**Análise dos algoritmos de ordenação conforme utilização na questão 4.**

* **Insertion sort:** É um dos métodos de ordenação mais simples, sua forma de ordenação torna-se semelhante a de um baralho. Compara a chave (x) com os elementos a sua esquerda, ` deslocando para direita cada elemento maior do que a chave, depois, Insere a chave na posição correta a sua esquerda, onde os elementos já estão ordenados, e então, Repete os passos anteriores atualizando a chave para a próxima posição a direita até o fim do vetor. Na questão 4 ao testa-lo, ele apresentou no compilador 4 números que indicavam “ms”, 0, 15, 78, 339. Dentre os outros algoritmos de ordenação, ele foi o segundo mais rápido, perdendo só para o quick sort.
* **Bubble sort:** É um algoritmo simples, útil para ordenação de vetores pequenos. Ele Compara o primeiro elemento com o segundo. Se estiverem desordenados, então efetua a troca de posição, e então, Compara o segundo elemento com o terceiro e efetua a troca de posição, se necessário, repete a operação anterior até que o penúltimo elemento seja comparado com o último. Ao final desta repetição o elemento de maior valor estará em sua posição correta, a n-ésima posição do vetor, e por fim, continua a ordenação posicionando o segundo maior elemento, o terceiro..., ate que todo o vetor esteja ordenado. Quando testei esse algoritmo de ordenação na questão 4, ele ficou em penúltimo lugar, os números em “ms” foram: 0, 31, 102, 359.
* **Quick sort:** É um método de algoritmo de ordenação mais rápido que existe, e não é um algoritmo instável. Basicamente sua operação pode ser resumida na seguinte estratégia: divide sua lista de entrada em duas sub-listas a partir de um pivô, para em seguida realiza o mesmo procedimento nas duas listas menores até uma lista unitária. Dentre todos os algoritmos que testei na questão 4, esse foi o que me deu resultado em “ms” menores, ou seja, o mais rápido, os números foram: 0, 31, 78, 327.
* **Merge sort:** É um algoritmo de ordenação que sua estratégia é “dividir para conquistar”, para resolver seus problemas. Ele divide cada pedaço, em pedaço menores, resolve cada pedaço, e depois junta eles. Para juntar as partes ordenadas os dois elementos de cada parte são separados e o menor deles é selecionado e retirado de sua parte. Em seguida os menores entre os restantes são comparados e assim se prossegue até juntar as partes. Dentre os outros algoritmos de ordenação, esse foi o que me deu os números em “ms” mais altos: 0, 16, 78, 343.
* Entre o Quick sorte, Insertion sort e Merge sort, percebi que na compilação de ambos três, o terceiro número em “ms” dado foi 78, só o Bubble sort que deu um número alto que foi 102ms. E entre todos o Quicksort foi o de mais rápida compilação.