Análise e Projeto de Algoritmos - Trabalho 02

Maurício El Uri

Universidade Federal do Pampa

1.6 Onde está a bolinha de gude?

Entrada de uma certa quantidade de bolinhas com números e números a serem consultados.

O algoritmo deve consultar nas bolinhas e retornar se cada consulta foi encontrada e onde foi encontrada na lista de bolinhas.

Solução desenvolvida

Classe abstrata JogoBolinha
Classe JogoBF
Classe JogoDC
Classe BubbleSort
Classe TextualGame

JogoBF

Método executaConsulta → T(n) é O(n)

```
private void executaConsulta(int consulta) {
        int resultado = -1;
        for (int i = 1; i < getBolinhas().size(); i++) {
                if (consulta == getBolinhas().get(i - 1)) {
                resultado = i - 1;
                }
        }
        setaResultado(consulta, resultado);
}
```

JogoBF

Método executa → T(n) é O(consulta*bolinhas)

```
public void executa() {
        for (int consulta : getConsultas()) {
                executaConsulta(consulta);
        }
}
```

JogoBF

Complexidade total, processamento por força bruta:

T(n) → O(consultas*bolinhas)

Bubble Sort

Método moveMaiorValorParaOFinal → T(n) é O(n)

```
private void moveMaiorValorParaOFinal(int tamFinal) {
        for (int index = 1; index < tamFinal; index++) {
                if (list.get(index) < list.get(index - 1)) {
                trocaValoresList(index, index - 1);
                }
        }
}
```

Bubble Sort

Método sort → T(n) é $O(n^2)$

```
public void sort() {
        for (int count = list.size(); count > 1; count--) {
                moveMaiorValorParaOFinal(count);
        }
}
```

JogoDC

Função buscaBinaria → T(n) é O(logn)

```
private int buscaBinaria(ArrayList<Integer> bolinhas, int minimo, int maximo, int consulta) {
        int meio = ((maximo + minimo) / 2);
        if (bolinhas.get(meio) == consulta) {
        return getBolinhas().indexOf(
                bolinhas.get(meio));
        }
        if (minimo >= maximo) {
                return -1;
        } else if (bolinhas.get(meio) < consulta) {
                return buscaBinaria(bolinhas, meio + 1, maximo, consulta);
        } else {
                return buscaBinaria(bolinhas, minimo, meio - 1, consulta);
        }
}
```

JogoDC

Método executa → T(n) é $O(n^2)$

```
public void executa() {
        ArrayList<Integer> listaOrdenada = ordenaBolinhas();
        for (int consulta : getConsultas()) {
                int resultado = buscaBinaria(listaOrdenada, 0,
                        listaOrdenada.size() - 1, consulta);
                setaResultado(consulta, resultado);
        }
}
```

Conclusões

- Podemos concluir que a ordenação significa maior parte do custo do algoritmo JogoDC ;
- JogoBF tem complexidade $O(n^2)$;
- O método de Busca binária (sem a ordenação) tem complexidade O(nlogn);
- Existem algoritmos mais eficientes para fazer a ordenação que também podem ser utilizados no lugar do Bubble Sort.

Muito obrigado!

Análise e Projeto de Algoritmos - Trabalho 02

Maurício El Uri

Universidade Federal do Pampa