AED2 - Lista 4
Ordenação

Seguem alguns exercícios relacionados com ordenação.

1 - A seguinte solução do problema da intercalação está correta? Quais os invariantes do while? (Observe que a função faz a intercalação in loco, ou seja, sem usar vetor auxiliar.) Qual o consumo de tempo?

```
int i, k, x;
i = p;
while (i < q && q < r) {
   if (v[i] >= v[q]) {
      x = v[q];
      for (k = q - 1; k >= i; --k)
         v[k+1] = v[k];
      v[i] = x;
      ++q; }
   ++i; }
```

2 - Considere a seguinte implementação do mergeSort.

```
void mergeSort (int p, int r, int v[]) {
   if (p < r-1) {                // 1
      int q = (p + r)/2;         // 2
      mergeSort (p, q, v);       // 3
      mergeSort (q, r, v);       // 4
      intercala (p, q, r, v);    // 5
   }
}
```

a) O que acontece se trocarmos (p < r-1) por (p < r) na linha // 1? A função termina? Por que?

b) O que acontece se trocarmos (p + r)/2 por (p + r - 1)/2 na linha // 2? Os subproblemas sempre diminuem? Por que?

c) O que acontece se trocarmos (p + r)/2 por (p + r + 1)/2 na linha // 2? Os subproblemas sempre diminuem? Por que?

4 - Escreva uma versão recursiva do algoritmo mergeSort que rearranje um vetor v[p..r-1] em ordem decrescente. Será preciso reescrever o algoritmo da intercalação.

5* - Digamos que um vetor v[p..r] está arrumado se existe j em p..r tal que v[p..j-1] <= v[j] < v[j+1..r]. Escreva um algoritmo que decida se v[p..r] está arrumado. Em caso afirmativo, o seu algoritmo deve devolver o valor de j.

6 - Critique a seguinte variante da função separa. Quais os invariantes?

```
int separa (int v[], int p, int r) {
   int c = v[p], i = p+1, j = r, t;
   while (i <= j) {
      if (v[i] <= c) ++i;
      else {
         t = v[i], v[i] = v[j], v[j] = t;
         --j; } }
   v[p] = v[j], v[j] = c;
   return j; }
```

7 - Considere a seguinte implementação do quickSort.

```
void quicksort (int v[], int p, int r) {
   int j;                        // 1
   if (p < r) {                  // 2
      j = separa (v, p, r);      // 3
      quicksort (v, p, j-1);     // 4
      quicksort (v, j+1, r);     // 5
   }
}
```

a) Que acontece se trocarmos p < r por p != r na linha 2? A função termina? Por que?

b) Que acontece se trocarmos j-1 por j na linha 4? Os subproblemas sempre diminuem? Por que?

c) Que acontece se trocarmos j+1 por j na linha 5? Os subproblemas sempre diminuem? Por que?

8 - Escreva uma função que decida se um vetor v[0..m-1] é ou não um max-heap.

9* - Suponha que v[1..2^k - 1] é um max-heap. Mostre que mais da metade dos elementos de v está na última "camada" do max-heap, ou seja, em v[2^(k-1)..2^k -1].

10 - Escreva uma função eficiente que rearranje um vetor arbitrário de modo a transformá-lo em um max-heap. Sugestão: use a função desceHeap.

Para revisar conceitos sobre ordenação e encontrar mais exercícios acesse:

- https://www.ime.usp.br/~pf/algoritmos/aulas/mrgsrt.html
- https://www.ime.usp.br/~pf/algoritmos/aulas/quick.html
- https://www.ime.usp.br/~pf/algoritmos/aulas/hpsrt.html