores ao mesmo. Neste caso, na posição 0 e na posição 1 temos os valores 6 e 7 e estes são menores que o valor que vamos realizar a mudança de lugar (4) (Figura 6.2).

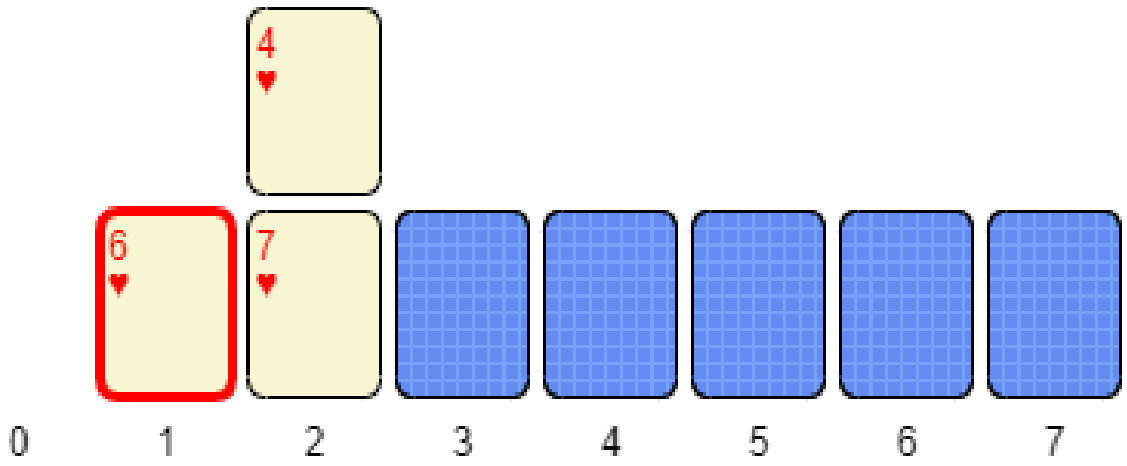


Figura 6.2: Inserção em Algoritmos Insertion Sort.

Fonte: https://pt.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/insertion-sort

Assim, a posição 0 fica vazia e receberá o valor de número 4. Houve, assim, uma alteração de posições, onde os elementos número 6 e 7 cederam espaço para o elemento de número 4.

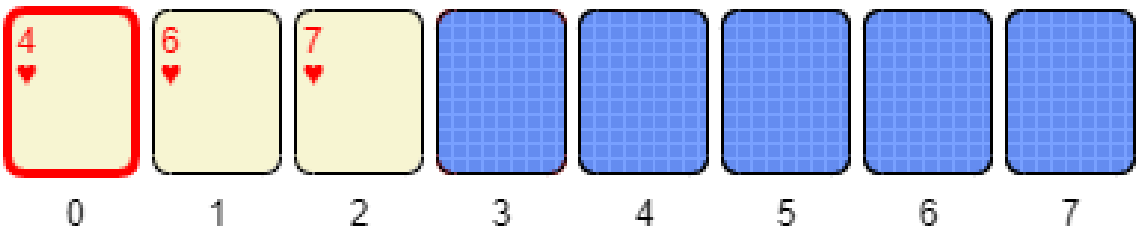


Figura 6.3: Inserção em Algoritmos Insertion Sort.

Fonte: https://pt.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/insertion-sort

#ANOTA AÍ#

Quanto ao seu desempenho, temos:

- Sua complexidade de pior caso é $O(n^2)$;
- Sua complexidade de caso médio é $O(n^2)$;
- Sua complexidade de melhor caso é $O(n^2)$;

#ANOTA AÍ#

6.2 Estruturação do algoritmo ***Insertion Sort.***

Existe um conjunto de etapas que são necessárias para resolver um algoritmo ***insertion sort***. A ideia principal desse algoritmo é ordenar os itens, inserindo-os em uma determinada posição.

No caso deste algoritmo existe uma chave. Essa chave se torna o valor de referência, e então começamos a comparar esse valor com os valores que estão dentro do nosso vetor. Vamos verificar o conjunto de etapas necessárias:

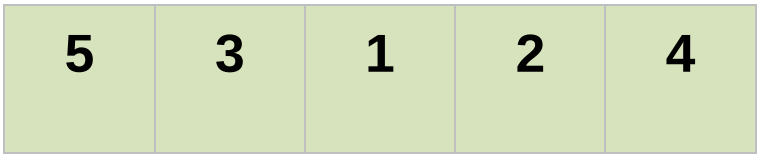
Primeiro passo: precisamos comparar o valor de determinado item que temos (nossa chave) com os outros itens que pertence ao nosso vetor. Essa comparação precisa ser executada pela esquerda.

Segundo passo: se determinado item for menor que o número que estamos tentando inserir no nosso vetor, precisamos deslocar esse item para a direta, com a finalidade de abrir um novo espaço.

Terceiro passo: por fim, se um determinado item for maior ou não tivermos mais nenhum item, logo significa que encontramos nossa posição. Precisamos inserir nossa chave no vetor.

Fácil né caro(a) aluno(a)? Que tal vermos um exemplo para ilustrar a aplicabilidade desse algoritmo?

Vamos considerar a seguinte lista:



Nossa chave inicial será o valor de 3 (a partir dessa chave faremos um processo comparativo com os demais elementos da nossa lista). Não optamos pelo primeiro elemento da lista, uma vez que não haveria com quem compará-lo. Desta forma consideramos que esse elemento já está ordenado.

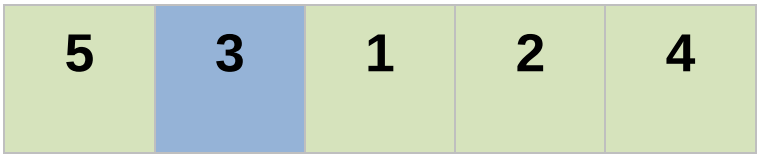
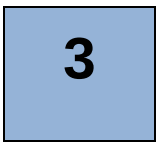
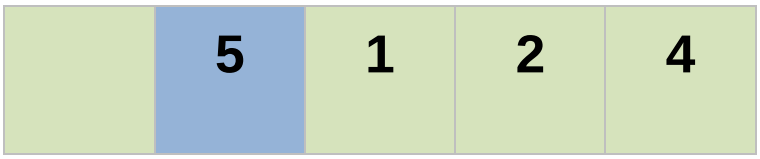


Figura 6.4: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Precisamos, agora, comparar nossa chave com os valores da esquerda. Comparamos o valor correspondente do elemento na posição 0 (que possui o valor de 5), com o elemento da nossa chave (possui o valor 3) e percebemos que o valor é menor. Logo precisamos realizar a movimentação do nosso elemento (5) para a direita.





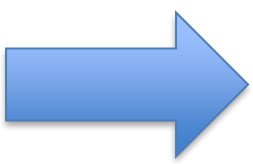


Figura 6.5: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Com isso passaremos a ter o seguinte vetor (uma vez que não nenhum valor a esquerda além do valor de número 5). Podemos realizar a modificação de posições.

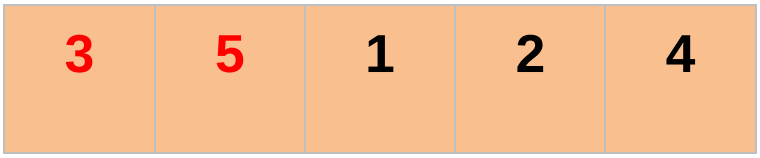


Figura 6.6: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

O item que desejamos inserir (bem como, ordenar) agora é elemento que possui o valor de 1 (a partir dessa chave faremos um processo comparativo com os demais elementos da nossa lista).

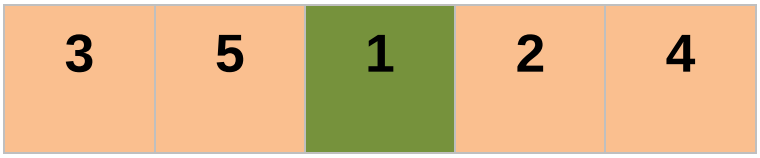


Figura 6.7: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Precisamos, agora, comparar nossa chave com os valores da esquerda. Comparamos o valor correspondente do elemento na posição 1 (que possui o valor de 5 e é imediatamente a esquerda), com o elemento da nossa chave (possui o valor 1) e percebemos que o valor é menor. Logo precisamos realizar a movimentação do nosso elemento (5) para a direita.

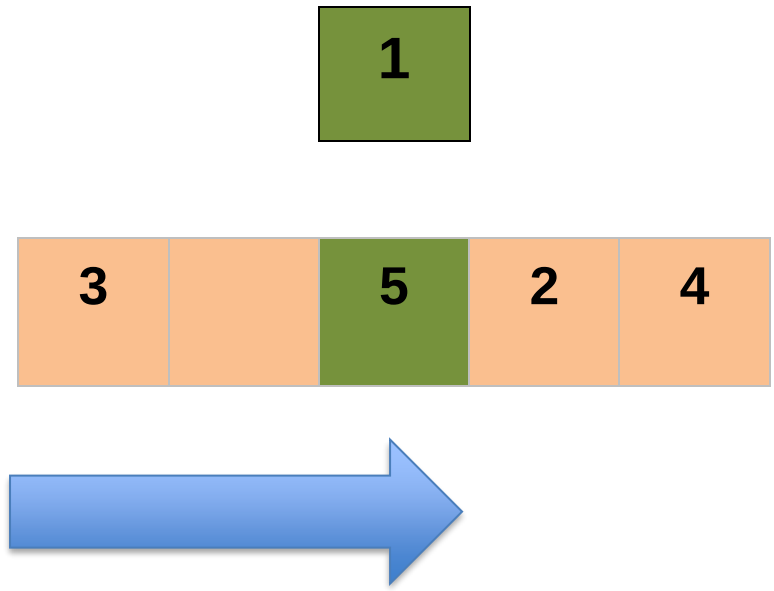


Figura 6.8: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Com isso passaremos a ter a seguinte situação:

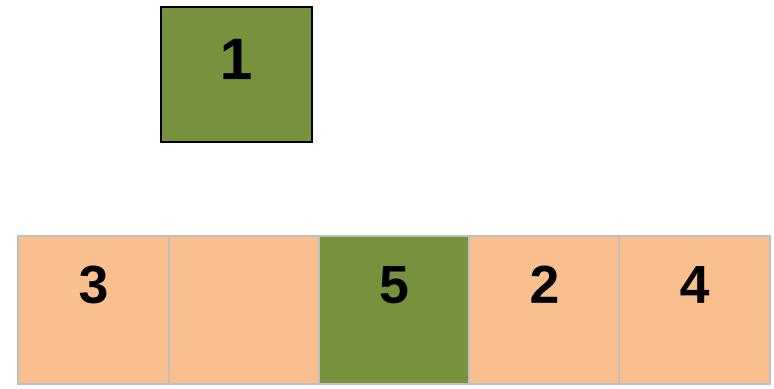


Figura 6.9: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Precisamos, agora, comparar nossa chave com os valores da esquerda. Comparamos o valor correspondente do elemento na posição 0 (que possui o valor de 3), com o elemento da nossa chave (possui o valor 3) e percebemos que o valor é menor. Logo precisamos realizar a movimentação do nosso elemento (1) para a direita.

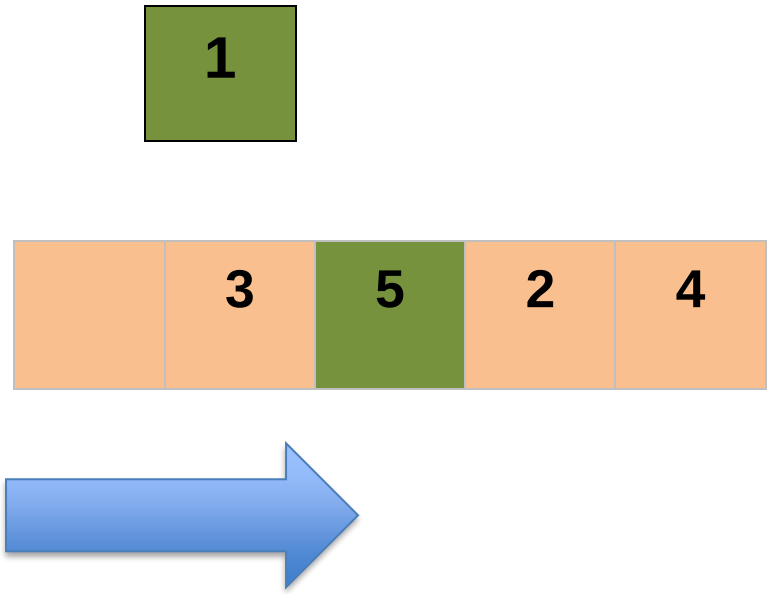


Figura 6.11: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Com isso passaremos a ter o seguinte vetor (uma vez que não nenhum valor a esquerda). Podemos realizar a modificação de posições.

Figura 6.12: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

O item que desejamos inserir agora é elemento que possui o valor de 2 (a partir dessa chave faremos um processo comparativo com os demais elementos da nossa lista).

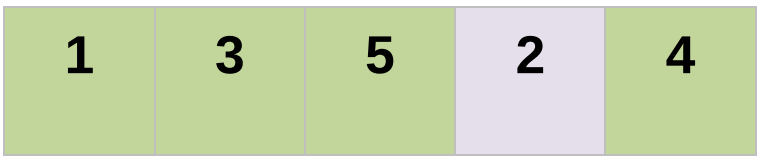


Figura 6.13: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Precisamos, agora, comparar nossa chave com os valores da esquerda. Comparamos o valor correspondente do elemento na posição 2 (que possui o valor de 5 e é imediatamente a esquerda), com o elemento da nossa chave (possui o valor 2) e percebemos que o valor é menor. Logo precisamos realizar a movimentação do nosso elemento (2) para a direita.

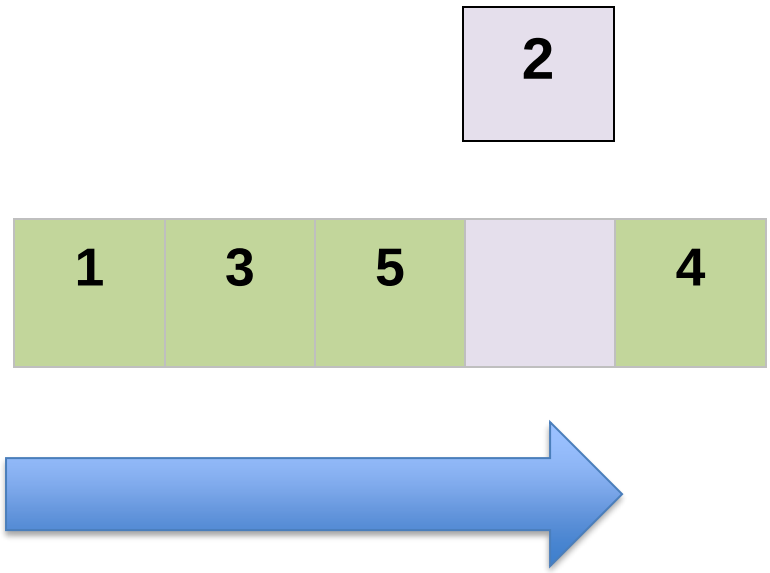


Figura 6.14: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Com isso passaremos a ter a seguinte situação:

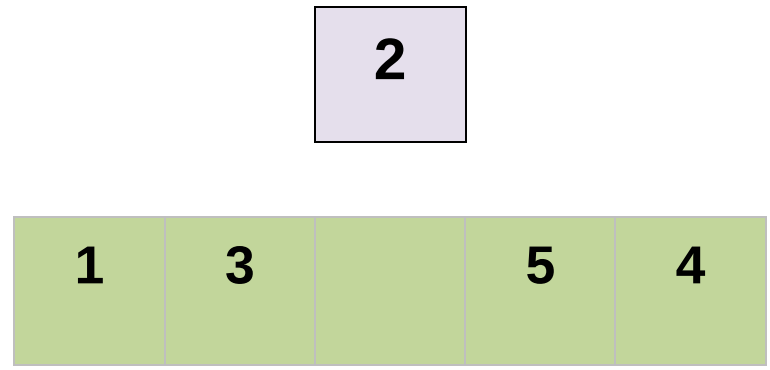


Figura 6.15: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

E continuamos com nossa comparação. Sempre considerando nossa chave principal com os valores da esquerda do nosso vetor.

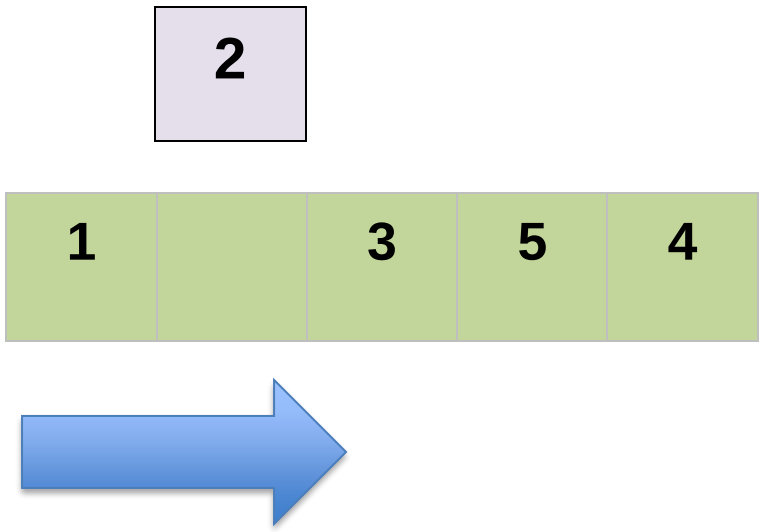


Figura 6.16: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Com isso passaremos a ter o seguinte vetor (uma vez que não nenhum valor a esquerda). Logo podemos realizar a modificação de posições.

Figura 6.17: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Por fim, nós temos, agora, o valor de número 4 que será nossa chave final. Vamos realizar as comparações do valor desta chave com os valores imediatos a esquerda dele.

Precisamos, agora, comparar nossa chave com os valores da esquerda. Comparamos o valor correspondente do elemento na posição 3 (que possui o valor de 5 e é imediatamente a esquerda), com o elemento da nossa chave (possui o valor

4) e percebemos que o valor é menor. Logo precisamos realizar a movimentação do nosso elemento (4) para a direita.

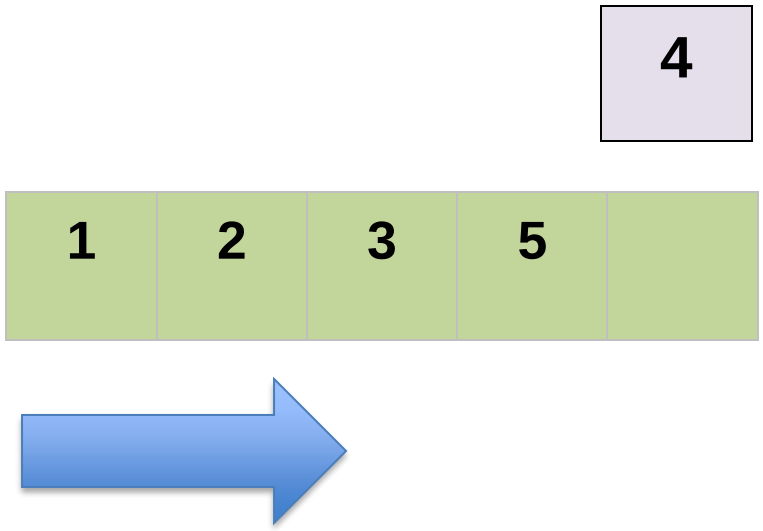


Figura 6.18: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Com isso passaremos a ter a seguinte situação:

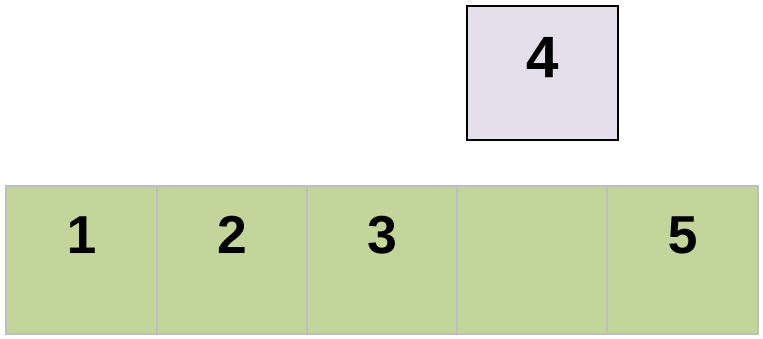


Figura 6.19: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Como o elemento a esquerda é menor que o elemento da nossa chave (elemento da esquerda possui o valor de 3 e o elemento da chave possui o valor de 4) podemos finalizar nosso algoritmo.

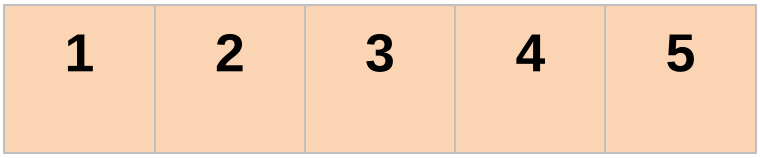


Figura 6.20: Execução do algoritmo Insertion Sort.

Fonte: Elaborado pelo Autor, 2021.

Caro(a) aluno(a), com isso finalizamos mais um capítulo. Aqui entendemos o funcionamento de um algoritmo de grande relevância, que é o *insertion sort*. No próximo capítulo começaremos a estudar sobre os algoritmos de pesquisa que são crucias na análise e busca de dados.