Algoritmos e Lógica de Programação (JavaScript): Programando algoritmos de ordenação e busca em JavaScript

Resumo

Referências Bibliográficas

- DEITEL, Paul J.; DEITEL, Harvey M. Ajax Rich. **Internet Applications e Desenvolvimento Web para programadores.** Pearson Prentice Hall: São Paulo, 2008.
- MORRISON, Michael. **Use a Cabeça! JavaScript.** 1ª edição. Rio de Janeiro: Alta Books, 2009.

Vamos ver aqui algoritmos de ordenação e busca que vamos escrever como um programa em JavaScript. Existem vários tipos diferentes de algoritmos de ordenação como o bubblesort, quicksort, mergesort, selectionsort, radixsort, dentre vários outros. Cada algoritmo de ordenação varia desde a sua simplicidade na escrita do código com menos recursos computacionais até sua complexidade com mais recursos computacionais.

Há também diversos tipos de algoritmos de busca como a busca sequencial, a busca binária, a busca com árvores binárias, a busca em árvores balanceadas, dentre vários outros. Cada algoritmo de busca varia desde a sua simplicidade até sua complexidade no desenvolvimento do programa com mais ou menos recursos computacionais e de lógica.

Ordenação em JavaScript

Um dos algoritmos de ordenação e um dos mais simples de se entender e de escrever o programa JavaScript é o Bubble Sort. O Bubble Sort é um algoritmo que precisa de duas estruturas de repetição encadeada para o seu desenvolvimento e usa o conceito de que o número maior vai flutuando até o final de uma lista ou o número menor até o início da lista.

Vamos exemplificar o funcionamento do bubble sort com uma lista com cinco números inteiros.

Bubble Sort

15	*36*	*27*	*10*	*84*	
15	*36*	*27*	*10*	*84*	*compara 15 e 36 – não efetua a troca*
15	*36*	*27*	*10*	*84*	*compara 36 e 27 – realiza a troca*
15	*27*	*36*	*10*	*84*	*compara 36 e 10 – realiza a troca*
15	*27*	*10*	*36*	*84*	*compara 36 e 84 – não realiza a troca*
15	*27*	*10*	*36*	*84*	*final do primeiro processo*
15	*27*	*10*	*36*	*84*	*compara 15 e 27 – não efetua a troca*
15	*27*	*10*	*36*	*84*	*compara 27 e 10 – efetua a troca*
15	*10*	*27*	*36*	*84*	*compara 27 e 36 – não efetua a troca*
16	*10*	*27*	*36*	*84*	*final do segundo processo*
15	*10*	*27*	*36*	*84*	*compara 15 e 10 – efetua a troca*
10	*15*	*27*	*36*	*84*	*compara 15 e 27 – não efetua a troca*
10	*15*	*27*	*36*	*84*	*final do terceiro processo*
10	*15*	*27*	*36*	*84*	*compara 10 e 15 – não efetua a troca*
10	*15*	*27*	*36*	*84*	*final do último processo*

O método de ordenação BubbleSort em JavaScript recebe uma lista com os números a serem ordenados e segue o processo acima para a ordenação dos números da lista, conforme pode ser visto no método seguinte:

```
<script>

function BubbleSort(num){
    for (i=0 ; i<num.length-1 ; i++){
        for (j=0 ; j<num.length-1-i ; j++){
            if(num[j] > num[j+1]){
                temp = num[j];
                num[j] = num[j+1];
                num[j+1] = temp;
            }
        }
    }
    return num;
}
```

Observe que já iniciamos o programa JavaScript com uma lista de números num vetor, que é passado como parâmetro ao método BubbleSort para ser ordenado. A função BubbleSort retorna a lista ordenada que é apresentada ao usuário.

```
15
16 num = [15, 36, 27, 10, 84];
17 document.write(num + "<br>");
18 ord = BubbleSort(num);
19 document.write(ord);
20
21 </script>
```

Observe os resultados da execução do programa JavaScript, que ordena uma lista de números pelo método BubbleSort. A primeira lista é dos números não ordenados que vai no parâmetro do método e a segunda lista é a dos números já ordenados pelo método BubbleSort:

Estrutura de Ordenação em JavaScript

Para praticar a estrutura de ordenação num programa JavaScript, vamos desenvolver um programa que ordena 10 números inteiros, utilizando a técnica do bubblesort. Perceba que, neste caso, o programa precisa receber dez números inteiros do usuário, armazená-los em um vetor de 10 posições inteiras, aplicar o método do BubbleSort e apresentar as informações ordenadas.

Primeiro, escrevemos o método BubbleSort que ordena os números:

```
<script>

function BubbleSort(num){
   for (i=0 ; i<num.length-1 ; i++){
      for (j=0 ; j<num.length-1-i ; j++){
         if(num[j] > num[j+1]){
            temp = num[j];
            num[j] = num[j+1];
            num[j+1] = temp;
         }
      }
   }
   return num;
}
```

Agora, desenvolvemos na parte principal do programa JavaScript, que declara um vetor de inteiros com 10 posições, recebe 10 inteiros e armazena esses elementos no vetor, utilizando uma estrutura de repetição. Depois, faz a chamada do método BubbleSort enviando o vetor com os números recebidos do usuário e, ao final, apresenta os números ordenados:

```

num = new Array();
for (x=0; x<10; x++){
   num[x] = parseInt(window.prompt("Digite um inteiro: "));
}

document.write(num + "<br>");
ord = BubbleSort(num);
document.write(ord);

</script>
```

Observe o resultado da execução deste programa JavaScript de ordenação.

Busca em JavaScript

Um dos algoritmos de busca e um dos mais simples de se entender e de desenvolver o um programa JavaScript é a busca sequencial. Na busca sequencial, o algoritmo vai passando por cada posição da lista, iniciando no início da lista, comparando se a informação da lista é a informação buscada. Caso encontre, a busca é parada e a informação é apresentada ou utilizada. Pode acontecer de percorrer a lista toda e o elemento não seja encontrado na lista.

Vamos exemplificar o funcionamento da busca sequencial com uma lista com 5 números inteiros; vamos buscar o número 10 que existe na lista.

Busca Sequencial do 10

Perceba que aqui, o número 10 a ser buscado é comparado com o primeiro elemento, depois com o segundo e depois com o terceiro. Para cada um destes elementos, a busca é um insucesso. Porém, ao comparar o quarto elemento, a busca é bem-sucedida e as informações são apresentadas:

15 36 27 10 84

compara 15 – não encontrou

compara 36 – não encontrou

compara 27 – não encontrou

compara 10 – encontrou, apresenta

Perceba que aqui: o número 20 a ser buscado é comparado com cada um dos cinco elementos da lista, até finalizar a lista e o emento 20 não é encontrado. Para cada um destes elementos, a busca é um insucesso e a lista termina, apresentando a informação de que o elemento não foi encontrado:

Busca Sequencial do 20

15 36 27 10 84

compara 15 – não encontrou

compara 36 – não encontrou

compara 27 – não encontrou

compara 10 – não encontrou

Compara 84 – não encontrou

O método do algoritmo BuscaSequencial em JavaScript recebe um número a ser buscado e uma lista com números e com os elementos a serem comparados, seguindo o processo acima para a busca de um elemento de uma lista. Como pode ser observado no programa seguinte:

```
<script>

function BuscaSequencial(n, num){
   Achou = false;
   i = 0;

   while ((i<num.length) && (!Achou)){
      if(n==num[i]){
         Achou = true;
      }
      i = i + 1;
   }

   if(Achou){
      document.write("Achou " + num[i-1] + "<br><br>");
   }
   else{
      document.write(n + " não encontrado <br>");
   }
   return num;
}
```

Observe que na parte principal do programa JavaScript, declaramos o vetor com a lista de números e a chamada do método BuscaSequencial, passando como parâmetro a lista e o número a ser buscado:

```

   if(Achou){
      document.write("Achou " + num[i-1] + "<br><br>");
   }
   else{
      document.write(n + " não encontrado <br>");
   }
   return num;
}

Lista = [15, 36, 27, 10, 84];
BuscaSequencial(10, Lista);
BuscaSequencial(20, Lista);

</script>
```

O resultado da execução do programa JavaScript pode ser visto a seguir.

Estrutura de Busca em JavaScript

Para praticar a estrutura de busca, vamos desenvolva um programa JavaScript que recebe um número e busca essa informação, em um vetor de inteiros, utilizando a técnica da Busca Sequencial.

Perceba que, neste caso, o programa precisa receber uma lista e um número fornecidos pelo usuário, aplicar o módulo de BuscaSequencial e apresentar a informação se o elemento está ou não na lista.

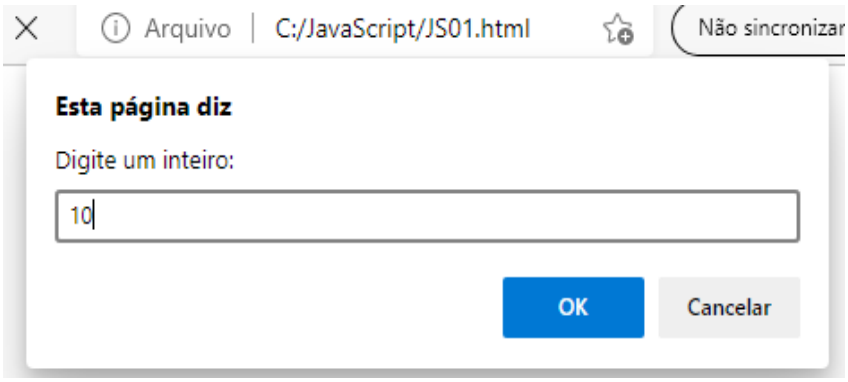
Primeiro, escrevemos o método BuscaSequencial que verifica se um elemento está numa lista:

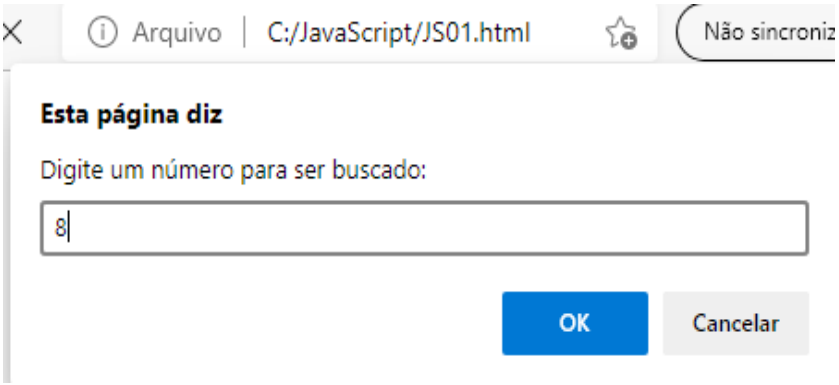
```
<script>

function BuscaSequencial(n, num){
   Achou = false;
   i = 0;

   while ((i<num.length) && (!Achou)){
      if(n==num[i]){
         Achou = true;
      }
      i = i + 1;
   }

   if(Achou){
      document.write("Achou " + num[i-1] + "<br><br>");
   }
   else{
      document.write(n + " não encontrado <br>");
   }
   return num;
}
```

Na parte principal do programa JavaScript, temos a declaração das variáveis, a entrada de dados pelo usuário e a chamada adequada do método BuscaSequencial:

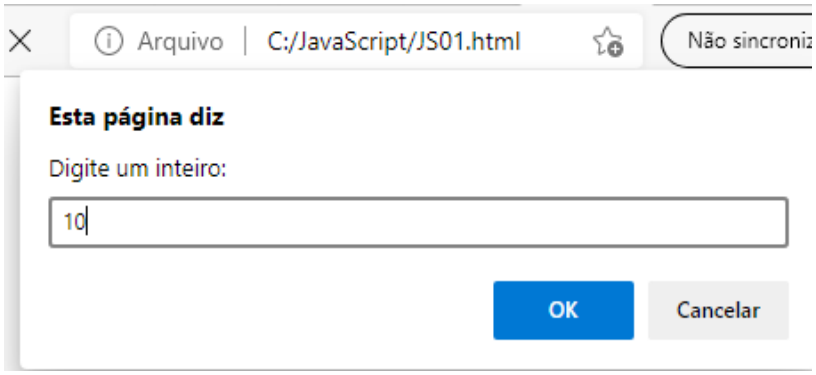
```
22
23 Lista = new Array();
24
25 for (x=0; x<10 ; x++){
26    Lista[x] = parseInt(window.prompt("Digite um inteiro: "));
27 }
28
29 valor = parseInt(window.prompt("Digite um número para ser buscado: "));
30 document.write(Lista + "<br><br>");
31 BuscaSequencial(valor, Lista);
32
33 </script>
```

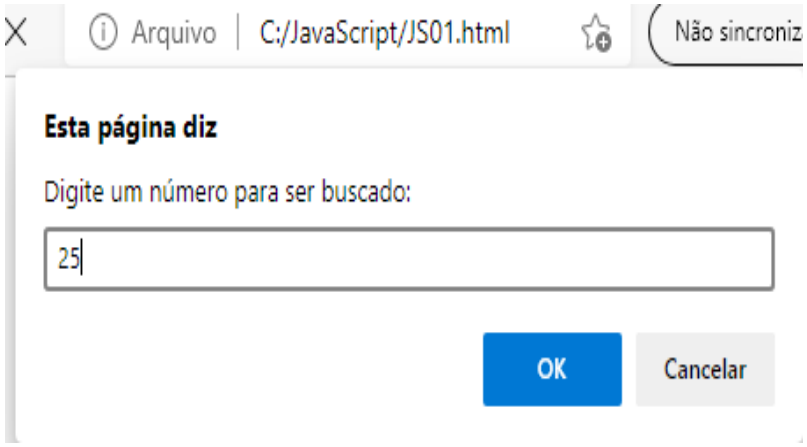
O resultado da execução do programa JavaScript ficou como segue, sendo o primeiro para a busca de um número existente na lista e o segundo para a busca de um número inexistente na lista:






Arquivo | C:/JavaScript/JS01.html
Programamando JavaScript
1,2,3,4,5,6,7,8,9,10
Achou 8


Arquivo | C:/JavaScript/JS01.html
Não sincroniz
Esta página diz
Digite um inteiro:
10
OK
Cancelar


Arquivo | C:/JavaScript/JS01.html
Não sincroniz
Esta página diz
Digite um número para ser buscado:
25
OK
Cancelar


Arquivo | C:/JavaScript/JS01.html
Programamando JavaScript
1,2,3,4,5,6,7,8,9,10
25 não encontrado

Exercícios

1. Assinale a alternativa CORRETA sobre o método de ordenação que pode ser utilizado em um programa JavaScript e que utiliza a ideia de flutuar o maior valor até o final de sua lista, seguindo a mesma ideia a cada sublista:

a) BubbleSort.
b) QuickSort.
c) MergeSort.
d) SelectionSort.
e) RadixSort.

2. Sobre o método de ordenação BubbleSort, que pode ser utilizado numa programação em JavaScript, é CORRETO afirmar que:

a) é o algoritmo que usa o conceito de que a ordenação acontece das pontas para o centro da lista.
b) é o algoritmo que usa o conceito de que a ordenação precisa acontecer de uma lista já ordenada.
c) é o algoritmo que usa o conceito de que é melhor ordenar do meio para as pontas das listas.
d) é o algoritmo que usa o conceito de que o maior número permanece sempre onde está.
e) é o algoritmo que usa o conceito de que o número maior vai flutuando até o final de uma lista.

3. Sobre o algoritmo de ordenação BubbleSort que pode ser utilizado numa programação em JavaScript, é INCORRETO afirmar que:

a) é um algoritmo de ordenação.
b) é um dos algoritmos de ordenação mais simples de se entender.
c) é um dos algoritmos de ordenação mais simples de se desenvolver.
d) utiliza apenas uma estrutura de repetição para o seu desenvolvimento.
e) usa o conceito de que o maior número vai flutuando até o final de uma lista.

4. Assinale a alternativa CORRETA sobre o algoritmo de busca, que pode ser utilizado em um programa JavaScript e que utiliza a ideia de passar por cada posição da lista, comparando com a informação a ser buscada:

a) Busca sequencial.
b) Busca binária.
c) Busca com árvores binárias.
d) Busca com árvores balanceadas.
e) Busca com árvores não balanceadas.

5. Sobre o algoritmo de Busca Sequencial que pode ser utilizado em um programa JavaScript, é CORRETO afirmar que:

a) é um dos algoritmos de busca mais difíceis de se desenvolver.
b) é um dos algoritmos de busca mais difíceis de se entender.
c) é um dos algoritmos de busca mais complexos de se desenvolver.
d) é um dos algoritmos de busca mais complexos de se entender.
e) é um dos algoritmos de busca mais simples de se entender e de desenvolver.

6. Sobre o algoritmo de busca sequencial que pode ser utilizado em um programa JavaScript, é INCORRETO afirmar que:

a) é um dos algoritmos de busca mais simples de se entender.
b) é um dos algoritmos de busca mais simples de se desenvolver.
c) o algoritmo vai passado por cada posição da lista, iniciando no início da lista, comparando se a informação da lista é a informação buscada.
d) caso a informação é encontrada, a busca continua até o final da lista.
e) pode acontecer de percorrer a lista toda e o elemento não seja encontrado na lista.

Gabaritos

1. **A**
   Um dos algoritmos de ordenação e um dos mais simples de se entender e de desenvolver o programa é o Bubble Sort. O Bubble Sort é um método que precisa de duas estruturas de repetição encadeada para o seu desenvolvimento e usa o conceito de que o número maior vai flutuando até o final de uma lista ou o número menor até o início da lista.

2. **E**
   O Bubble Sort é um método que precisa de duas estruturas de repetição encadeada para o seu desenvolvimento e usa o conceito de que o número maior vai flutuando até o final de uma lista ou o número menor até o início da lista.

3. **D**
   Um dos algoritmos de ordenação e um dos mais simples de se entender e de desenvolver o algoritmo é o Bubble Sort. O Bubble Sort é um método que precisa de duas estruturas de repetição encadeada para o seu desenvolvimento e usa o conceito de que o número maior vai flutuando até o final de uma lista ou o número menor até o início da lista.

4. **A**
   Na busca sequencial, o algoritmo vai passando por cada posição da lista, iniciando no início da lista, comparando se a informação da lista é a informação buscada.

5. **E**
   Um dos algoritmos de busca e um dos mais simples de se entender e de desenvolver o método é a busca sequencial. Na busca sequencial, o método vai passando por cada posição da lista, iniciando no início da lista, comparando se a informação da lista é a informação buscada.

6. **D**
   Um dos algoritmos de busca e um dos mais simples de se entender e de desenvolver o algoritmo é a busca sequencial. Na busca sequencial, o método vai passando por cada posição da lista, iniciando no início da lista, comparando se a informação da lista é a informação buscada. Caso encontre, a busca é parada e a informação é apresentada ou utilizada. Pode acontecer de percorrer a lista toda e o elemento não seja encontrado na lista.