

# MC-202

## Escolhendo uma Estrutura de Dados

Rafael C. S. Schouery  
rafael@ic.unicamp.br

Universidade Estadual de Campinas

Atualizado em: 2024-07-17 11:11

## Durante o curso vimos várias EDs...

- Listas Ligadas

## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça

## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça
- Pilhas e Filas

## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça
- Pilhas e Filas
- Árvores

## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça
- Pilhas e Filas
- Árvores
  - Heaps Binários (fila de prioridade)

## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça
- Pilhas e Filas
- Árvores
  - Heaps Binários (fila de prioridade)
  - Árvores Binárias de Busca

## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça
- Pilhas e Filas
- Árvores
  - Heaps Binários (fila de prioridade)
  - Árvores Binárias de Busca
  - Árvores Rubro-Negras



## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça
- Pilhas e Filas
- Árvores
  - Heaps Binários (fila de prioridade)
  - Árvores Binárias de Busca
  - Árvores Rubro-Negras
  - Árvores B

## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça
- Pilhas e Filas
- Árvores
  - Heaps Binários (fila de prioridade)
  - Árvores Binárias de Busca
  - Árvores Rubro-Negras
  - Árvores B
- Hashing

## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça
- Pilhas e Filas
- Árvores
  - Heaps Binários (fila de prioridade)
  - Árvores Binárias de Busca
  - Árvores Rubro-Negras
  - Árvores B
- Hashing
- Grafos

## Durante o curso vimos várias EDs...

- Listas Ligadas
  - simples, duplas, circulares, com cabeça
- Pilhas e Filas
- Árvores
  - Heaps Binários (fila de prioridade)
  - Árvores Binárias de Busca
  - Árvores Rubro-Negras
  - Árvores B
- Hashing
- Grafos

Qual delas usar?

# Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

## Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens

## Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo

## Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:



## Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:
  - remoção acontece de acordo com a ordem de inserção

# Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:
  - remoção acontece de acordo com a ordem de inserção
- Filas de Prioridade:

# Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:
  - remoção acontece de acordo com a ordem de inserção
- Filas de Prioridade:
  - remoção acontece de acordo com a prioridade

# Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:
  - remoção acontece de acordo com a ordem de inserção
- Filas de Prioridade:
  - remoção acontece de acordo com a prioridade

É comum termos que usar essas EDs em algoritmos

## Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:
  - remoção acontece de acordo com a ordem de inserção
- Filas de Prioridade:
  - remoção acontece de acordo com a prioridade

É comum termos que usar essas EDs em algoritmos

- Ex: Conversão de notação infixa para pós-fixa

# Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:
  - remoção acontece de acordo com a ordem de inserção
- Filas de Prioridade:
  - remoção acontece de acordo com a prioridade

É comum termos que usar essas EDs em algoritmos

- Ex: Conversão de notação infixa para pós-fixa
- Ex: Percurso em Largura em Árvores

# Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:
  - remoção acontece de acordo com a ordem de inserção
- Filas de Prioridade:
  - remoção acontece de acordo com a prioridade

É comum termos que usar essas EDs em algoritmos

- Ex: Conversão de notação infixa para pós-fixa
- Ex: Percurso em Largura em Árvores
- Ex: Caminhos mínimos

# Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:
  - remoção acontece de acordo com a ordem de inserção
- Filas de Prioridade:
  - remoção acontece de acordo com a prioridade

É comum termos que usar essas EDs em algoritmos

- Ex: Conversão de notação infixa para pós-fixa
- Ex: Percurso em Largura em Árvores
- Ex: Caminhos mínimos

Isto é, a necessidade faz com que usemos a ED



# Estruturas de Dados Específicas

Algumas das EDs que vimos servem para tarefas específicas

- Grafos são usados quando há relações entre itens
  - Ex: rede social, ordem de tarefas, caminho mínimo
- Pilhas e Filas:
  - remoção acontece de acordo com a ordem de inserção
- Filas de Prioridade:
  - remoção acontece de acordo com a prioridade

É comum termos que usar essas EDs em algoritmos

- Ex: Conversão de notação infixa para pós-fixa
- Ex: Percurso em Largura em Árvores
- Ex: Caminhos mínimos

Isto é, a necessidade faz com que usemos a ED

- Não tem muito o que escolher...

# EDs com Inserção, Remoção e Busca

Outras EDs têm um conjunto de operações em comum

# EDs com Inserção, Remoção e Busca

Outras EDs têm um conjunto de operações em comum

- Vetores

# EDs com Inserção, Remoção e Busca

Outras EDs têm um conjunto de operações em comum

- Vetores
- Listas Ligadas

# EDs com Inserção, Remoção e Busca

Outras EDs têm um conjunto de operações em comum

- Vetores
- Listas Ligadas
- Árvores

# EDs com Inserção, Remoção e Busca

Outras EDs têm um conjunto de operações em comum

- Vetores
- Listas Ligadas
- Árvores
- Hashing

# EDs com Inserção, Remoção e Busca

Outras EDs têm um conjunto de operações em comum

- Vetores
- Listas Ligadas
- Árvores
- Hashing

Existe uma estrutura que é melhor do que as outras?

# EDs com Inserção, Remoção e Busca

Outras EDs têm um conjunto de operações em comum

- Vetores
- Listas Ligadas
- Árvores
- Hashing

Existe uma estrutura que é melhor do que as outras?

- Não de maneira geral...



## Como escolher?

Precisamos ver quais operações são necessárias

## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

Quero fazer buscas frequentes usando uma chave

## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

Quero fazer buscas frequentes usando uma chave

- vetores não ordenados e listas não são boas opções

## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

Quero fazer buscas frequentes usando uma chave

- vetores não ordenados e listas não são boas opções

Faço operações frequentes que dependem de ordem

## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

Quero fazer buscas frequentes usando uma chave

- vetores não ordenados e listas não são boas opções

Faço operações frequentes que dependem de ordem

- hash e estruturas não ordenadas não são boas opções

## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

Quero fazer buscas frequentes usando uma chave

- vetores não ordenados e listas não são boas opções

Faço operações frequentes que dependem de ordem

- hash e estruturas não ordenadas não são boas opções

Preciso remover com frequência elementos usando a chave

## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

Quero fazer buscas frequentes usando uma chave

- vetores não ordenados e listas não são boas opções

Faço operações frequentes que dependem de ordem

- hash e estruturas não ordenadas não são boas opções

Preciso remover com frequência elementos usando a chave

- vetores ordenados e listas ligadas não são boas opções



## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

Quero fazer buscas frequentes usando uma chave

- vetores não ordenados e listas não são boas opções

Faço operações frequentes que dependem de ordem

- hash e estruturas não ordenadas não são boas opções

Preciso remover com frequência elementos usando a chave

- vetores ordenados e listas ligadas não são boas opções

ABBs balanceadas são sempre boas opções

## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

Quero fazer buscas frequentes usando uma chave

- vetores não ordenados e listas não são boas opções

Faço operações frequentes que dependem de ordem

- hash e estruturas não ordenadas não são boas opções

Preciso remover com frequência elementos usando a chave

- vetores ordenados e listas ligadas não são boas opções

ABBs balanceadas são sempre boas opções

- Suportam um grande número de operações em  $O(\lg n)$

## Como escolher?

Precisamos ver quais operações são necessárias

- E qual a frequência de cada operação...

Quero fazer buscas frequentes usando uma chave

- vetores não ordenados e listas não são boas opções

Faço operações frequentes que dependem de ordem

- hash e estruturas não ordenadas não são boas opções

Preciso remover com frequência elementos usando a chave

- vetores ordenados e listas ligadas não são boas opções

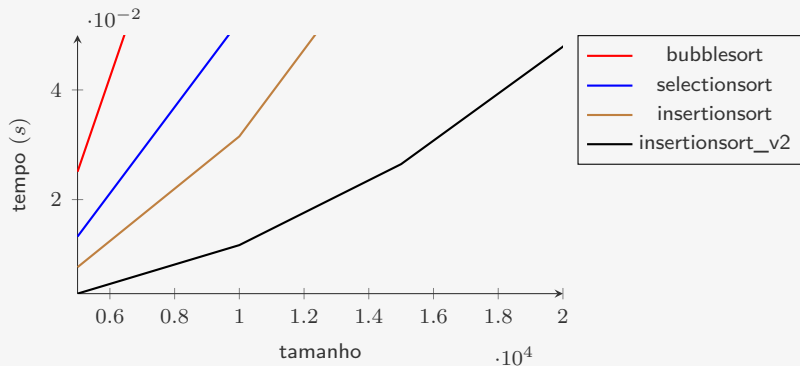
ABBs balanceadas são sempre boas opções

- Suportam um grande número de operações em  $O(\lg n)$
- Mas nem sempre são a **melhor** opção...

# Análise de tempo

Relembrando:

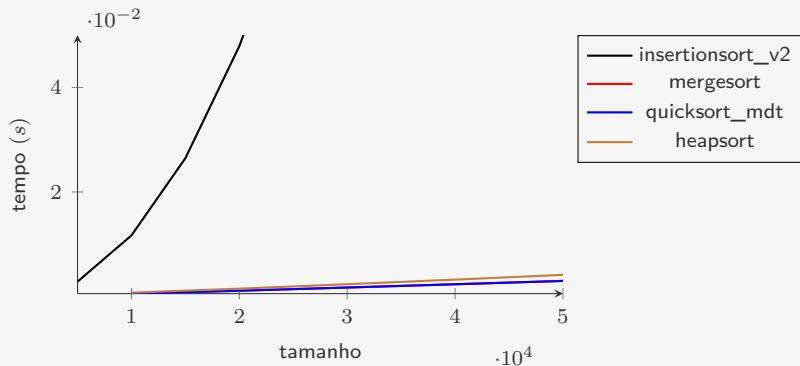
- Algoritmo  $O(n^2)$  pode ser mais rápido do que outro  $O(n^2)$
- Otimizações no código levam a programas mais rápidos
- A escolha do algoritmo é o principal fator de impacto



# Análise de tempo

Relembrando:

- Algoritmo  $O(n^2)$  pode ser mais rápido do que outro  $O(n^2)$
- Otimizações no código levam a programas mais rápidos
- A escolha do algoritmo é o principal fator de impacto
  - Algoritmo  $O(n \lg n)$  vs.  $O(n^2)$



# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

## Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio



## Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica

# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica
  - Bom se você espera que as instâncias sejam aleatórias

# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica
  - Bom se você espera que as instâncias sejam aleatórias
- A média pode ser em relação aos bits aleatórios usados

# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica
  - Bom se você espera que as instâncias sejam aleatórias
- A média pode ser em relação aos bits aleatórios usados
  - Nesse caso, o algoritmo é aleatorizado

# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica
  - Bom se você espera que as instâncias sejam aleatórias
- A média pode ser em relação aos bits aleatórios usados
  - Nesse caso, o algoritmo é aleatorizado
  - Tempo de execução diferente para a mesma instância

# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica
  - Bom se você espera que as instâncias sejam aleatórias
- A média pode ser em relação aos bits aleatórios usados
  - Nesse caso, o algoritmo é aleatorizado
  - Tempo de execução diferente para a mesma instância

Algoritmo é  $O(f(n))$  amortizado

# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica
  - Bom se você espera que as instâncias sejam aleatórias
- A média pode ser em relação aos bits aleatórios usados
  - Nesse caso, o algoritmo é aleatorizado
  - Tempo de execução diferente para a mesma instância

Algoritmo é  $O(f(n))$  amortizado

- A média das operações realizadas tem custo  $O(f(n))$

# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica
  - Bom se você espera que as instâncias sejam aleatórias
- A média pode ser em relação aos bits aleatórios usados
  - Nesse caso, o algoritmo é aleatorizado
  - Tempo de execução diferente para a mesma instância

Algoritmo é  $O(f(n))$  amortizado

- A média das operações realizadas tem custo  $O(f(n))$
- Operações mais lentas compensadas por mais rápidas



# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica
  - Bom se você espera que as instâncias sejam aleatórias
- A média pode ser em relação aos bits aleatórios usados
  - Nesse caso, o algoritmo é aleatorizado
  - Tempo de execução diferente para a mesma instância

Algoritmo é  $O(f(n))$  amortizado

- A média das operações realizadas tem custo  $O(f(n))$
- Operações mais lentas compensadas por mais rápidas

Análise empírica:

# Tipos de Análise de Tempo

Algoritmo é  $O(f(n))$  no pior caso

- Dá uma certeza sobre a qualidade do algoritmo

Algoritmo é  $O(f(n))$  no caso médio

- Podendo ser melhor ou pior para uma instância específica
  - Bom se você espera que as instâncias sejam aleatórias
- A média pode ser em relação aos bits aleatórios usados
  - Nesse caso, o algoritmo é aleatorizado
  - Tempo de execução diferente para a mesma instância

Algoritmo é  $O(f(n))$  amortizado

- A média das operações realizadas tem custo  $O(f(n))$
- Operações mais lentas compensadas por mais rápidas

Análise empírica:

- Análise estatística do tempo de execução do algoritmo

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

Árvores Splay:  $m$  inserções/buscas em  $O((n + m) \lg(n + m))$

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

Árvores Splay:  $m$  inserções/buscas em  $O((n + m) \lg(n + m))$

- Análise amortizada

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

Árvores Splay:  $m$  inserções/buscas em  $O((n + m) \lg(n + m))$

- Análise amortizada

Árvores de Busca Binária (simples) têm altura média  $O(\lg n)$

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

Árvores Splay:  $m$  inserções/buscas em  $O((n + m) \lg(n + m))$

- Análise amortizada

Árvores de Busca Binária (simples) têm altura média  $O(\lg n)$

- Considerando permutações aleatória de  $n$  chaves



## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

Árvores Splay:  $m$  inserções/buscas em  $O((n + m) \lg(n + m))$

- Análise amortizada

Árvores de Busca Binária (simples) têm altura média  $O(\lg n)$

- Considerando permutações aleatória de  $n$  chaves
- Versão aleatorizada tem altura média  $O(\lg n)$

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

Árvores Splay:  $m$  inserções/buscas em  $O((n + m) \lg(n + m))$

- Análise amortizada

Árvores de Busca Binária (simples) têm altura média  $O(\lg n)$

- Considerando permutações aleatória de  $n$  chaves
- Versão aleatorizada tem altura média  $O(\lg n)$ 
  - Média na altura em relação as execuções

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

Árvores Splay:  $m$  inserções/buscas em  $O((n + m) \lg(n + m))$

- Análise amortizada

Árvores de Busca Binária (simples) têm altura média  $O(\lg n)$

- Considerando permutações aleatória de  $n$  chaves
- Versão aleatorizada tem altura média  $O(\lg n)$ 
  - Média na altura em relação as execuções

Posso ter um algoritmo que as vezes é pior, mas que é melhor em geral?

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

Árvores Splay:  $m$  inserções/buscas em  $O((n + m) \lg(n + m))$

- Análise amortizada

Árvores de Busca Binária (simples) têm altura média  $O(\lg n)$

- Considerando permutações aleatória de  $n$  chaves
- Versão aleatorizada tem altura média  $O(\lg n)$ 
  - Média na altura em relação as execuções

Posso ter um algoritmo que as vezes é pior, mas que é melhor em geral?

- Em alguns sistemas sim

## Exemplos

Árvores Rubro-Negras: altura  $O(\lg n)$

- No pior caso

Árvores Splay:  $m$  inserções/buscas em  $O((n + m) \lg(n + m))$

- Análise amortizada

Árvores de Busca Binária (simples) têm altura média  $O(\lg n)$

- Considerando permutações aleatória de  $n$  chaves
- Versão aleatorizada tem altura média  $O(\lg n)$ 
  - Média na altura em relação as execuções

Posso ter um algoritmo que as vezes é pior, mas que é melhor em geral?

- Em alguns sistemas sim
- Em outros sistemas definitivamente não

# Informações sobre o Problema

Informações sobre o problema a ser resolvido podem ser úteis

# Informações sobre o Problema

Informações sobre o problema a ser resolvido podem ser úteis

Ex: dados a serem inseridos são praticamente aleatórios

# Informações sobre o Problema

Informações sobre o problema a ser resolvido podem ser úteis

Ex: dados a serem inseridos são praticamente aleatórios

- Posso usar ABBs ao invés de Rubro-Negra



# Informações sobre o Problema

Informações sobre o problema a ser resolvido podem ser úteis

Ex: dados a serem inseridos são praticamente aleatórios

- Posso usar ABBs ao invés de Rubro-Negra

Ex: a função de hashing não é boa para o conjunto de chaves

# Informações sobre o Problema

Informações sobre o problema a ser resolvido podem ser úteis

Ex: dados a serem inseridos são praticamente aleatórios

- Posso usar ABBs ao invés de Rubro-Negra

Ex: a função de hashing não é boa para o conjunto de chaves

- posso usar outra função de hashing

# Informações sobre o Problema

Informações sobre o problema a ser resolvido podem ser úteis

Ex: dados a serem inseridos são praticamente aleatórios

- Posso usar ABBs ao invés de Rubro-Negra

Ex: a função de hashing não é boa para o conjunto de chaves

- posso usar outra função de hashing
- ou usar uma Rubro-Negra

# Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

## Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs

# Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs
- Principalmente em Java, C++, Python, etc...

## Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs
- Principalmente em Java, C++, Python, etc...

Ex: STD — (C++ Standard Library)

## Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs
- Principalmente em Java, C++, Python, etc...

Ex: STD — (C++ Standard Library)

- array, vector (dinâmico)



# Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs
- Principalmente em Java, C++, Python, etc...

Ex: STD — (C++ Standard Library)

- `array`, `vector` (dinâmico)
- `stack`, `queue`, `deque`, `priority_queue`

## Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs
- Principalmente em Java, C++, Python, etc...

Ex: STD — (C++ Standard Library)

- `array`, `vector` (dinâmico)
- `stack`, `queue`, `deque`, `priority_queue`
- `forward_list` (simples), `list` (dupla)

## Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs
- Principalmente em Java, C++, Python, etc...

Ex: STD — (C++ Standard Library)

- `array`, `vector` (dinâmico)
- `stack`, `queue`, `deque`, `priority_queue`
- `forward_list` (simples), `list` (dupla)
- `set`, `multiset`, `unordered_set`, `unordered_multiset`

## Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs
- Principalmente em Java, C++, Python, etc...

Ex: STD — (C++ Standard Library)

- `array`, `vector` (dinâmico)
- `stack`, `queue`, `deque`, `priority_queue`
- `forward_list` (simples), `list` (dupla)
- `set`, `multiset`, `unordered_set`, `unordered_multiset`
- `map`, `multimap`, `unordered_map`, `unordered_multimap`

## Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs
- Principalmente em Java, C++, Python, etc...

Ex: STD — (C++ Standard Library)

- `array`, `vector` (dinâmico)
- `stack`, `queue`, `deque`, `priority_queue`
- `forward_list` (simples), `list` (dupla)
- `set`, `multiset`, `unordered_set`, `unordered_multiset`
- `map`, `multimap`, `unordered_map`, `unordered_multimap`

São estruturas genéricas, talvez o conhecimento do problema permita fazer algo melhor...

# Bibliotecas de Estruturas de Dados

Na maior parte das vezes, não implementamos as nossas EDs

- Usamos bibliotecas prontas de EDs
- Principalmente em Java, C++, Python, etc...

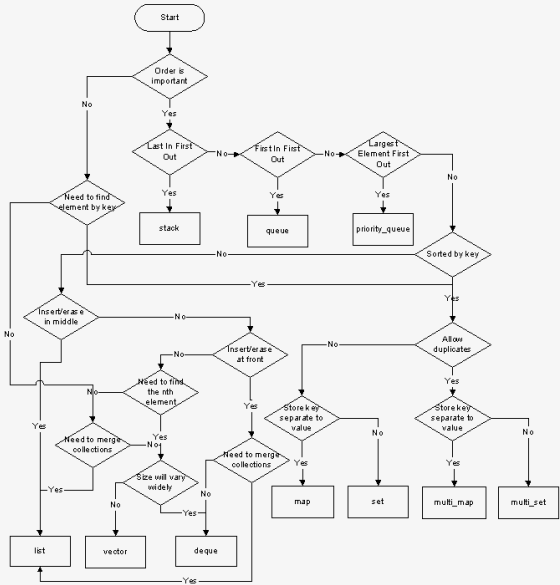
Ex: STD — (C++ Standard Library)

- `array`, `vector` (dinâmico)
- `stack`, `queue`, `deque`, `priority_queue`
- `forward_list` (simples), `list` (dupla)
- `set`, `multiset`, `unordered_set`, `unordered_multiset`
- `map`, `multimap`, `unordered_map`, `unordered_multimap`

São estruturas genéricas, talvez o conhecimento do problema permita fazer algo melhor...

- É importante entender ao invés de só usar

# Escolhendo em C++<sup>1</sup>



<sup>1</sup><http://homepages.e3.net.nz/~djm/cppcontainers.html>

# E no Python?

Algumas estruturas de dados básicas do Python 3



# E no Python?

Algumas estruturas de dados básicas do Python 3

- `list`

# E no Python?

Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade

# E no Python?

Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha

# E no Python?

Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha
- `dict`

# E no Python?

## Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha
- `dict`
  - *“A mapping object maps hashable values to arbitrary objects.”*

# E no Python?

## Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha
- `dict`
  - *“A mapping object maps hashable values to arbitrary objects.”*
- `set`

# E no Python?

## Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha
- `dict`
  - *“A mapping object maps hashable values to arbitrary objects.”*
- `set`
  - *“A set object is an unordered collection of distinct hashable objects.”*

# E no Python?

## Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha
- `dict`
  - *“A mapping object maps hashable values to arbitrary objects.”*
- `set`
  - *“A set object is an unordered collection of distinct hashable objects.”*
- `deque` (de `collections`)



# E no Python?

## Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha
- `dict`
  - *“A mapping object maps hashable values to arbitrary objects.”*
- `set`
  - *“A set object is an unordered collection of distinct hashable objects.”*
- `deque` (de `collections`)
  - funciona como `deque` ou fila

# E no Python?

## Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha
- `dict`
  - *“A mapping object maps hashable values to arbitrary objects.”*
- `set`
  - *“A set object is an unordered collection of distinct hashable objects.”*
- `deque` (de `collections`)
  - funciona como `deque` ou fila
- módulo `heapq`

# E no Python?

## Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha
- `dict`
  - *“A mapping object maps hashable values to arbitrary objects.”*
- `set`
  - *“A set object is an unordered collection of distinct hashable objects.”*
- `deque` (de `collections`)
  - funciona como `deque` ou fila
- módulo `heapq`
  - funções de fila de prioridades

# E no Python?

## Algumas estruturas de dados básicas do Python 3

- `list`
  - Cresce de acordo com a necessidade
  - Pode ser usada como uma pilha
- `dict`
  - *“A mapping object maps hashable values to arbitrary objects.”*
- `set`
  - *“A set object is an unordered collection of distinct hashable objects.”*
- `deque` (de `collections`)
  - funciona como `deque` ou fila
- módulo `heapq`
  - funções de fila de prioridades

E é possível encontrar outras bibliotecas...

## Exercício

Vamos discutir qual ED usar nos seguintes problemas:

## Exercício

Vamos discutir qual ED usar nos seguintes problemas:

1. Tabela de alunos da DAC

## Exercício

Vamos discutir qual ED usar nos seguintes problemas:

1. Tabela de alunos da DAC
2. Tabela de símbolos do compilador

## Exercício

Vamos discutir qual ED usar nos seguintes problemas:

1. Tabela de alunos da DAC
2. Tabela de símbolos do compilador
3. Sistema de arquivos



# Exercício

Vamos discutir qual ED usar nos seguintes problemas:

1. Tabela de alunos da DAC
2. Tabela de símbolos do compilador
3. Sistema de arquivos
4. Tabela de aberturas de xadrez

# Disciplinas da Computação

MC322 — Programação Orientada a Objetos

# Disciplinas da Computação

MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores

# Disciplinas da Computação

## MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores
- De maneira organizada...

# Disciplinas da Computação

MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores
- De maneira organizada...

Disciplinas da Teoria da Computação

# Disciplinas da Computação

MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores
- De maneira organizada...

Disciplinas da Teoria da Computação

- MC358 — Fundamentos Matemáticos da Computação

# Disciplinas da Computação

## MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores
- De maneira organizada...

## Disciplinas da Teoria da Computação

- MC358 — Fundamentos Matemáticos da Computação
  - base teórica para análise de algoritmos

# Disciplinas da Computação

## MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores
- De maneira organizada...

## Disciplinas da Teoria da Computação

- MC358 — Fundamentos Matemáticos da Computação
  - base teórica para análise de algoritmos
  - matemática discreta



# Disciplinas da Computação

## MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores
- De maneira organizada...

## Disciplinas da Teoria da Computação

- MC358 — Fundamentos Matemáticos da Computação
  - base teórica para análise de algoritmos
  - matemática discreta
- MC458 — Projeto e Análise de Algoritmos I

# Disciplinas da Computação

## MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores
- De maneira organizada...

## Disciplinas da Teoria da Computação

- MC358 — Fundamentos Matemáticos da Computação
  - base teórica para análise de algoritmos
  - matemática discreta
- MC458 — Projeto e Análise de Algoritmos I
  - Notação assintótica e análise de algoritmos

# Disciplinas da Computação

## MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores
- De maneira organizada...

## Disciplinas da Teoria da Computação

- MC358 — Fundamentos Matemáticos da Computação
  - base teórica para análise de algoritmos
  - matemática discreta
- MC458 — Projeto e Análise de Algoritmos I
  - Notação assintótica e análise de algoritmos
- MC558 — Projeto e Análise de Algoritmos II

# Disciplinas da Computação

## MC322 — Programação Orientada a Objetos

- Como desenvolver sistemas computacionais maiores
- De maneira organizada...

## Disciplinas da Teoria da Computação

- MC358 — Fundamentos Matemáticos da Computação
  - base teórica para análise de algoritmos
  - matemática discreta
- MC458 — Projeto e Análise de Algoritmos I
  - Notação assintótica e análise de algoritmos
- MC558 — Projeto e Análise de Algoritmos II
  - Algoritmos em Grafos