Algoritmos de Ordenação (Parte 2)

Prof. Jefferson T. Oliva

Algoritmos e Estrutura de Dados I (AE22CP)
Engenharia de Computação
Departamento Acadêmico de Informática (Dainf)
Universidade Tecnológica Federal do Paraná (UTFPR)
Campus Pato Branco

DAINF
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
CÂMPUS PATO BRANCO

UTFPR
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Sumário

- Ordenação por Seleção
  - Select sort
  - Heap sort

Ordenação por Seleção

- Seleciona o maior ou o menor elemento do conjunto para cada iteração para colocá-lo em sua devida posição

- Exemplos de algoritmos de ordenação por seleção
  - *Select sort*
  - *Heap sort*

Ordenação por Seleção

Select sort

- Ideia básica
  1. Selecionar o maior elemento do conjunto
  2. Trocá-lo com o último elemento
  3. Repetir os dois passos anteriores com os $n-1$ elementos restantes, após com os $n-2$ e assim por diante até sobrar o primeiro elemento que será o menor do conjunto

Ordenação por Seleção

Select sort

- Implementação

```
void selectsort(int v[], int n){
  int i, j, p, aux;

  for (i = n - 1; i >= 1; i--){
    p = i;

    for (j = 0; j < i; j++)
       if (v[j] > v[p])
          p = j;

    aux = v[i];
    v[i] = v[p];
    v[p] = aux;
  }
}
```

Ordenação por Seleção

Select sort

- Exemplo

i	p	X[0]	X[1]	X[2]	X[3]	X[4]	X[5]	X[6]	X[7]
-	-	25	57	48	37	12	92	86	33
7	5	25	57	48	37	12	33	86	92
6	6	25	57	48	37	12	33	86	92
5	1	25	33	48	37	12	57	86	92
4	2	25	33	12	37	48	57	86	92
3	3	25	33	12	37	48	57	86	92
2	1	25	12	33	37	48	57	86	92
1	0	12	25	33	37	48	57	86	92

Ordenação por Seleção

Select sort

- Complexidade do método: $O(n^2)$
- Esse método de ordenação é estável
- Simples implementação
- Um dos algoritmos mais rápidos para a ordenação de vetores pequenos
- Em vetores grandes, esse algoritmo é um dos mais lentos

Ordenação por Seleção

Select sort

- Links interessantes:
  - Dança cigana:
    `https://www.youtube.com/watch?v=Ns4TPTC8whw`
  - Simulador gráfico do *select sort*:
    `https://visualgo.net/bn/sorting`

Ordenação por Seleção

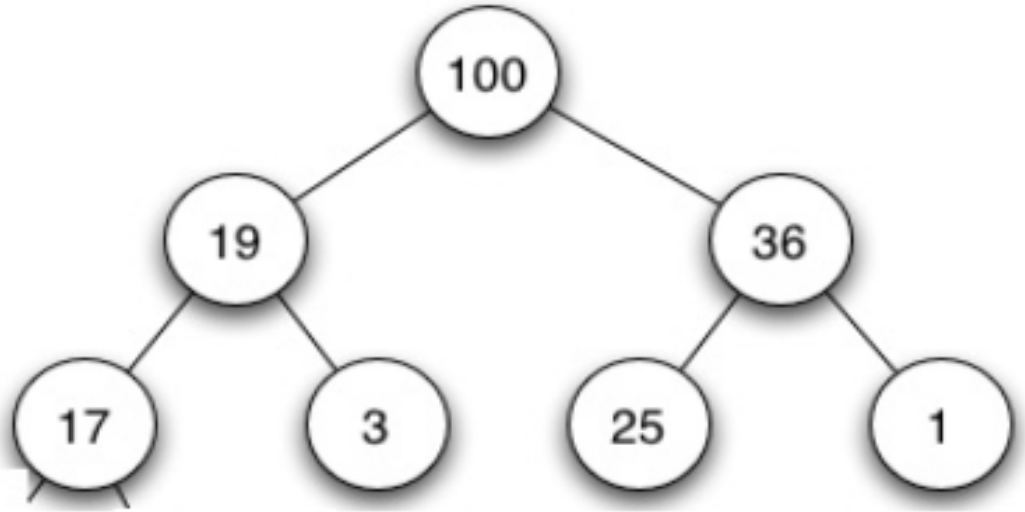
Heap sort

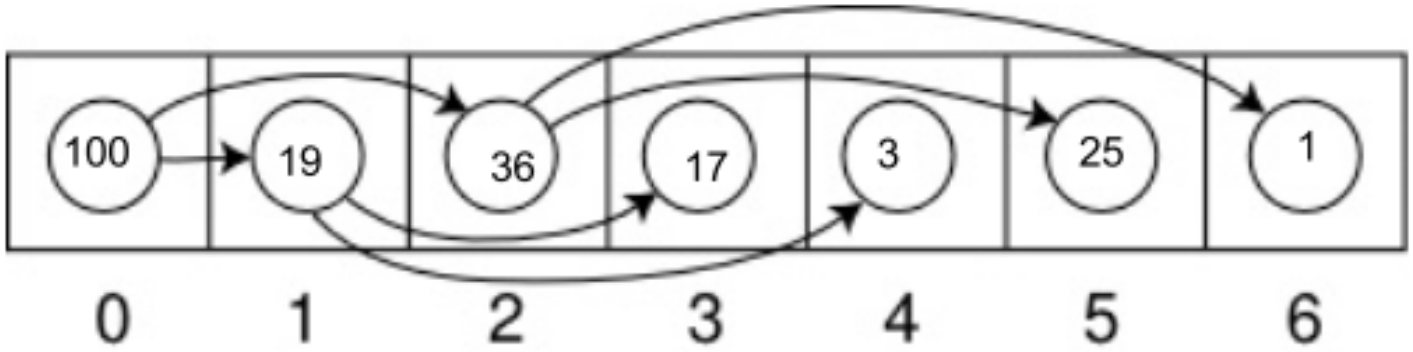
- Baseado no princípio de ordenação por seleção em árvore binária

- O método consiste em duas fases distintas:
  1. Montagem da árvore binária (HEAP)
  2. Seleção dos elementos na ordem desejada

Ordenação por Seleção

Heap sort





- Em uma *heap*
  - O sucessor à esquerda do elemento de índice i é o elemento de índice $2 * (i + 1) - 1$, se $2 * (i + 1) - 1 < n$, caso contrário não existe
  - O sucessor à direita do elemento de índice i é o elemento de índice $2 * (i + 1)$, se $2 * (i + 1) < n$, caso contrário não existe

Ordenação por Seleção

Heap sort

- Código para rearranjar um vetor para que o mesmo atenda a condição para ser uma *heap*

```
void gerarHeap(int v[], int n){
  int esq = n / 2;

  while (esq >= 0){
    refazer(v, esq, n - 1);
    esq--;
  }
}
```

Ordenação por Seleção

Heap sort

- Código para rearranjar um vetor para que o mesmo atenda a condição para ser uma *heap* (continuação)

```
void refazer(int v[], int esq, int dir){
  int j = (esq + 1) * 2 - 1;
  int x = v[esq];

  while (j <= dir){
    if ((j < dir) && (v[j] < v[j + 1]))
       j++;

    if (x >= v[j])
       break;

    v[esq] = v[j];
    esq = j;
    j = (esq + 1) * 2 - 1;
  }

  v[esq] = x;
}
```

- Função principal

```
void heapsort(int v[], int n){
  int x;
  int dir = n - 1;

  gerarHeap(v, n);

  while (dir > 1){
    x = v[0];
    v[0] = v[dir];
    v[dir] = x;
    dir--;
    refazer(v, 0, dir);
  }
}
```

Ordenação por Seleção

Heap sort

- Exemplo

iteração	x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
fazheap: i=3	25	57	48	37	12	92	86	33
fazheap: i=2	25	57	92	37	12	48	86	33
fazheap: i=1	25	57	92	37	12	48	86	33
fazheap: i=0	92	57	86	37	12	48	25	33
heapsort: i=7	86	57	48	37	12	33	25	92
heapsort: i=6	57	37	48	25	12	33	86	92
heapsort: i=5	48	37	33	25	12	57	86	92
heapsort: i=4	37	25	33	12	48	57	86	92
heapsort: i=3	33	25	12	37	48	57	86	92
heapsort: i=2	25	12	33	37	48	57	86	92
heapsort: i=1	12	25	33	37	48	57	86	92

Ordenação por Seleção

Heap sort

- À primeira vista, parece que o *heap sort* não apresenta bons resultados
- Não é um algoritmo de ordenação estável
- O algoritmo não é recomendado para pequenos conjuntos de elementos
- Custo de tempo (melhor, médio e pior caso): $O(n \log_2(n))$

- Exercício: aplicar o *heap sort* em um arranjo de 10 elementos:
  - Organizado em ordem crescente
  - Organizado em ordem decrescente

Ordenação por Seleção

Heap sort

- Links interessantes:
  - Dança húngara: `https://www.youtube.com/watch?v=Xw2D9aJRBY4&list=RDCMUCIqiLefbVHsOAXDAxQJH7Xw&index=10`
  - Simulador gráfico do *heap sort*: `https://www.cs.usfca.edu/~galles/visualization/HeapSort.html`

Referências I


Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C.
Introduction to Algorithms.
Third edition, The MIT Press, 2009.


Horowitz, E., Sahni, S. Rajasekaran, S.
Computer Algorithms.
Computer Science Press, 1998.


Rosa, J. L. G.
Métodos de Ordenação. SCE-181 – Introdução à Ciência da Computação II.
Slides. Ciência de Computação. ICMC/USP, 2018.


Ziviani, N.
Projeto de Algoritmos - com implementações em Java e C++.
Thomson, 2007.