Aula 01

David Déharbe
Programa de Pós-graduação em Sistemas e Computação
Universidade Federal do Rio Grande do Norte
Centro de Ciências Exatas e da Terra
Departamento de Informática e Matemáica Aplicada

Plano da aula

Informações administrativas

Introdução

Problemas computacionais

Análise de algoritmos

PPgSC

Meta

Garantir que os egressos do Programa de Pós-graduação em Sistemas e Computação tenham uma base de conhecimentos suficientemente sólida em Ciência da Computação.

Objetivos

- algoritmos;
- estruturas de dados;
- estratégias algorítmicas;
- análise de algoritmos.

PPgSC

Informações administrativas

- Docente: David Déharbe.
- Carga horária: 60 horas, 4 créditos.
- Sala 3D6.
- Segundas e quartas, 08:55.
  - Horário de reposição: sextas 08:55.
- Aulas expositivas.
- Listas de exercícios.
- Dúvidas: fórum da turma virtual (SIGAA)
- Material:

  2015.1 `http://ufrn.academia.edu/DavidDeharbe`

  2015.2 `http://DavidDeharbe.github.io` ⇒ Lectures

PPgSC

Informações administrativas

Calendário

- Datas sem aula:

missões	27/07, 14/10, 16/11, 18/11
proficiência	29/07
feriados	07/09, 12/10, 02/11
férias	02/12, 07/12, 09/12

- Datas de reposição: 31/07 14/08, 28/08, 11/09, 25/09, 09/10, 23/10
- Datas das avaliações:
  1. 31/08
  2. 07/10
  3. 30/11

PPgSC

Avaliação

- Três provas escritas. não há prova de recuperação
- Média: $(P1 + P2 + P3)/3$
- Conversão nota crédito:

[0, 2[	[2; 4[	[4, 6]	]6, 8]	]9, 10]
E	D	C	B	A

PPgSC

Programa

1. Complexidade de algoritmos.
2. Algoritmos de busca e ordenação.
3. Estruturas de dados:
   - arranjos dinâmicos; listas encadeadas; árvores binárias; árvores binárias de busca; árvores B; tabelas de dispersão; estruturas união busca.
4. Estratégias algorítmicas:
   - divisão e conquista; força bruta; abordagem gulosa; programação dinâmica.
5. Algoritmos em grafos.
6. $\mathcal{NP}$-completude.

PPgSC

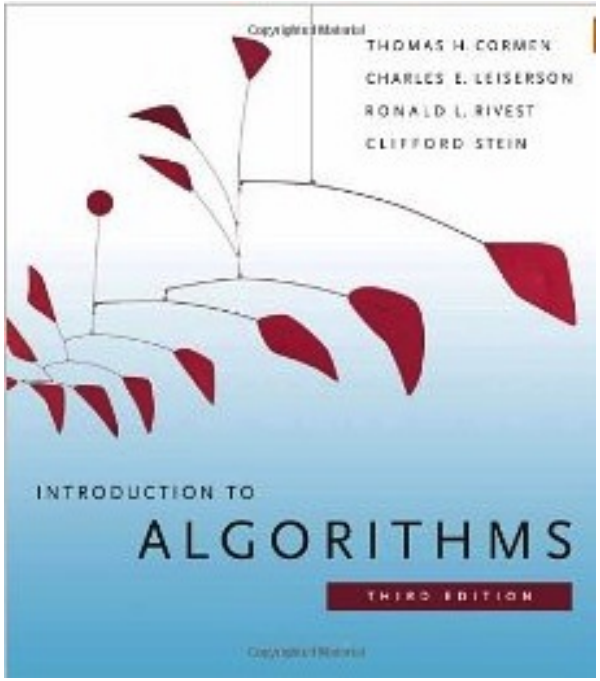
Objetivos

Fixar os conceitos de

- problema computacional
  - instância de problema
- algoritmo
- análise de algoritmos
  - correção
  - complexidade computacional

PPgSC

Referências bibliográficas



Introduction to Algorithms. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. MIT Press, 1990 (disponível em português, em edições mais recentes).

PPgSC

Definição (problema computacional)

Um *problema computacional* define como devem ser relacionados dados de entrada com dados de saída.

Exemplo

O problema da *ordenação* pode der definido da seguinte maneira:

> *Seja A um conjunto munido de uma relação de ordem total $\prec$.*
> *O problema da ordenação consiste em, dado uma sequência a de n valores $a_1, a_2, \ldots a_n$ de A, encontrar $a' = a'_1, a'_2, \ldots a'_n$ tal que*
>
> - *a' é uma permutação de a e*
> - *$a'_i \preceq a'_{i+1}$, para $1 \leq i < n$.*
>
> *Dados de entrada: a. Dados de saída: a'.*

PPgSC

Definição (instância de um problema computacional)

A *instância* de um problema computacional é um caso particular de dados de entrada para um problema computacional.

Exemplo

Ordenar a seguinte sequência de inteiros usando a relação de ordem total $<$:

$$3123, 214, 1011, 1125, 215, 10981, 42, 44648.$$

PPgSC

Observações

- Utilizamos várias noções matemáticas para definir o problema;
- Desvantagem: é um vocabulário específico.
- Vantagem: permite expressar-se de forma geral e precisa.

PPgSC

Prática

Considere o seguinte enunciado (Manber 1989, ex 1.1):

> *Write down the numbers 1 to 100 each on a separate card. Shuffle the cards and rearrange them in order again.*

Qual(is) problema(s) computacional(is) está(ão) envolvidos nesta tarefa?
Expressar a resposta de duas maneiras:

informalmente: em português somente;

formalmente: em português e usando conceitos matemáticos.

PPgSC

Prática

Considere o seguinte enunciado (Manber 1989, ex 1.2):

> *Write down the following 100 numbers each on a separate card. Shuffle the cards and rearrange them in order again.*
>
> $$32918, 21192, 11923, 4233, 88231 \cdots 11329, 2253.$$

Qual(is) problema(s) computacional(is) está(ão) envolvidos nesta tarefa?
Expressar a resposta de duas maneiras:

informalmente: em português somente;

formalmente: em português e usando conceitos matemáticos.

PPgSC

Prática

Considere o seguinte enunciado (Manber 1989 1.3):

> *Consider the following list of numbers. Your job is to erase as few of those numbers as possible such that the remaining numbers appear in increasing order.*
>
> $9, 44, 32, 12, 7, 42, 34, 92, 35, 37, 41, 8, 20, 27, 83, 64, 61,$
> $28, 39, 93, 29, 17, 13, 14, 55, 21, 66, 72, 23, 73, 99, 1, 2, 88,$
> $77, 4, 65, 83, 84, 62, 5, 11, 74, 68, 76, 78, 67, 75, 69, 70, 22,$
> $71, 24, 25, 26$

Qual(is) problema(s) computacional(is) está(ão) envolvidos nesta tarefa?

Expressar a resposta de duas maneiras:

informalmente: em português somente;

formalmente: em português e usando conceitos matemáticos.

PPgSC

Algoritmos: introdução

> *"Science is what we understand well enough to explain to a computer; art is everything else."*
>
> — *Donald E. Knuth, Things a Computer Scientist Rarely Talks About.*

Esta frase do cientista Knuth pode ser usada par ilustrar tanto o que ciência é quanto o que significa "explicar coisas para um computador".
E explicar coisas para um computador é justamente o propósito de um *algoritmo*.

PPgSC

Algoritmos: introdução

Um algoritmo é uma explicação passo a passo de como um computador deve resolver um problema computacional.

1. *ler uma instância do problema;*
2. *calcular o resultado esperado para esta instância;*
3. *fornecer este resultado.*

PPgSC

Algoritmos: introdução

Um algoritmo é uma explicação passo a passo de como um computador deve resolver um problema computacional.

2. *calcular o resultado esperado para esta instância;*

PPgSC

Algoritmos: introdução

Um algoritmo é uma explicação passo a passo de como um computador deve resolver um problema computacional.

2. *calcular o resultado esperado para esta instância;*

explicação: em qual linguagem deve-se realizada?

passo a passo: quais são os passos básicos que um computador pode executar?

Esta explicação depois é traduzida *manualmente* em uma linguagem de programação de computadores.

PPgSC

Algoritmos: o modelo computacional

- *Random Access Machine*:
  - processador
  - memória
  - (dispositivo de entrada)
  - (dispositivo de saída)
- algoritmo/programa: lista de instruções;
- o processador executa uma instrução por vez;
- fluxo: instruções são executadas em sequência;
- mas o fluxo pode ser desviado para uma instrução qualquer condicionalmente ou incondicionalmente.

PPgSC

Algoritmos: precisão mas não padronização

- Quase cada livro e cada autor possui uma notação algorítmica diferente!
- O algoritmo é traduzido *manualmente* em uma linguagem de programação de computadores.
- A notação algorítmica é intepretada por um ser humano.
- A linguagem de programção é interpretada por uma máquina.
- A notação algorítmica deve ao mesmo tempo não ter ambigüidade e pode ser livre das amarras impostas pelas linguagens de programação.
- As notações algorítmicas se equivalem essencialmente.

PPgSC

Algoritmos: os padrões estruturais

- identificar dados a serem manipulados: var
- estruturar dados: array, pointer, record, enum
- execução condicional de um bloco de comandos: **if**
- repetição condicional de um bloco de comandos: **while**
  - **for** , **to** , **downto**
- alterar um dado: $\leftarrow$
- agrupar blocos de comandos
- agrupar e nomear um bloco de comandos (sub-rotina)
- redirecionamento: **goto**
- indentação é usada para identificar os blocos.

PPgSC

Algoritmo: ordenação por inserção

(Cormen et al. 1990)

INSERTION-SORT(A)

```
1  for j ← 2 to length[A]
2       do key ← A[j]
3          ▷ Insert A[j] into the sorted sequence A[1 . . j − 1].
4          i ← j − 1
5          while i > 0 and A[i] > key
6               do A[i + 1] ← A[i]
7                  i ← i − 1
8          A[i + 1] ← key
```

▷ indica que o resto da linha é um *comentário*.

PPgSC

Ordenação por inserção: exemplo

$$A = \quad 8 \quad 2 \quad 4 \quad 9 \quad 3 \quad 6$$

INSERTION-SORT(A)

```
1  for j ← 2 to length[A]
2        do key ← A[j]
3           ▷ Insert A[j] into the sorted sequence A[1 . . j − 1].
4           i ← j − 1
5           while i > 0 and A[i] > key
6                 do A[i + 1] ← A[i]
7                    i ← i − 1
8           A[i + 1] ← key
```

PPgSC

Ordenação por inserção: exercício

INSERTION-SORT(A)

```
1  for j ← 2 to length[A]
2       do key ← A[j]
3          ▷ Insert A[j] into the sorted sequence A[1 . . j − 1].
4          i ← j − 1
5          while i > 0 and A[i] > key
6               do A[i + 1] ← A[i]
7                  i ← i − 1
8          A[i + 1] ← key
```

1. Aplique o algoritmo para $A = \langle 31, 41, 59, 26, 41, 58 \rangle$.
2. Reescreva o procedimento INSERTION-SORT para ordenar em ordem decrescente.

(Cormen et al. 1990)

PPgSC

Algoritmos: exercício

Considere o problema da *busca*:

entrada Uma sequência de n valores $A = \langle a_1, a_2, \ldots a_n \rangle$ e um valor v.

resultado Um índice i tal que $a_i = v$ ou um valor especial NIL se v não pertence a A.

Escreva o algoritmo da *busca linear* que resolve este problema. (Cormen et al. 1990)

PPgSC

Algoritmo: análise

O algoritmo calcule o resultado esperado? o algoritmo faz uso eficiente dos recursos computacionais?

correção O algoritmo é correto?

complexidade Quais recursos o computador usará para executar este algoritmo?

- tempo
- memória

PPgSC

Correção de algoritmos

Um algoritmo para um determinado problema computacional P é *correto* quando cada vez que é aplicado a uma instância do problema:

PPgSC

Correção de algoritmos

Um algoritmo para um determinado problema computacional P é *correto* quando cada vez que é aplicado a uma instância do problema:

- ele calcula o resultado em um número finito de passos;
- o resultado calculado é o mesmo que especificado na descrição do problema P.

PPgSC

Correção de algoritmos

É mais fácil mostrar que um algoritmo *não* é correto!

PPgSC

Correção de algoritmos

É mais fácil mostrar que um algoritmo *não* é correto!

Receita para provar que um algoritmo é incorreto.

Basta encontrar uma instância do problema tratado tal que:

- o algoritmo executa-se infinitamente, ou
- o algoritmo termina em um número finito de passos e o resultado calculado não é o mesmo que especificado na descrição do problema P.

PPgSC

Exercício: Mostre que o algoritmo é incorreto

LINEAR-SEARCH(A, v)

```
1  j ← 1
2  while A[j] ≠ v
3        do j ← j + 1
4  return j
```

PPgSC

Exercício: Mostre que o algoritmo é incorreto

LINEAR-SEARCH(A, v)

1 $j \leftarrow 1$
2 **while** $A[j] \neq v$ and $j < length[A]$
3 **do** $j \leftarrow j + 1$
4 **return** j

PPgSC

Exercício: Mostre que o algoritmo é incorreto

LINEAR-SEARCH(A, v)

1 $j \leftarrow 1$
2 **while** $A[j] \neq v$ and $j \leq length[A]$
3 **do** $j \leftarrow j + 1$
4 **return** j

PPgSC

Versão final

LINEAR-SEARCH(A, v)

```
1  j ← 1
2  while A[j] ≠ v and j ≤ length[A]
3        do j ← j + 1
4  if j ≤ length[A]
5     then return j
6     else  return NIL
```

PPgSC

Versão final

LINEAR-SEARCH(A, v)

```
1  j ← 1
2  while A[j] ≠ v and j ≤ length[A]
3        do j ← j + 1
4  if j ≤ length[A]
5     then return j
6     else  return NIL
```

O algoritmo A é correto se e somente se

- ele não é incorreto.

PPgSC

Versão final

LINEAR-SEARCH(A, v)

```
1  j ← 1
2  while A[j] ≠ v and j ≤ length[A]
3        do j ← j + 1
4  if j ≤ length[A]
5     then return j
6     else  return NIL
```

O algoritmo A é correto se e somente se

- ele não é incorreto.
- $\neg\exists$ instância i do problema t.q. $A(i)$ não termina ou tem um resultado errado.

PPgSC

Versão final

LINEAR-SEARCH(A, v)

```
1  j ← 1
2  while A[j] ≠ v and j ≤ length[A]
3        do j ← j + 1
4  if j ≤ length[A]
5     then return j
6     else  return NIL
```

O algoritmo A é correto se e somente se

- ele não é incorreto.
- $\neg\exists$ instância i do problema t.q. $A(i)$ não termina ou tem um resultado errado.
- $\forall$ instancia i do problema, $A(i)$ termina com resultado certo.

PPgSC

Exercício

- Supondo que não seja autorizado avaliar $A[j]$ quando $j \leq 0$ ou $j > \mathit{length}[A]$, o algoritmo continua correto?
- Se não for correto, como corrigir?
- O que acontece quando A é uma sequência de zero elementos? Está em conformidade com a especificação?

PPgSC

“Mostre que o algoritmo é correto”. Como responder?

- A definição de algoritmo correto não é operacional: não é possível enumerar todas as instâncias e verificar que o algoritmo termina com o resultado certo.

  Program testing can be used to show bugs, but never their absence!
  — Edsger Dijkstra (1930–2002).

- Mostrar que um algoritmo é correto requer utilizar técnicas de prova mais elaboradas que enumeração e cálculo.

PPgSC

“Mostre que o algoritmo é correto”. Como responder?

- A definição de algoritmo correto não é operacional: não é possível enumerar todas as instâncias e verificar que o algoritmo termina com o resultado certo.

  Program testing can be used to show bugs, but never their absence!
  — Edsger Dijkstra (1930–2002).

- Mostrar que um algoritmo é correto requer utilizar técnicas de prova mais elaboradas que enumeração e cálculo.

$\Rightarrow$ *Lógica de Hoare*

PPgSC

Complexidade de um algoritmo

- Um mesmo problema pode ser resolvido de várias maneiras diferentes.
- Como escolher entre diversos algoritmos?
- Estimar *o tempo de execução* de cada algoritmo.
  Complexidade do algoritmo

PPgSC

Complexidade de algoritmo

- Como quantificar a complexidade (temporal) de um algoritmo?
- Conta como?

PPgSC

Complexidade de algoritmo

- Como quantificar a complexidade (temporal) de um algoritmo?
- Conta como?
  - Quantas vezes cada instrução é executada.
- Depende de quê?

PPgSC

Complexidade de algoritmo

- Como quantificar a complexidade (temporal) de um algoritmo?
- Conta como?
  - Quantas vezes cada instrução é executada.
- Depende de quê?
  - Do *tamanho* da entrada.
  - Do *valor* da entrada.

PPgSC