CPS740 Algoritmos e Grafos (PESC)
COS242 Teoria de Grafos (ECI)

Aula 1

- Principais informações estão em nossa página:

http://www.cos.ufrj.br/~franklin/cps740
/cos242

- Confiram com freqüência!

- Este é um curso introdutório de Algoritmos e Grafos.

- Seguindo livro do Jayme (referência na página)

- Enfoque computacional: os algoritmos são desenvolvidos considerando-se uma posterior implementação em computador; preocupação com eficiência de tempo e espaço

- Tratamento matemático: verificação de corretude do algoritmo, análise de eficiência

- Vamos cobrir os 8 primeiros capítulos do livro

1. Introdução, complexidade etc
2. Iniciação à T. dos Grafos
3. Técnicas básicas
5. Outras técnicas

FRANKLIN

4. Buscas em grafos
6. Fluxo máximo em redes
7. Caminhos mínimos
8. Emparelhamentos máximos

CELINA

Um pouco de história

- Marco inicial foi um problema algorítmico:

O PROBLEMA DAS PONTES DE KÖNIGSBERG

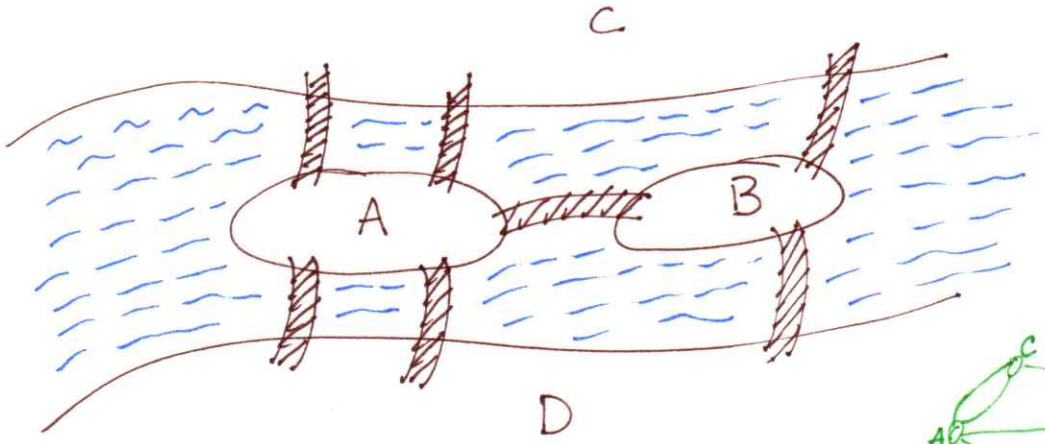
Fazia parte da PRÚSSIA.

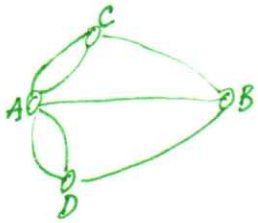
Hoje Kaliningrado, RÚSSIA.

- Resolvido por Leonhard Euler em 1736(?) 1735(?)

"Solutio problematis ad geometriam situs pertinentis"
In: Commentarii academiae scientiarum Petropolitanae 8, 1741

Vejam na página do curso um link para o artigo original e outro para uma tradução.





- Quatro regiões (A, B, C, D)
- Sete pontes
- Existe um trajeto que passe por TODAS as pontes, cada uma delas exatamente UMA vez, retornando à região de partida?
- Euler mostrou que NÃO É possível, e ainda generalizou, usando um modelo em grafos.
- Euler provou que existe o trajeto desejado se e somente se cada região tem um número par de pontes.

Considerado o primeiro teorema em Teoria dos Grafos!

Mais um pouco de história

Meados do século XIX:
- formulação do problema das quatro cores (1852)
- problema do ciclo Hamiltoniano por Hamilton
- teoria das árvores (uma classe especial de grafos) por Kirchhoff e Cayley
  - aplicações em circuitos elétricos
  - aplicações em química org.

- Problema das quatro cores: é possível colorir um mapa arbitrário usando ~~quatro cores~~ não mais que quatro cores? (Note que países fronteiriços não podem usar a mesma cor!)
  - Primeiro foi provado para 5 cores
  - Conjectura de 4 cores aberta até 1977,
    provada por Appel e Haken. (Prova com uso intenso de computador!)

- Problema do ciclo Hamiltoniano: considere n cidades, cada par pode ser adjacente ou não; partindo de uma cidade qualquer, existe trajeto que passe exatamente uma vez em cada cidade e retorne ao ponto de partida?
  - solução por força bruta não é satisfatória (n! permutações)
  - decidir se um grafo tem ciclo Hamiltoniano: NP-completo

Algoritmos

- "Desenvolvimento de uma técnica sistemática para resolver um desejado problema" (p.4)

- Interesse muito antigo (p. ex., Algoritmo de Euclides, 300 a.C.)

- Formalização mais recente (primeiras décadas do século passado)

- Aparecimento de computadores → - Implementação, computação automática - Muitas aplicações → - Preocupação não só com existência do algoritmo, mas também com eficiência

- Vários formatos possíveis para apresentar algoritmos
  - linguagem natural (grego, latim, português etc)
    - Euclides
    - Euler
  - diagramas diversos
  - pseudo-código, "ALGOL"

- muita flexibilidade
- Pode-se usar um conjunto de sentenças especiais para encontrar e simplificar

Em nossas apresentações de algoritmos vamos seguir a convenção do livro do Jayme. Resumidamente:

- uso de blocos com indentação
- as seguintes palavras reservadas:
  - algoritmo <comentário>
  - dados: <descrição>
  - procedimento <nome>
  - se <condição> então <...> senão <...>
  - para <variação> efetuar <...>
  - enquanto <condição> efetuar <...>
  - repetir <...> até que <condição>
- atribuições com :=
- sequências indexadas com (...) e iniciando em 1.

Por exemplo, vamos inverter uma sequência

$$s(1),\ s(2),\ \ldots,\ s(n)$$

Um algoritmo para resolver esse problema é:

Algoritmo: Inversão de uma sequência

Dados: sequência $s_1, s_2, \ldots, s_n$

para $j = 1, 2, \ldots, \lfloor \frac{n}{2} \rfloor$ efetuar

$b := s(j)$

$s(j) := s(n-j+1)$

$s(n-j+1) := b$

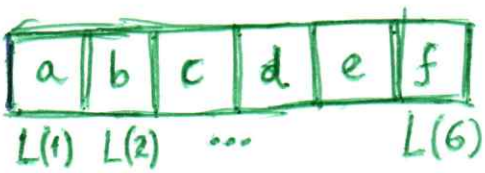
Breve revisão de Estruturas de Dados

Mais detalhes em "Estruturas de Dados e seus Algoritmos", também do Jayme.

Uma classificação baseada na representação.

- **Lista**. Notação: $L = \{a_1, \ldots, a_n\}$, onde a_i são os elementos.

  Representação em memória: alocados sequencialmente



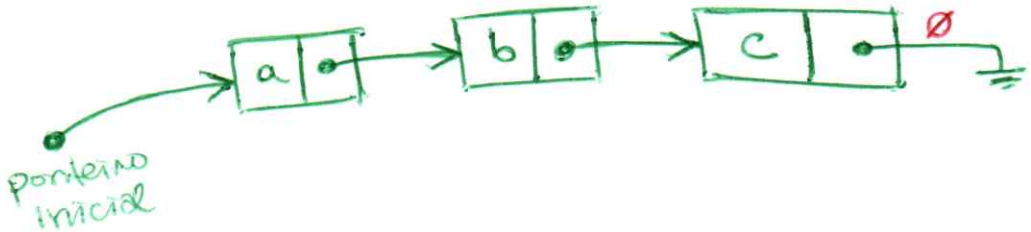
- **Lista encadeada**. Memória: elementos alocados em posições quaisquer

  Precisa associar um ponteiro a cada elemento

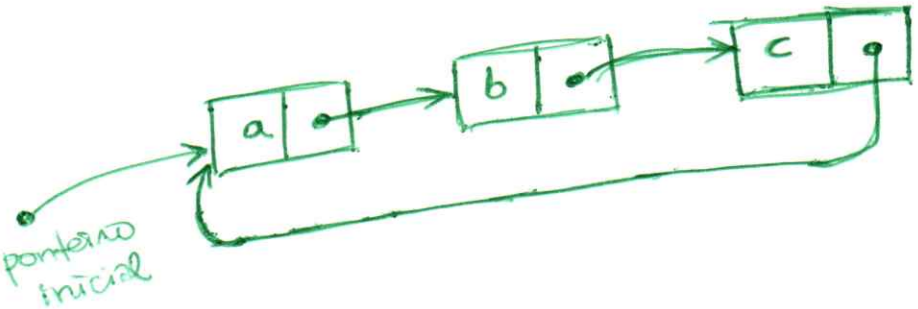
  Precisa de um ponteiro inicial.

  Um ponteiro especial, Ø, marca o fim.

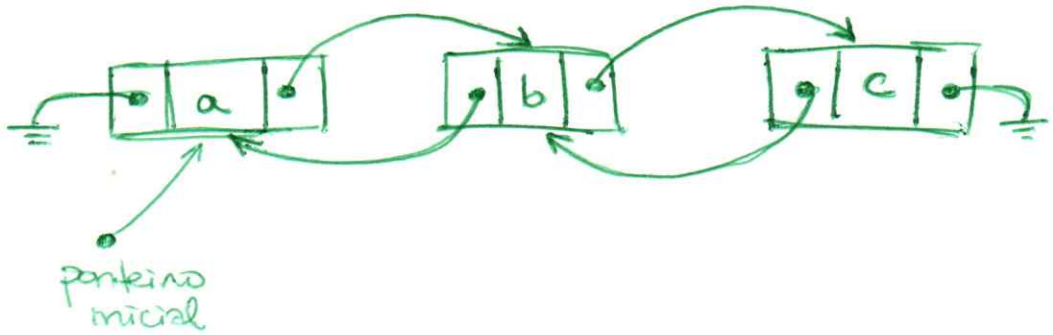
  Uma lista pode ser vazia.



- **Lista circular**. O ponteiro do último elemento aponta para o primeiro elemento (ao invés de Ø).



- Lista duplamente encadeada. - A cada elemento é associado um par de ponteiros.



Uma classificação baseada em restrições nas operações

- Pilha (stack): - inserção e remoção sempre pela mesma "extremidade"
  - primeiro a entrar, último a sair

- Fila (queue): - primeiro a entrar, primeiro a sair

- Fila dupla (deque): - inclusões e exclusões/remoções somente pelas extremidades
  - pode haver restrições somente de entrada ou somente de saída

Noções básicas de Python

- Download do interpretador em www.python.org
  Download de alguma IDE razoável: Spyder, PyCharm, etc.
  Para quem está no ~~linux~~ windows, melhor baixar logo o Anaconda.

Algoritmos

- uso de blocos de indentação
- atribuições com :=
- seqüências indexadas com ( ) e iniciando em 1.
- algoritmo <comentário>
- dados: <descrição>

Python

- uso de blocos de indentação
- atribuições com =
  (não confundir com ==)
- atribuições simultaneas permitidas, e.g. x, y = 10, 20
  x, y = y, x
- sequencias indexadas com [] e iniciando em 0 (zero).
- vamos usar um elemento extra no início da lista e começar de fato do 1.
- #algoritmo <comentário>
  ou
  """ algoritmo <comentário longo> """
- - não tem equivalente em Python
  - pode usar comentários
  - PEP484

Algoritmos

- Procedimento <nome>

- Se <condição> então <sentença>
  Caso contrário <sentença>

  -Observação
  Caso contrário se <condição>

- para <variação> efetuar <sentença>

- enquanto <condição> efetuar <sentença>

- repetir <sentença> até que <condição>

Python

- def nomeProcedimento(param):
  ...

- if <condição>:
  <sentença>
  else:
  <sentença>

  -observação:
  elif <condição>:
  <sentença>

- for i in range(ini, fim, passo):
  <sentença>

  ou

  for v in V:
  <sentença>

- while <condição>:
  <sentença>

- while True:
  <sentença>
  if <condição>:
  break