5

Tecnologia em Análise e Desenvolvimento de Sistemas

Algoritmos e Estruturas de Dados

Introdução e TADs

Prof. Ramide Dantas

Algoritmos

Os algoritmos fazem parte do dia-a-dia das pessoas:

- Instruções para o uso de medicamentos;
- Indicações de como montar um aparelho;
- Uma receita de culinária.

Sequência de ações executáveis para a obtenção de uma solução para um determinado tipo de problema

- Segundo Dijkstra, um algoritmo corresponde a uma descrição de um padrão de comportamento, expresso em termos de um conjunto finito de ações
- Ex.: executando a operação a + b percebemos um padrão de comportamento, mesmo que a operação seja realizada para valores diferentes de a e b.

Estruturas de Dados

Estrutras de Dados definem como os dados estão organizados e como são operados

Para resolver um problema é necessário escolher um conjunto de dados que representa a realidade, em seguida a forma de representar esses dados.

A escolha da representação dos dados é determinada, entre outras, pelas operações a serem realizadas sobre os dados.

Estruturas e Algoritmos estão intimamente ligados:

Não se pode estudar estruturas de dados sem considerar os algoritmos associados a elas,

Assim como a escolha dos algoritmos em geral depende da representação e da estrutura dos dados.

Programas

"Programa = estruturas de dados + algoritmos"

Programas são formulações concretas de algoritmos abstratos, baseados em representações e estruturas específicas de dados.

Programas representam uma classe especial de algoritmos capazes de serem seguidos por computadores.

Tipos de Dados

Caracteriza o conjunto de valores a que uma constante pertence, ou que podem ser assumidos por uma variável ou expressão, ou que podem ser gerados por uma função.

Tipos simples de dados são grupos de valores indivisíveis (por exemplo: int, char e float, em C).

> Exemplo: uma variável do tipo boolean pode assumir o valor verdadeiro ou o valor falso, e nenhum outro valor.

Os tipos estruturados em geral definem uma coleção de valores simples, ou um agregado de valores de tipos diferentes.

Tipos Abstratos de Dados (TAD)

Modelo matemático, acompanhado das operações definidas sobre o modelo.

Exemplo: o conjunto dos inteiros acompanhado das operações de adição, subtração e multiplicação.

TADs são utilizados como base para o projeto de algoritmos e estruturas de dados

A implementação do algoritmo em uma linguagem de programação exige a representação do TAD em termos dos tipos de dados e dos operadores suportados.

A representação do modelo matemático por trás do TAD é realizada mediante uma estrutura de dados.

Exemplos de TAD

Exemplo de TAD: Lista.

Seguintes operações definidas para Lista:

- Faça a lista vazia;
- Obtenha o primeiro elemento da lista; se a lista estiver vazia, então retorne nulo;
- Insira um elemento na lista.

Há várias opções de estruturas de dados que permitem uma implementação eficiente para listas (por ex., o tipo estruturado arranjo).

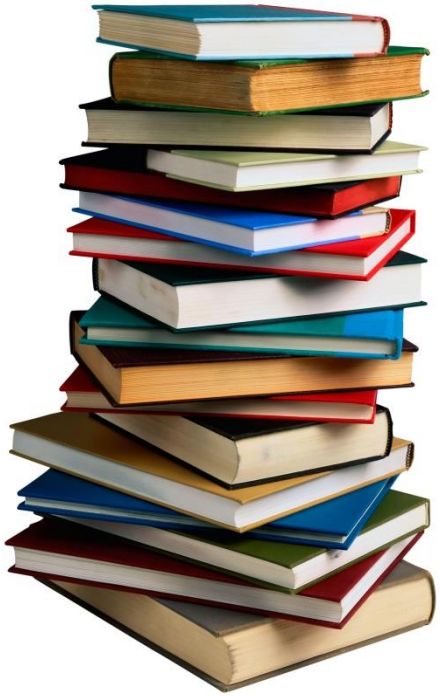
Implementação de TADs

Cada operação do tipo abstrato de dados é implementada como um procedimento na linguagem de programação escolhida.

Qualquer alteração na implementação do TAD fica restrita à parte encapsulada, sem causar impactos em outras partes do código.

A escolha adequada de uma implementação depende fortemente das operações a serem realizadas sobre o modelo.

Pilhas / Stacks



A política de inserção e remoção na Pilha é conhecida como "*LIFO*": *Last In, First Out*

Pilhas / Stacks

Dada uma pilha *P*, podemos definir as operações:

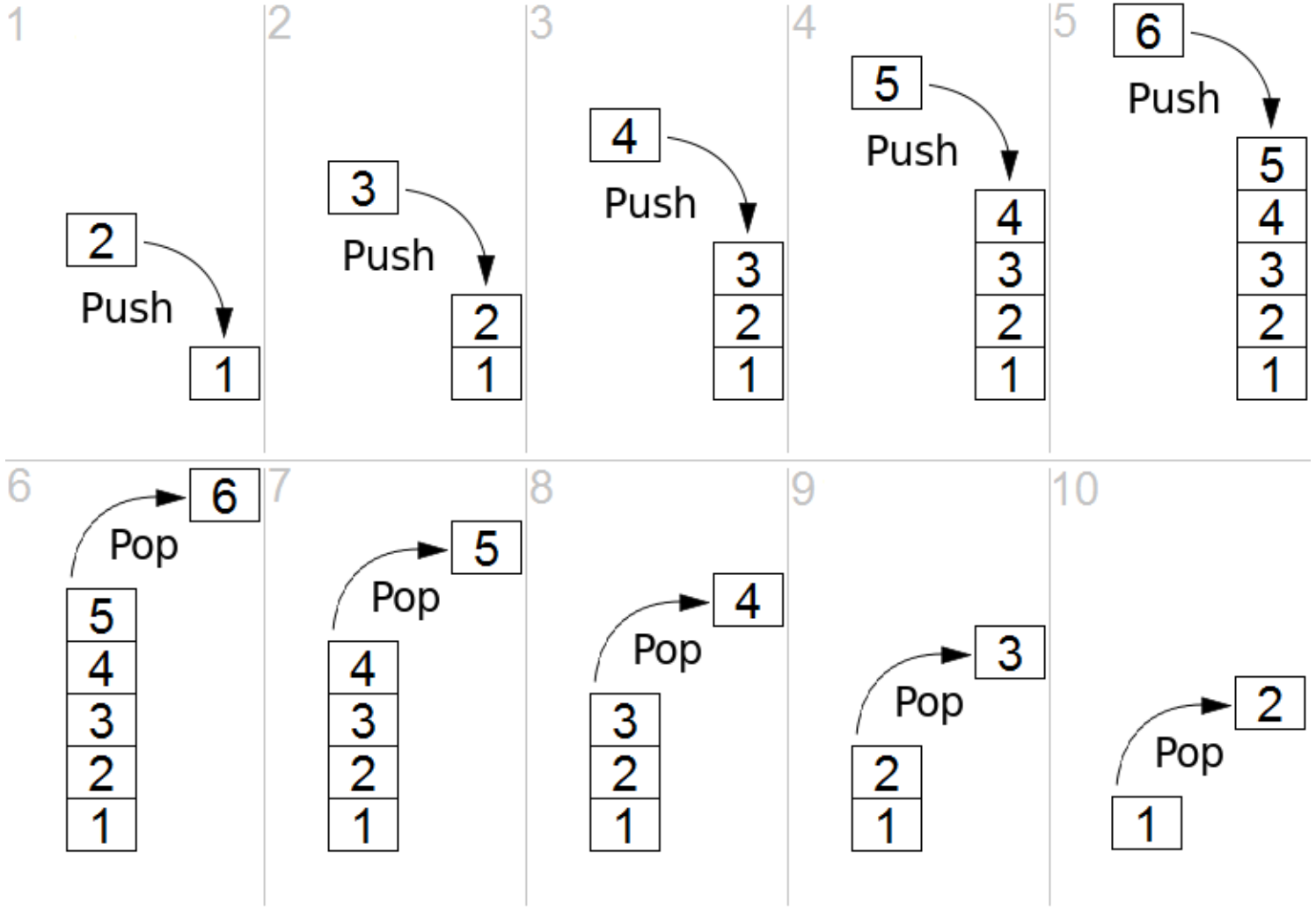
Empilha/push (dado *v*, pilha *P*): acrescenta o dado *v* no topo da pilha. Pode ocasionar *overflow*

Desempilha/pop (pilha *P*): descarta o dado mais recentemente empilhado (no topo da pilha). Pode ocasionar *underflow*

Altura (pilha *P*): retorna o número de elementos de *P*

Topo/top/peek (pilha *P*): retorna o dado mais recentemente empilhado. Definida apenas se *Altura* (*P*) > 0

Pilhas / Stacks



By Maxtremus - Own work, CC0, https://commons.wikimedia.org/w/index.php?curid=44458752

Implementando Pilhas

Assumimos que uma pilha *P* tem os seguintes campos/componentes:

P.max = número máximo de dados da pilha

P.A [0 .. *P.max* – 1] = array com *P.max* elementos

P.n = número de elementos presentes na pilha (inicialmente 0)

Os dados estão em *P.A*[0 .. *P.n* – 1], na ordem em que foram empilhados:

P.A [0] é o dado mais antigo

P.A [*P.n* – 1] é o dado mais recente

Implementando Pilhas

```
proc empilha (dado v, pilha P) {
    se P.n < P.max então {
        P.A [P.n] ← v
        P.n ← P.n + 1
    }
     senão reportar ("Overflow")
}
```

```
proc desempilha (pilha P) {
    se P.n > 0 então P.n ← P.n – 1
    senão reportar ("Underflow")
}
```

```
proc altura (pilha P) {
    retornar (P.n)
}
```

```
proc topo (pilha P) {
    retornar (P.A [P.n – 1])
}
```

Filas / Queues

A política de inserção e remoção de dados na fila é conhecida como "*FIFO*" – *First In First Out*

Filas / Queues

São comumente definidas as seguintes operações sobre filas:

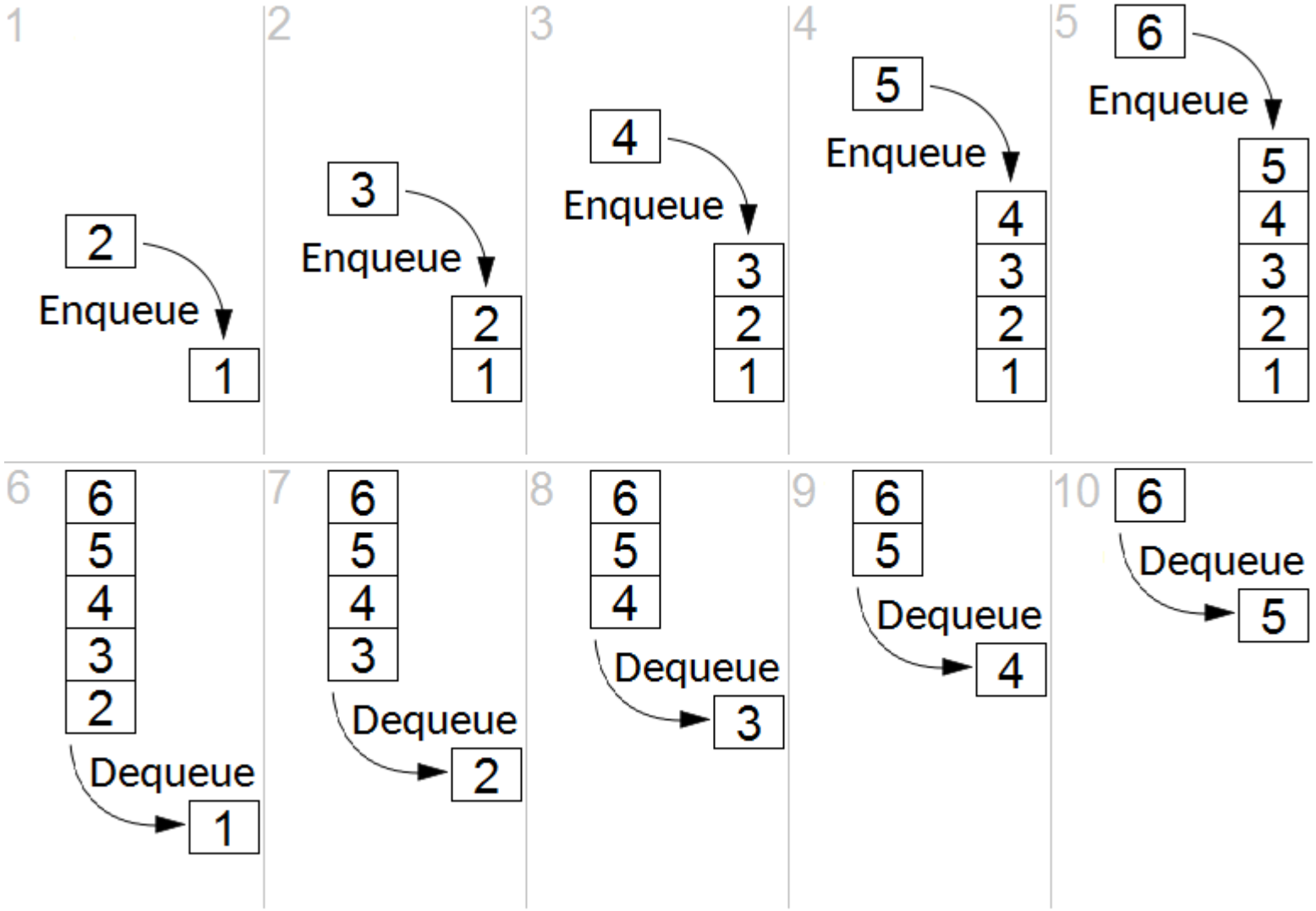
Enfileira / enqueue (dado *v*, fila *F*) : *v* é posto na fila *F*

Desenfileira / dequeue (fila *F*) : descarta o dado mais antigo (menos recentemente enfileirado) da fila *F*

Comprimento (fila *F*) : retorna o número de elementos na fila *F*

Próximo (fila *F*) : retorna o dado mais antigo da fila *F*

Filas / Queues



Implementando Filas

Uma fila *F* pode ser implementada usando uma estrutura com os seguintes campos

F.max = número máximo de dados

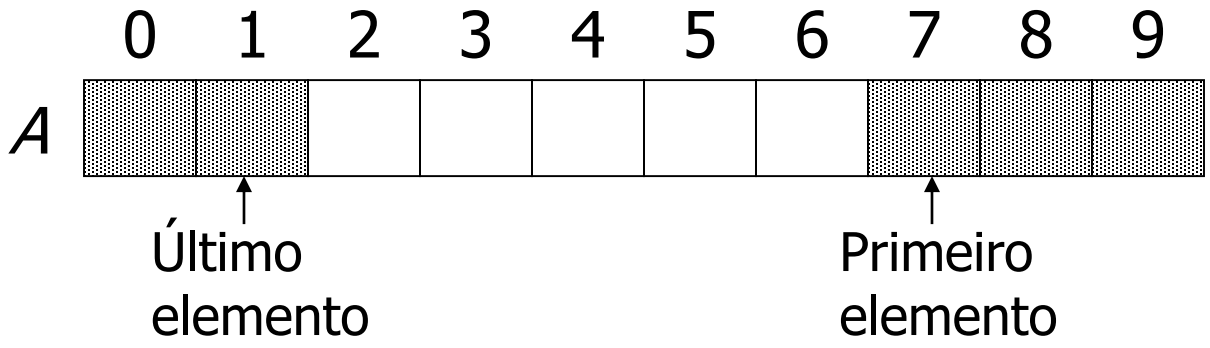
F.A [0 .. *F.max*–1] = array onde os dados são postos

F.início = índice do 1º elemento da fila (inicialmente 0)

F.n = número de elementos da fila

Implementando Filas

Os elementos da fila são armazenados consecutivamente a partir de *F.A* [*F.início*] podendo "dar a volta" e continuar a partir de *F.A*[0].



max = 10
início = 7
n = 5

Implementando Filas

```
proc enfileira (dado v, fila F) {
    se F.n < F.max então {
        F.A [(F.início + F.n) mod F.max] ← v
        F.n ← F.n + 1
    }
    senão reportar ("Overflow")
}
```

```
proc desenfileira (fila F) {
    se F.n > 0 então {
        F.início ← (F.início + 1) mod F.max
        F.n ← F.n – 1
    }
    senão reportar ("Underflow")
}
```

```
proc comprimento (fila F)
{
    retornar F.n
}
```

```
proc próximo (fila F) {
    retornar F.A [F.início]
}
```

Outros TADs

Set (Conjunto)

Conjunto não ordenado de elementos únicos.

Operações típicas: inclusão, remoção, pertencimento, união (entre conjuntos), etc.

Bag (Multiset)

Conjunto não ordenado de elementos que permite repetição.

Operações: similar ao set.

Map (Mapa)

Mapeia um elemento (chave) em outro (valor).

Operações típicas: inserir (chave, valor), remover, pegar pela chave, etc.

5

Tecnologia em Análise e Desenvolvimento de Sistemas

Algoritmos e Estruturas de Dados

Introdução e TADs

Prof. Ramide Dantas