

Discente: Ronaldo Ribeiro Porto Filho – 202410131

Data: 06/09/2025

Exercício prova de correção do algoritmo Shakesort

Código:

```
1 void shakesort(int *vet, int tamanho){
2     int aux;
3     int esq = 0, dir = tamanho-1;
4     while(dir > esq){
5         for (int i = esq; i < dir; i++){
6             if (vet[i] > vet[i+1]){
7                 aux = vet[i+1];
8                 vet[i+1] = vet[i];
9                 vet[i] = aux;
10            }
11        }
12        for (int i = dir-1; i > esq; i--){
13            if (vet[i] < vet[i-1]){
14                aux = vet[i-1];
15                vet[i-1] = vet[i];
16                vet[i] = aux;
17            }
18        }
19        esq++;
20        dir--;
21    }
22    return;
23 }
```

Primeira parte:

Base: Considera-se que o sub vetor formado apenas pela última posição do vetor Vet esteja ordenado, por ser unitário ($k = 1$).

Passo Indutivo: Considera-se que as K últimas posições do vetor Vet estejam ordenadas antes da linha 4. Então, sendo N o tamanho do vetor, $Vet[N - K] \leq \dots \leq Vet[N - 2] \leq Vet[N - 1]$, sendo estes também os maiores elementos do vetor Vet. Por conta da lógica das linhas 6 até 10, o maior elemento do sub vetor que vai da posição 0 até $(N - (K + 1))$ estará na posição $(N - (K + 1))$. Pela hipótese inicial (Hipótese Indutiva), todas as K últimas posições do vetor Vet estão ordenadas, ou seja, um elemento x ocupará a posição $Vet[N - (K + 1)]$, sendo que x é menor ou igual a todos os elementos à sua direita, e maior que todos os

elementos à sua esquerda. Logo, $\text{Vet}[N - (K + 1)] \leq \text{Vet}[N - K] \leq \dots \leq \text{Vet}[N - 2] \leq \text{Vet}[N - 1]$.

Conclusão: Provou-se que o último elemento do vetor é um sub vetor ordenado, por ser unitário ($K = 1$). Além disso, supondo-se que os K últimos elementos do vetor estejam ordenados, antes da linha 4, provou-se que os $(K + 1)$ últimos elementos ficam ordenados na próxima interação do laço. Logo, o laço while na linha 4 permite o vetor ficar ordenado para $K \geq 1$. Como $N \geq 1$, o algoritmo termina com o vetor ordenado, juntamente com a segunda parte da prova.

Segunda parte:

Base: Considera-se que o sub vetor formado apenas pela primeira posição do vetor Vet esteja ordenado, por ser unitário ($k = 1$).

Passo Indutivo: Considera-se que as K primeiras posições do vetor Vet estejam ordenadas antes da linha 12. Então, sendo $(N - 1)$ o novo tamanho do vetor, pois o segundo laço de interação desconsidera a última posição por já estar ordenada, $\text{Vet}[K - 1] \geq \dots \geq \text{Vet}[1] \geq \text{Vet}[0]$, sendo estes também os menores elementos do vetor Vet. Pela hipótese inicial (Hipótese Indutiva), todas as K primeiras posições do vetor Vet estão ordenadas, ou seja, um elemento x ocupará a posição $\text{Vet}[K]$, sendo que x é maior ou igual a todos os elementos à sua esquerda, e menor que todos os elementos à sua direita, por conta da lógica das linhas 13 a 17. Ou seja, o menor elemento do sub vetor desordenado estará na posição K . Logo, $\text{Vet}[K] \geq \text{Vet}[K - 1] \geq \dots \geq \text{Vet}[1] \geq \text{Vet}[0]$. Ademais, a posição 0 do vetor será substituída pela variável “esq”, pois o laço while reduz as posições já ordenadas, permitindo maior eficiência nas próximas interações dos outros laços.

Conclusão: Provou-se que o primeiro elemento do vetor é um sub vetor ordenado, por ser unitário ($K = 1$). Além disso, supondo-se que os K primeiros elementos do vetor estejam ordenados, antes da linha 12, provou-se que os $(K + 1)$ primeiros elementos ficam ordenados na próxima interação do laço. Logo, o laço while na linha 4 permite o vetor ficar ordenado para $K \geq 1$. Como $N \geq 1$, o algoritmo termina com o vetor ordenado, juntamente com a primeira parte da prova.

Conclusão Geral: Provou-se, a partir dessas duas provas de correção usando indução matemática, que o algoritmo do Shakesort ordena um vetor de qualquer tamanho. Por conta do laço while da linha 4, o principal laço de repetição, ele executa os dois laços “for” $N/2$ vezes, aproximadamente, pois a cada vez o algoritmo desconsidera a primeira e a última posição, por já estarem com os valores corretos, e assim, interage com os outros laços novamente até ordenar por completo todos os elementos.