Aula 1 - Eficiência Computacional

Luís Felipe

UFF

29 de Agosto de 2023

Luís Felipe

29/08/23

Algoritmos

- Um algoritmo é uma sequência estruturada de passos para solucionar um dado problema.
- Dois aspectos são considerados quando lidamos com algoritmos: corretude e eficiência
  - A corretude de um algoritmo é um "comprovante" de que o algoritmo soluciona o problema de maneira correta.
    - Característica intrínseca de um algoritmo.
    - Quando não verificada de modo imediato, necessita ser provada formalmente.
  - A eficiência pode ser analisada sob duas perspectivas: tempo e espaço.
  - O que significa ser eficiente com relação ao tempo? Tempo cronológico? Tempo de implementação?
    - A complexidade de tempo de um algoritmo é dada pelo número de passos que ele executa.
    - O processo que determina quão eficiente é um algoritmo é chamado de análise.
    - Neste curso estaremos predominantemente interessados na análise temporal.

Luís Felipe

29/08/23

- Normalmente, estamos interessados em algoritmos exatos e eficientes, i.e., cujo tempo de execução para encontrar a resposta exata para o problema seja um polinômio em função do tamanho da entrada.

- Em alguns contextos, contudo, uma solução ineficiente pode ser interessante.

- Em outros, uma solução eficiente pode ser super importante, mesmo que a solução apresentada não seja exata.

Luís Felipe

29/08/23

Algoritmo Iterativo vs Recursivo

Exemplos: Algoritmos Iterativos

Algoritmo I: Inversão de uma sequência

Para $i = 1, \ldots, \lfloor \frac{n}{2} \rfloor$ faça
    temp $:= S[i]$
    $S[i] := S[n - i + 1]$
    $S[n - i + 1] :=$ temp

Luís Felipe

29/08/23

Algoritmo Iterativo vs Recursivo

Exemplos: Algoritmos Iterativos

Algoritmo 2: Cálculo do fatorial

fat:= 1
Para $i = 1, \ldots, n$ faça
    fat := $i \times$ fat

Luís Felipe

29/08/23

Algoritmo Iterativo vs Recursivo

Exemplos: Algoritmos Recursivos

Algoritmo 3: Cálculo do fatorial (recursivo)

```
funcao fat(i)
Se i == 0 então
   fat(i) := 1
Senão
   fat(i) := i × fat(i − 1)
```

Luís Felipe

29/08/23

Algoritmo Iterativo vs Recursivo

Exemplos: Algoritmos Recursivos

Algoritmo 4: Torre de Hanoi

```
procedimento hanoi(n,A,C,B)
Se n > 0 então
   hanoi (n-1,A,B,C)
   Mover o disco do topo de A para C
   hanoi (n-1,B,C,A)
```

Luís Felipe

29/08/23

Complexidade de Algoritmos

- Uma característica importante que analisamos em um algoritmo é seu tempo de execução.
- A análise desse tempo de execução pode ser empírica ou analítica.
- Na análise empírica, consideram-se entradas diversas e avalia-se o tempo de execução de cada uma delas.
  - Fatores como a máquina, linguagem e compilador influenciam em uma análise empírica.
- O mesmo não acontece em uma análise analítica, onde obtém-se uma ordem de grandeza do tempo de execução do algoritmo.
  - Isto é, uma expressão matemática que traduza seu comportamento de tempo.

Luís Felipe

29/08/23

Observações

- Vamos considerar que temos uma grande quantidade de dados.
- Somente o comportamento assintótico é avaliado.
  - Constantes aditivas e multiplicativas serão desprezadas na expressão matemática obtida.
- É preciso definir uma variável em relação a qual a expressão avaliará o tempo de execução do algoritmo.
  - A ideia é exprimir o tempo de execução em função da entrada.

Luís Felipe

29/08/23

Conceitos fundamentais

- O processo de execução de um algoritmo se divide em etapas elementares chamadas passos.
  - Cada passo consiste na execução de um número fixo de operações cujo tempo de execução é constante.

- Operação dominante: operação básica de maior frequência.

- O número de passos de um algoritmo é dado pelo número de vezes que a operação dominante é executada.

- A expressão que avalia o tempo de execução de um algoritmo é dada pelo número de passos efetuados pelo algoritmo dada uma entrada.

Luís Felipe

29/08/23

Exemplos:

1. Número de passos do algoritmo de inversão de sequências: $\lfloor \frac{n}{2} \rfloor, n > 1$
2. Adição de matrizes

Algoritmo 5: Adição de matrizes

Para $i = 1, \ldots, n$ faça
  Para $j = 1, \ldots, n$ faça
    $c_{ij} = a_{ij} + b_{ij}$

número de passos ?

n^2

Luís Felipe

29/08/23

Exemplos:

3. Multiplicação de matrizes

   Algoritmo 5: Multiplicação de matrizes

   Para $i = 1, \ldots, n$ faça
   Para $j = 1, \ldots, n$ faça
   $c_{ij} = 0$
   Para $k = 1, \ldots, n$ faça
   $c_{ij} = c_{ij} + a_{ik} b_{kj}$

número de passos ?

n^3

Luís Felipe

29/08/23

Complexidade de Tempo

Sejam:

- A um algoritmo
- $E = \{E_1, E_2, \ldots, E_m\}$ o conjunto de todas as entradas possíveis para A
- t_i o número de passos de A dada E_i

Definem-se:

- Complexidade de Pior Caso: $\max_{E_i \in E} t_i$ limite superior
- Complexidade de Melhor Caso: $\min_{E_i \in E} t_i$ limite inferior
- Complexidade de Caso Médio: $\sum_{1 \leq i \leq m} p_i t_i$, onde p_i denota a probabilidade de ocorrência da entrada E_i.

OBS.: De modo geral, o termo complexidade referir-se-á à complexidade de tempo de pior caso.

Luís Felipe

29/08/23

Vantagens e desvantagens

- Pior caso:
  - determina o número de passos executados com a entrada mais desfavorável (muito pessimista)
  - quase sempre relevante
  - a mais utilizada
- Melhor caso:
  - determina o número de passos executados com a entrada mais favorável (muito otimista)
  - em geral, menor relevância
- Caso médio:
  - determina o número de passos de forma proporcional às entradas (mais justo)
  - em geral, relevante
  - o tratamento é matemático e pode não ser simples
  - depende das probabilidades de ocorrência das entradas