Ciência da Computação
Projeto e análise de algoritmos
Strand Sort

Danillo Rodrigues Abreu[1]

[1]Universidade Federal de Alagoas – UFAL
Campus Arapiraca

2019

Roteiro

1. Introdução
   - O que são algoritmos de ordenação?
   - Ordem numérica
   - Ordem lexicográfica
2. Algoritmo Strand Sort
   - Caracteristicas
   - Como funciona?
3. Referência

Roteiro

- Em ciência da computação, algoritmo de ordenação é um algoritmo que coloca os elementos de uma dada sequência em uma certa ordem.
- As ordens mais usadas são a **numérica** e a **lexicográfica**.

- **Ordem Crescente**: quando os números estão na ordem do menor para o maior.
  - Exemplo: 0 - 1 - 2 - 3 - 4 - 5
- **Ordem Decrescente**: quando os números estão na ordem do maior para o menor.
  - Exemplo: 5 - 4 - 3 - 2 - 1 - 0

- Ordem lexicográfica, também conhecida como ordem do dicionário ou ordem alfabética, é uma estrutura de ordem natural do produto cartesiano de dois conjuntos ordenados.
- Dados dois conjuntos parcialmente ordenados **A** e **B**, a ordem lexicográfica sobre o produto cartesiano **A** x **B** é definida como:

Exemplo:

- (a,b) $\leq$ (a‘,b‘) se e somente se a for maior que a‘ ou (a = a‘ e b $\leq$ b‘)

Roteiro

Algoritmo Strand Sort

Caracteristicas:

- **Strand Sort** é um algoritmo de classificação recursiva que classifica os itens de uma lista em ordem crescente.
- Possui complexidade de $\mathbf{O(n)}$ no melhor caso possível, que ocorre quando a entrada é uma lista que já está classificada.
- Complexidade de $\mathbf{O(n\ \sqrt{n})}$ no caso médio.
- E complexidade de $\mathbf{O(}n^2\mathbf{)}$ no pior caso, que ocorre quando a lista de entrada está ordenada em ordem inversa.

Algoritmo Strand Sort

Caracteristicas:

- O algoritmo **Strand Sort** não é **in-place**, pois sua complexidade de espaço é **O(n)**.
- O **Strand Sort** é mais útil para dados que são armazenados em uma lista vinculada, devido às frequentes **inserções** e **remoções** de dados.
- Usando uma outra estrutura de dados, como um array, aumenta consideravelmente o tempo de execução e a complexidade do algoritmo, devido às longas inserções e remoções.
- É estável.

Como funciona?

Passo a passo

- O algoritmo recebo como entrada uma lista qualquer e a ordena em ordem **crescente**.
- Exemplo de uma **lista desordenada**: $\{5, 2, 4, 1, 9\}$

Como funciona?

Passo a passo

- **Sequência do Algoritmo**:
  1. Mova o primeiro elemento da **lista desordenada** para uma nova **sub-lista**.
     - **Lista desordenada**: {2, 4, 1, 9}
     - **Sub-lista**: {5}
  2. Agora percorra a **lista desordenada** e compare cada elemento com 5 até que cruze com um elemento maior que 5, e logo encontra-la o 9.
  3. Mova o 9 para a **sub-lista** na ordem após o 5.
     - **Lista desordenada**: {2, 4, 1}
     - **Sub-lista**: {5, 9}

4. Agora compare 9 com os elementos restantes na **lista desordenada** até que haja um número maior que 9.
5. Como não há elemento maior que o 9 na **lista desordenada**, crie uma nova lista, chamada: **Lista ordenada**, e mova os elementos da **sub-lista** para ela.
   - **Lista desordenada**: $\{2, 4, 1\}$
   - **Sub-lista**: $\{\}$
   - **Lista ordenada**: $\{5, 9\}$

6. Como a **sub-lista** está vazia, então mova o primeiro elemento da **lista desordenada** para a **sub-lista**.
   - **Lista desordenada**: {4, 1}
   - **Sub-lista**: {2}
   - **Lista ordenada**: {5, 9}
7. Compare o resto dos elemento da **lista desordenada** com o 2 até encontra um elemento maior que ele, com isso irá encontrar o 4, então o 4 também é movido para a **sub-lista** após o 2.
   - **Lista desordenada**: {1}
   - **Sub-lista**: {2, 4}
   - **Lista ordenada**: {5, 9}

8. Agora compare 4 com os elementos restantes na **lista desordenada** até que haja um número maior que ele.
9. Como não tem nenhum elemento maior que 4 na **lista desordenada**, então os elementos da **sub-lista** são movidos para a **lista ordenada**.
   - **Lista desordenada**: $\{1\}$
   - **Sub-lista**: $\{\}$
   - **Lista ordenada**: $\{2, 4, 5, 9\}$

10. Como a **sub-lista** está vazia, mova o primeiro elemento da **lista desordenada** para a **sub-lista**.
    - **Lista desordenada**: {}
    - **Sub-lista**: {1}
    - **Lista ordenada**: {2, 4, 5, 9}
11. Agora que a lista **desordenada** está vazia, a **sub-lista** é movida para a **lista ordenada**.
    - **Lista desordenada**: {}
    - **Sub-lista**: {}
    - **Lista ordenada**: {1, 2, 4, 5, 9}

12. Com isso, não há mais elementos na lista **desordenada** e todos os elementos na **lista ordenada** foram classificados com êxito em **ordem numérica crescente**.
    - **Resultado final**: $\{1, 2, 4, 5, 9\}$

Roteiro

Referência

- Paul E. Black.
  Strand sort.
  https://www.nist.gov/dads/HTML/strandSort.html.
  Dictionary of Algorithms and Data Structures [online], 24 de novembro de 2008.
- Michael T. Goodrich e Roberto Tamassia.
  Projetos de Algoritmos.
  Bookman, 2004.
- I C Gupta; Deepak Jaroliya; Prestige Institute of Management and Research.
  IT enabled practices and emerging management paradigms.
  Indore: Prestige Institute of Management and Research, 2008.

Obrigado!