ATAL

2003.2

Análise e Técnicas de Algoritmos

Prof. Jorge Figueiredo

http://www.dsc.ufcg.edu.br/~abrantes/atal032.html

Agenda

- Apresentação do curso
- Motivação
- Introdução informal

Apresentação do Curso

- Homepage (http://www.dsc.ufcg.edu.br/~abrantes/atal032.html)
- Lista de discussão (atal-l@dsc.ufcg.edu.br)
- Pré-requisito
- Horário e sala de aula
- Comportamento do aluno
- Referências
- Avaliação
- Horário de dúvidas

Introdução Informal

- O nosso curso é sobre técnicas e análise de algoritmos (computacionais).
- Algoritmo:
  - Procedimento computacional que toma algum valor (conjunto) como entrada e produz um valor (conjunto) como saída.
  - Não é um programa.
  - Pode ser implementado de diferentes formas.

Que aspectos são importantes no desenvolvimento e estudo de programas computacionais?

- Modularidade
- Desempenho
- Corretude
- Funcionalidade
- Manutenção
- Amigável
- Robustez
- Simplicidade
- Confiabilidade
- Escalabilidade
- Reuso
- Portabilidade

- A análise de algoritmos permite o estudo teórico de programas computacionais:
  - Desempenho.
  - Utilização de recursos.
  - Corretude.
- Estudo de métodos, técnicas, idéias, dicas para desenvolver algoritmos (eficientes).

- Importância do estudo e análise de algoritmos:
  - Ajuda no entendimento de escalabilidade.
  - Permite definir o que é viável e o que é impossível.
  - A matemática utilizada serve como uma linguagem para lidar com o comportamento de um programa.
  - Provê meios para comparar diferentes soluções de um mesmo problema.

Problemas Computacionais

- Especifica a relação entre a entrada e a saída desejada.

Ordenação

Entrada: Uma seqüência de n números $\langle a_1, a_2, \ldots, a_n \rangle$.

Saída: Uma reordenação da seqüência de entrada $\langle a'_1, a'_2, \ldots, a'_n \rangle$, onde $a'_1 \leq a'_2 \leq \ldots \leq a'_n$.

Número Primo

Entrada: Uma número natural q.

Saída: *sim* ou *não*, dependendo se q é primo.

Problemas Computacionais

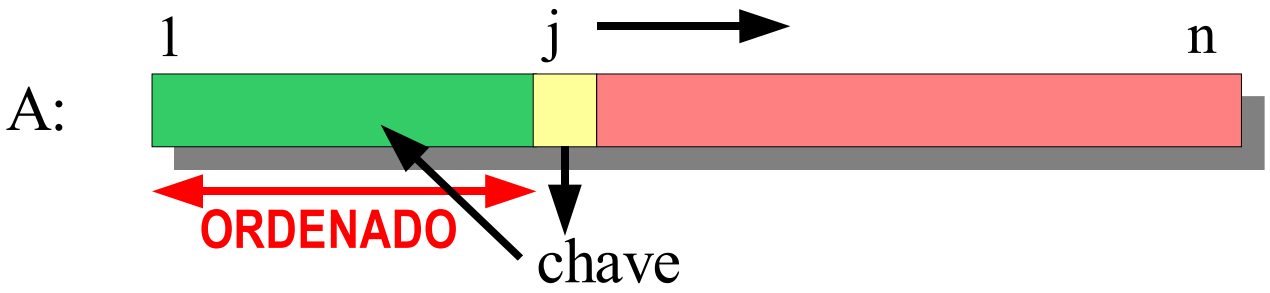
- Um instância de um problema computacional é um possível valor para a entrada.
  - *‹45, 7, 13, 23, 2›* é uma instância para o problema da ordenação.
  - *29* é uma instância para o problema dos números primos.
- Um algoritmo está correto se, para qualquer instância, ele termina e retorna como saída o valor esperado.

Como Descrever Algoritmos?

- Aspectos como precisão e facilidade de expressão são importantes.
- Três formas:
  - Linguagem natural (português ou inglês).
  - Pseudo-Código.
  - Linguagem de programação.

Sugestão: Usar as convenções definidas pelo Prof. Dalton
http://www.dsc.ufcg.edu.br/~dalton/2002.2/edados/convencoes

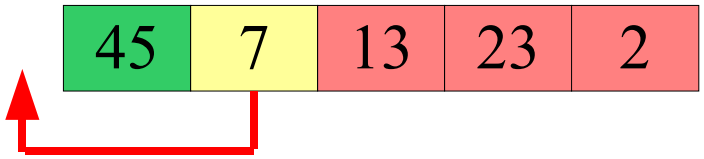
Insertion Sort



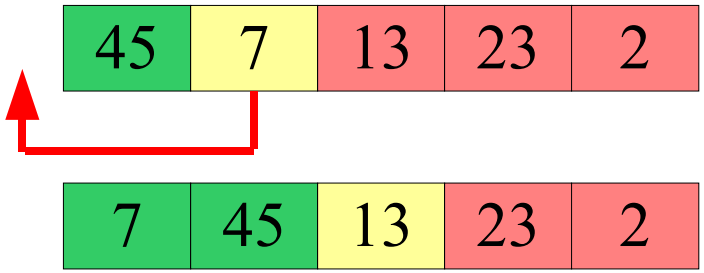

```
InsertionSort(A, n)
for j← 2 to n do
    chave ← A[j]
    ► insere A[j] na parte ordenada A[1..j-1]
    i ← j – 1
    while i > 0 e A[i] > chave do
        A[i + 1] ← A[i]
        i ← i – 1
    A[i + 1] ← chave
```

Exemplo do Insertion Sort

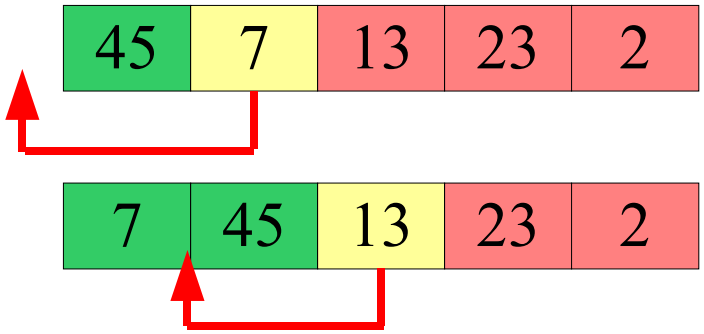
Exemplo do Insertion Sort



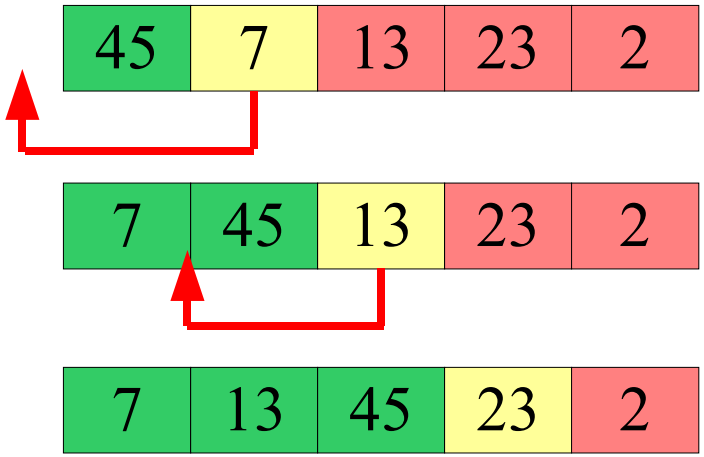
Exemplo do Insertion Sort



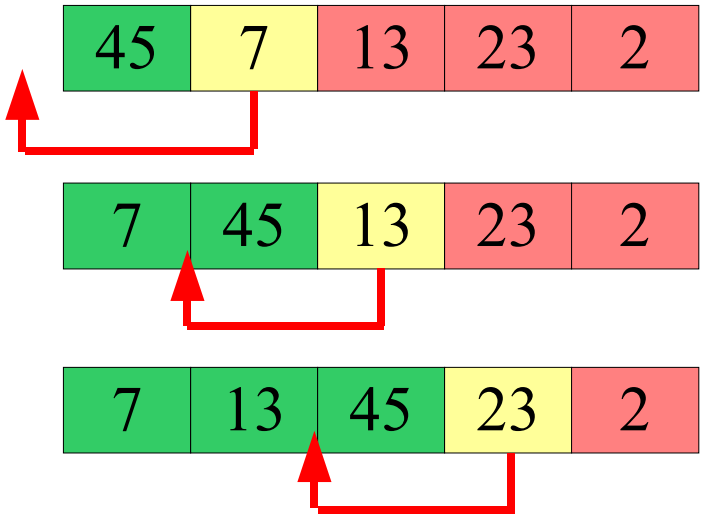
Exemplo do Insertion Sort



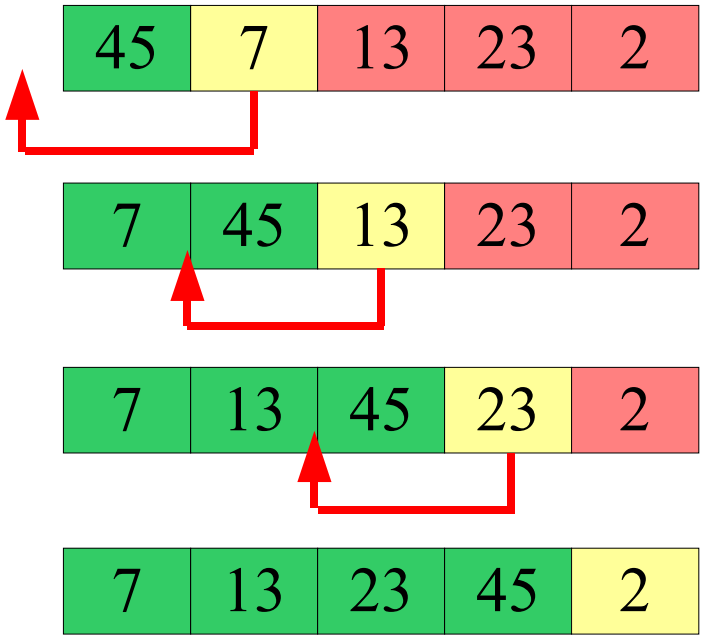
Exemplo do Insertion Sort



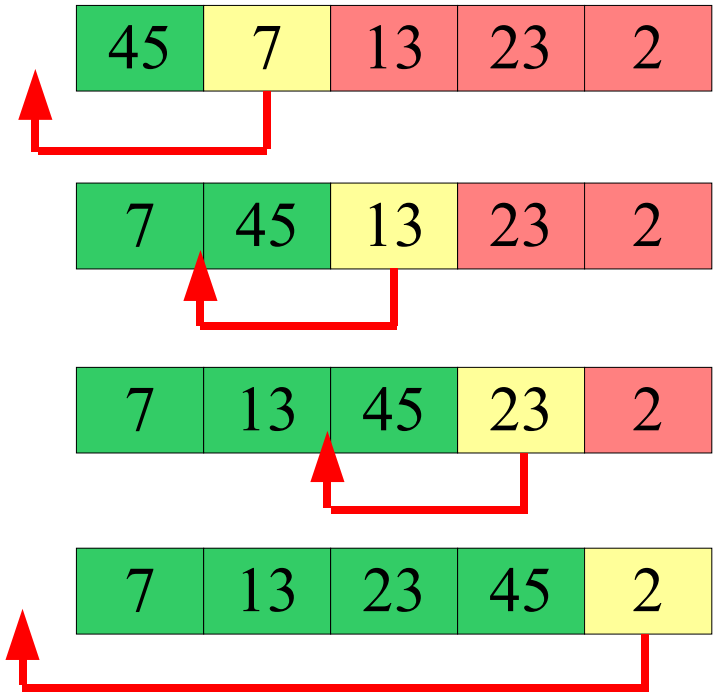
Exemplo do Insertion Sort



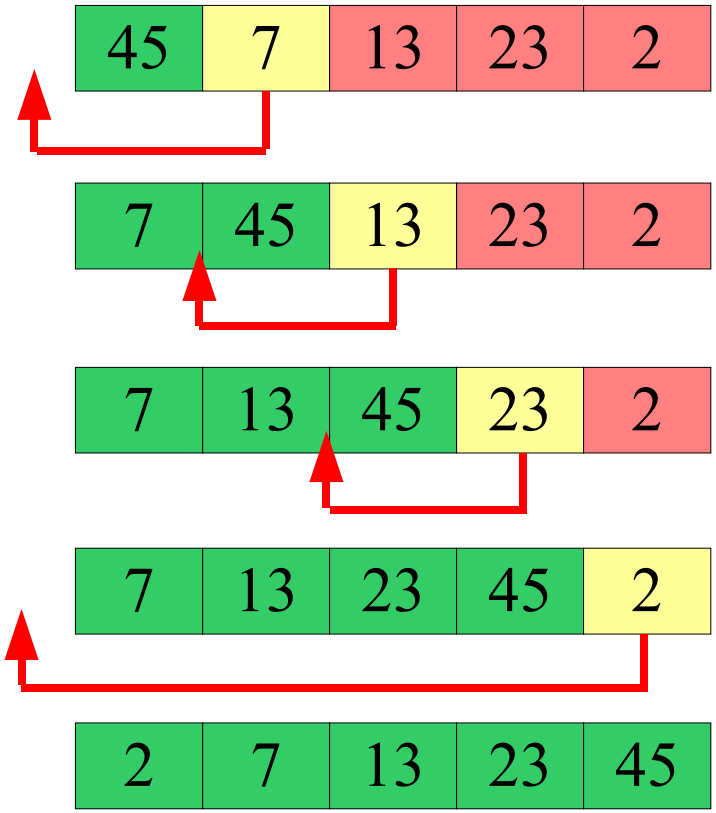
Exemplo do Insertion Sort



Exemplo do Insertion Sort



Exemplo do Insertion Sort



Tempo de Execução

- O tempo de execução de um algoritmo depende da *cara* da entrada.
- Na análise do tempo de execução, a parametrização é baseada no tamanho da entrada.
- Na análise de algoritmos, é uma forma de definir eficiência.

Tempo de Execução

- Tipos de análise:
  - Pior caso:
    - Maior tempo de execução de um algoritmo para qualquer entrada de tamanho n.
    - É o tipo mais utilizado. Todos gostam de garantia.
  - Caso médio:
    - Tempo esperado de um algoritmo sobre todas as entradas de tamanho n.
    - Necessidade de usar distribuição estatística.
  - Melhor caso:
    - Raramente é feita.

Independência de Máquina

- Para comparar os diferentes algoritmos, de forma justa, é necessário definir um modelo abstrato de máquina.
- Máquina de Acesso Aleatório:
  - Um único processador genérico.
  - Instruções executadas sequencialmente, sem operações concorrentes.
  - Memória ilimitada.
  - Instrução básica toma uma unidade de tempo.

Análise de Algoritmos

- Forma de fazer análise de tempo de execução é contar o número de instruções executadas.
- Na prática, em algoritmos reais, essa não é uma boa alternativa.
- Abordagem que estuda a ordem de crescimento de funções que identificam os algoritmos.

Análise Assintótica

Análise do Insertion Sort

- Pior caso:
  - Entrada em ordem reversa.
  - $O(n^2)$
- Caso médio:
  - $O(n^2)$
- Pior caso:
  - Entrada ordenada.
  - $O(n)$

Corretude de Algoritmos

- Provar corretude não é trivial.
- Utilizamos o conceito de indução matemática.
- Em muitos casos, verificação através da propriedade de invariante de laço:
  - Inicialização.
  - Manutenção.
  - Término.

Corretude do Insertion Sort

- É um algoritmo não recursivo.
- Verificação através do invariante de laço:
  - Os elementos A[1, j-1] estão ordenados.

Mais Objetivos do Curso

- Como expressar algoritmos.
- Como analisar algoritmos.
- Como validar algoritmos.
- Como criar algoritmos.
- Como reutilizar algoritmos.
- Como aplicar algoritmos bastante conhecidos

Razões para o estudo de Algoritmos

- Evitar reinventar a roda:
  - Existem bons algoritmos que solucionam problemas importantes.
- Ajudar no desenvolvimento de seus algoritmos:
  - Nem sempre existe um algoritmo de prateleira que sirva para resolver o seu problema.
  - O conhecimento de algoritmos bem estabelecidos é fonte de inspiração.
  - Muitos dos princípios de projetos de algoritmos são úteis em todos os problemas de programação.

Razões para o estudo de Algoritmos

- Ajudar a entender ferramentas que utilizam algoritmos particulares.
  - Por exemplo, ferramentas de compressão.
- Útil conhecer as técnicas de algoritmos empregadas para resolver determinadas classes de problemas.