Universidade Federal de Paraíba
Centro de Informática
Curso de Ciência da Computação
Analise e Projeto de Algoritmos

Casos de melhor complexidade dos algoritmos: Insertion sort, Quicksort, Mergesort e Heapsort

Thulio Guilherme Silva de Amorim
Agosto/2016
João Pessoa

Casos de melhor complexidade dos algoritmos: Insertion sort, Quicksort, Mergesort e Heapsort

Insertion sort

O Insertion sort apresenta melhor complexidade quando a entrada já se encontra ordenada. Neste caso, o Insertion sort apenas realiza uma passagem pela entrada para validação, fazendo com que o custo seja **O(*n*)**. Se existe apenas poucos elementos desordenados, a complexidade do algoritmo chega próximo a **O(*n*)**.

Mergesort

O Mergesort apresenta complexidade constante, sendo **O(*n*log n*)**, pelo fato de realizar a mesma quantidade de comparações e de trocas para qualquer em relação ao tamanho da entrada. O algoritmo divide a entrada em duas partes de mesmo tamanho até que todas as partes apresentem apenas um elemento. Depois disso é realizado o junção dos elementos. A complexidade do algoritmo pode ser obtida pela formula: **T(*n*)** = **T(*n/2*)** + **T(*n/2*)** + **n** = **2*T(*n/2*)** + ***n***, que é equivalente a **O(n*log n)**.

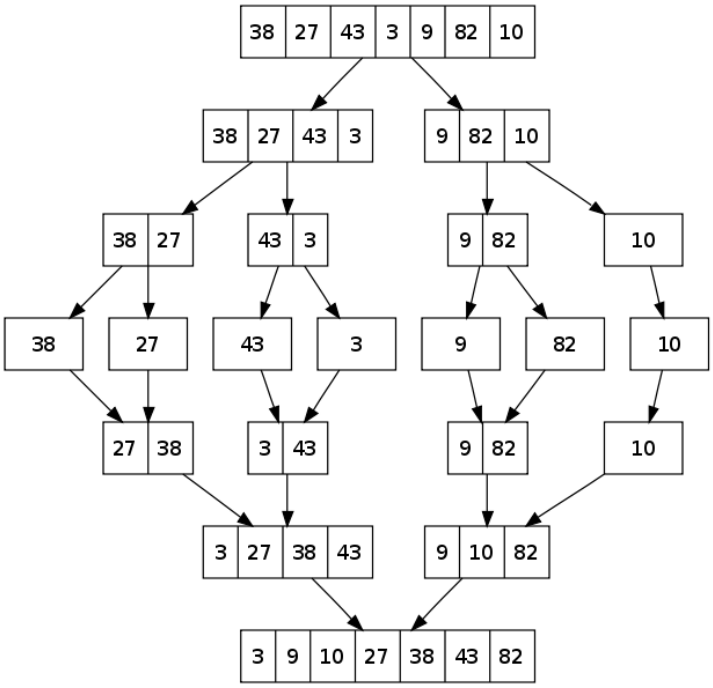

Figura 1: Exemplo de execução do algoritmo Mergesort

Quicksort

Já no caso do Quicksort, o algoritmo apresenta melhor complexidade quando toda a divisão durante a execução do algoritmo gera apenas vetores de mesmo tamanho. Como mostra a figura abaixo. A complexidade pode ser obtida a partir da formula: **T(n)** = **T(n/2)** + **T(n/2)** + **n** = **2*T(n/2)**, logo a complexidade no melhor caso é igual a **O(*n*log n*)**.

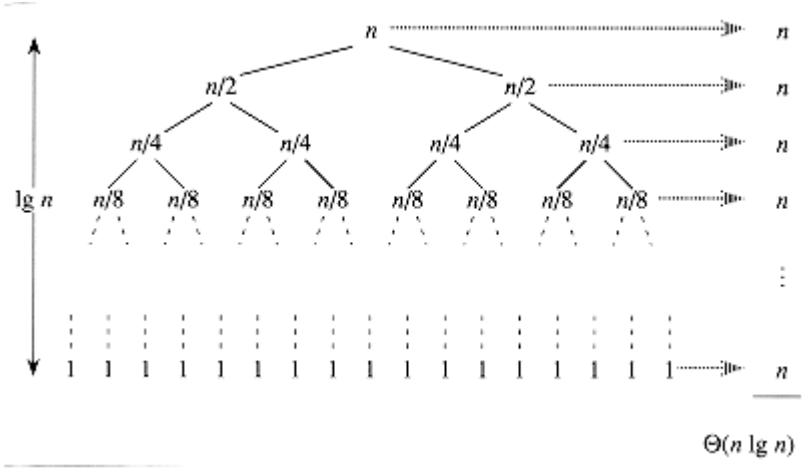

Figura 2: Representação do número de chamadas recursivas e o processamento realizado no melhor caso do Quicksort

Heapsort

No caso do Heapsort, ele apresenta complexidade **O(*n*log n*)** para qualquer entrada. Sobre a entrada é realizada uma operação para que o vetor possua a propriedade Heap, sendo o custo desta operação **O(*log n)***. Após a retirada do maior elemento, ou menor, é necessário reestabelecer a propriedade Heap do vetor originado pela remoção do elemento, realizando novamente a operação. Este procedimento é realizado ***n*** vezes, logo a complexidade do Heapsort é **O(*n*log n*)**. Os algoritmos de Heapsort e Mergesort não possuem nenhuma vantagem ou desvantagem que pode ser ocasionada a partir da entrada. Por isso possuem complexidade constante.

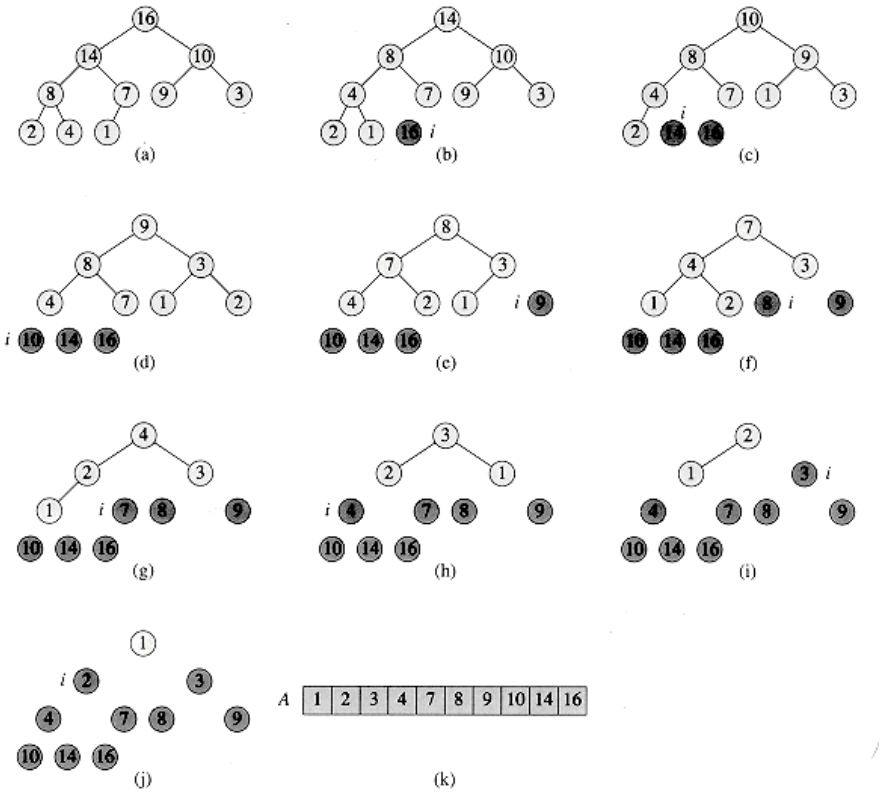

Figura 3: Exemplo de execução do algoritmo Heapsort