Ordenação Interna: Algoritmos de Troca

Considere a implementação em C da função classificação por trocas *QuickSort* para resolver os exercícios a seguir.

```
#define MAX 10

void quickSort(int *vetor, int esq, int dir)
{
  int i, j;
  int pivo, troca;

  i=esq; j=dir;
  pivo=vetor[(esq+dir)/2];
  printf("Pivo%d \n", pivo);

  do {
    while(vetor[i]<pivo && i<dir) i++;
    while(pivo<vetor[j] && j>esq) j--;

    if(i<=j) {
      troca = vetor[i];
      vetor[i] = vetor[j];
      vetor[j] = troca;
      i++; j--;
     }
    }while(i<=j);

    if(esq<j) quickSort(vetor, esq, j);
    if(i<dir) quickSort(vetor, i, dir);
}

int main()
{
    int i, vetor[]= {7, 5, 6, 9, 10, 2, 1, 3, 4, 8} ;
    quickSort(vetor, 0, MAX-1);
}
```

01. Mostre as etapas de funcionamento do *QuickSort* para ordenar as chaves (apresente os valores assumidos pelo pivô em toda execução do algoritmo):

`vetor = {17, 21, 9, 3, 11, 19, 15, 18, 20}`

quando se escolhe o elemento do meio como pivô: **`pivo = vetor[esq + dir] / 2`**

02. Mostre como o **`vetor = {1, 2, 1, 2, 1, 2, 1}`** é particionado quando se escolhe o elemento do meio como **`pivo = vetor[esq + dir] / 2`**. Mostre apenas os valores do pivô em cada uma das recursões.

03. O algoritmo *QuickSort* é **estável**? *Lembrando que um algoritmo de ordenação estável é aquele que não altera a ordem relativa das chaves iguais.*

( ) Sim ( ) Não

04. Qual o melhor caso?

05. Qual o pior caso?

06. Qual é o caso médio?

07. Compare o pior caso com o caso médio. Existe muita diferença entre esses dois casos?

08. O que acontece com a execução do *QuickSort* quando todos os elementos do vetor possuem o mesmo valor?

09. Modifique o algoritmo *QuickSort* (apresentado acima) para que ele ordene os valores em ordem decrescente ao invés de crescente.