ESTRUTURA DE DADOS

AGENDA 07/03/2017

- Função de complexidade
- Busca binária

ANALISAMOS PARA...

- Recursos necessários
- Avaliar desempenho
- Comparar algoritmos para a mesma tarefa
- Análise de tempo
- Análise de espaço

ANÁLISE DO ALGORITMO

- Tempo: Número de operações realizadas
- Espaço: Memória utilizada
- Modelo matemático: abstrai implementação e máquina para considerar o custo em função do tamanho da entrada

ANÁLISE DA CLASSE DE ALGORITMOS

- Algoritmos para uma tarefa específica
- Verifica-se toda a família de algoritmos
- Permite colocar limites para a complexidade dos algoritmos da classe
- Tem-se noção do mínimo necessário para realizar a tarefa
- Se um algoritmo tem a complexidade igual à mínima é considerado ótimo

SE POSSUEM A MESMA COMPLEXIDADE

- Considerar custos reais das operações
- Custos não aparentes: alocação de memória, indexação, carga, etc.

ANÁLISE PELA EXECUÇÃO

- Linguagem de programação
- Sistema operacional
- Hardware
- Verificar execução de programas na mesma máquina
- Custos do Algoritmo + não aparentes

CONTANDO OPERAÇÕES

```
função busca_linear(A, n, e):
        para i de 0 ate n faça:     //repete n+1 vezes
            se e == A[i]:           //repete n vezes
                retorna i ;         //pode repetir uma vez
        retorna -1;                 //pode repetir uma vez
```

- Tempo: 2n+2 ou 2(n+1) operações
- Espaço: n + 3 (tamanho n de A, +3 pelas variáveis auxiliares n, e e i)

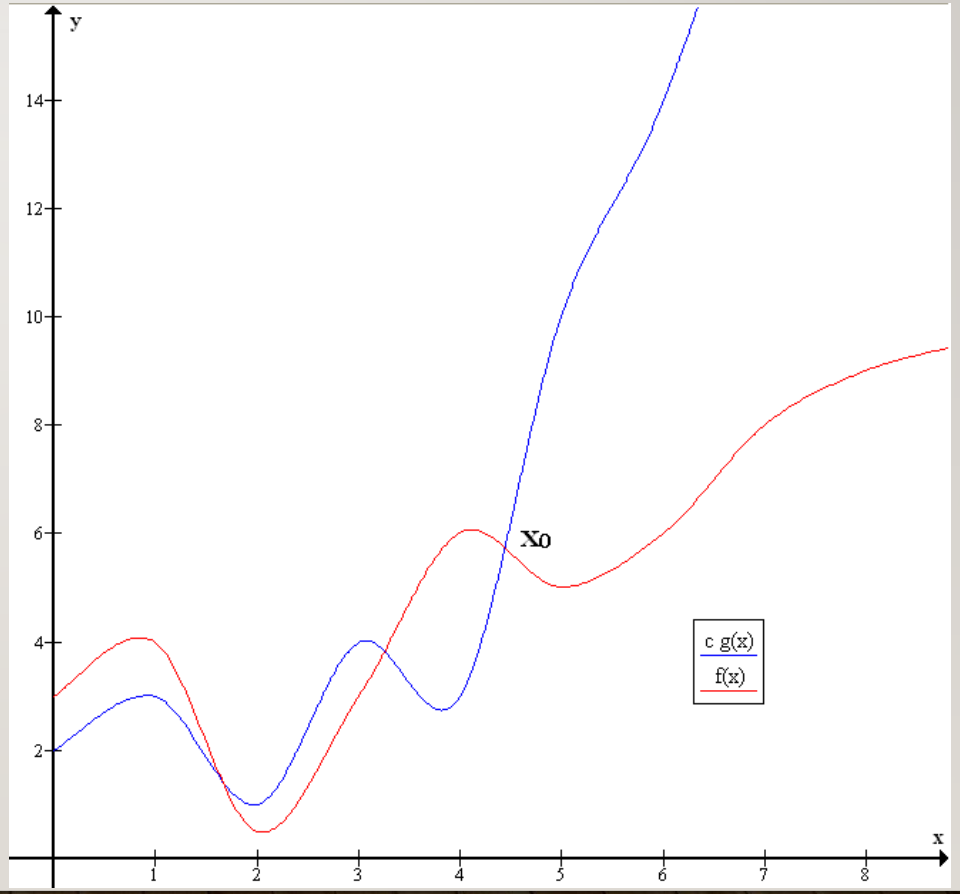
CONTANDO OPERAÇÕES

```
função busca_linear(A, n, e):
        i <- 0;                         //executa uma vez
        enquanto i < n:                 //repete n+1 vezes
                se e == A[i]:           //repete n vezes
                        retorna i;      //pode repetir uma vez
                i <- i+1;               //repete n vezes
        retorna -1;                     //pode repetir uma vez
```

- Tempo: 3n+3 ou 3(n+1) operações
- Espaço: n + 3 (tamanho n de A, +3 pelas variáveis auxiliares n, e e i)

NOTAÇÃO ASSINTÓTICA

- $f(x) = O(g(x))$
- $0 \leq f(x) \leq c \cdot g(x), \forall\, n \geq n_0$
- Análise para valores grandes de n



NOTAÇÃO ASSINTÓTICA

- $f(n) = O(f(n))$
- $cf(n) = O(f(n))$, c constante
- $O(f(n)) + O(f(n)) = O(f(n))$
- $O(O(f(n))) = O(f(n))$
- $O(f(n) + g(n)) = O(max(f(n), g(n)))$
- $O(f(n))O(g(n)) = O(f(n)g(n))$
- $f(n)O(g(n)) = O(f(n)g(n))$

EXEMPLO DE FUNÇÕES

- Polinomial: $O(p(n)), p(n)\ polinômio$
  - Busca, ordenação, etc.
- Exponencial: $O(a^n), a > 1$
  - Caixeiro viajante, clique, etc.
- Logarítmica: $O(\log(n))$
- Fatorial: $O(n!)$,

CASOS DE ESTUDO

- Melhor
- Médio
- Pior

BUSCA BINÁRIA

```
função busca_binaria(A, n, e): //A[0..n-1], i, f, e inteiros
        i = 0;
        j = n-1;
        enquanto(i <= j):
                m = chão((i+j)/2);
                se e == A[m]     retorna m;
                se e <  A[m]     j = m – 1;
                se não           i = m + 1;
        retorna -1;
```

BUSCA BINÁRIA

- Necessita está ordenado
- Complexidade O(log(n))