Estruturas de Dados II: Notação *O* e Classes de Complexidade*

Talles Brito Viana

*Slides baseados a partir dos slides elaborados por Charles Ornelas Almeida, Israel Guerra e Nivio Ziviani para o Livro "Projeto de Algoritmos", Nivio Ziviani, Editora CENGAGE Learning, 2010.

Análise de uma classe de algoritmos

- Qual é o algoritmo de menor custo possível para resolver um problema particular?
  - Toda uma família de algoritmos é investigada.
  - Procura-se identificar um que seja o melhor possível.
    - Coloca-se limites para a complexidade computacional dos algoritmos pertencentes à classe.
    - Quando o custo de um algoritmo é igual ao menor custo possível, o algoritmo é ótimo para a medida de custo considerada.

f(n)

- Para medir o custo de execução de um algoritmo é comum definir uma função de custo ou função de complexidade *f*.
- *f(n)* é a medida do tempo necessário para executar um algoritmo para um problema de tamanho *n*.
  - A complexidade de tempo na realidade não representa tempo diretamente, mas o número de vezes que determinada operação considerada relevante é executada.

Notação *O*

- A complexidade de algoritmos analisa o comportamento do tempo f em função de uma entrada n que tende ao infinito.
- O comportamento assintótico (no infinito) de pior caso de um algoritmo é representado pela notação O.
  - Quando dizemos que $f = O(g(n))$, a função de tempo f é limitada superiormente pela função $g(n)$.
  - O também chamado como Big O

Classes Assintóticas

- Constante: *O(1)*
  - Uso do algoritmo independe de *n*.
- Logarítmica: *O(lg(n))*
  - Típico em algoritmos que transformam um problema em outros menores.
- Linear: *O(n)*
  - Em geral, um pequeno trabalho é realizado sobre cada elemento de entrada.
  - É a melhor situação possível para um algoritmo que tem de processar/produzir n elementos de entrada/saída.
  - Cada vez que n dobra de tamanho, o tempo de execução dobra.

- *Lê-se "O de n", "O de 1", "O de lg(n)"...*

Classes Assintóticas

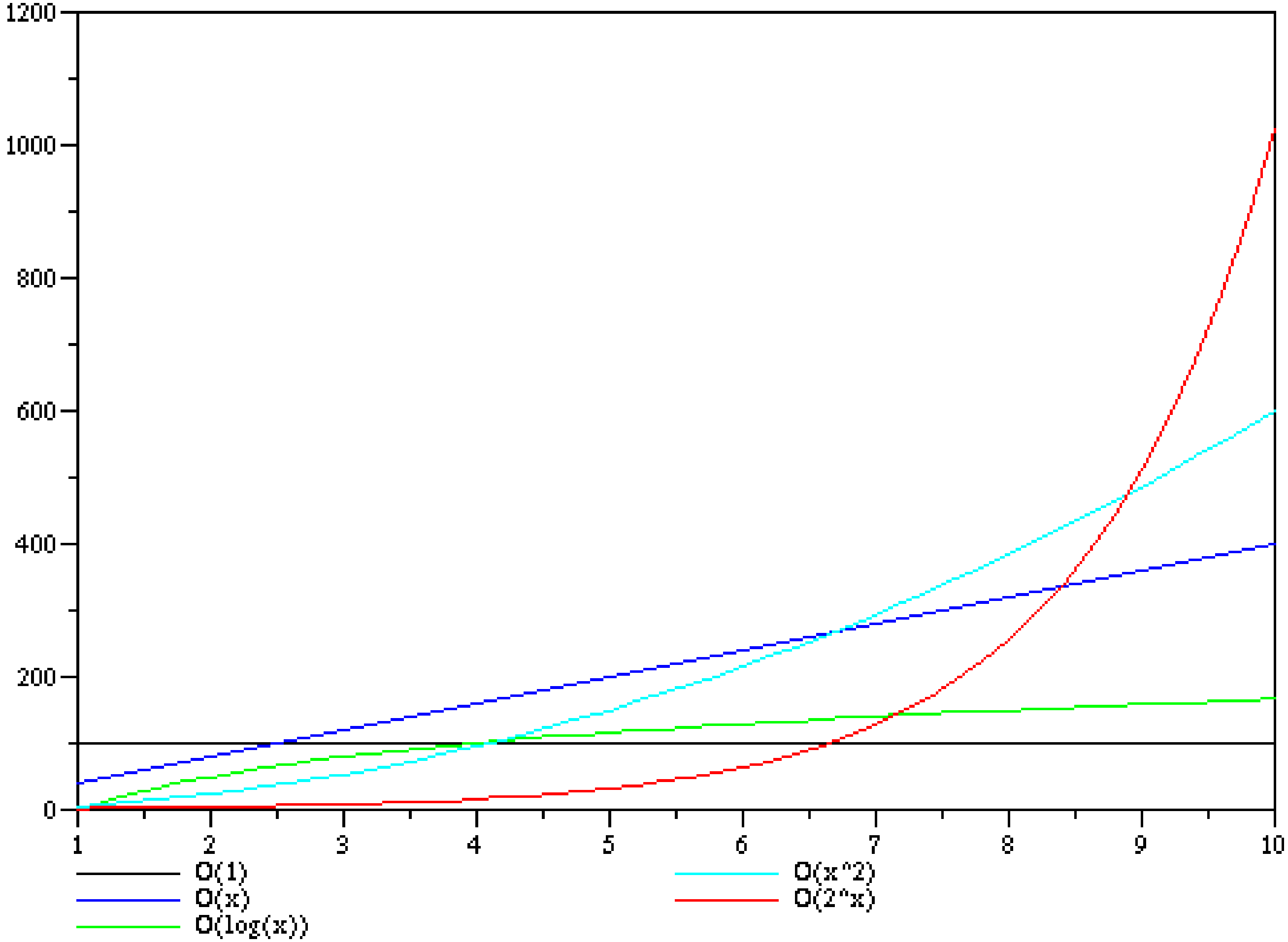
- Logaritmo-linear: $O(n\ lg(n))$
  - Típico em algoritmos que quebram um problema em outros menores,resolvem cada um deles independentemente e ajuntando as soluções depois.
- Quadrática: $O(n^2)$
  - Ocorrem quando os itens de dados são processados aos pares, muitas vezes em um anel dentro de outro.
  - Sempre que n dobra, o tempo de execução é multiplicado por 4.
  - Úteis para resolver problemas de tamanhos relativamente pequenos.
- Cúbica: $O(n^3)$
  - Úteis apenas para resolver pequenos problemas.... **porque**?

Classes Assintóticas

- Exponencial: $O(c^n)$ onde c é uma constante
  - Geralmente não são úteis sob o ponto de vista prático.
  - Ocorrem na solução de problemas quando se usa força bruta para resolvê-los.

- Dizemos que:
  - $O(1) < O(lg(n)) < O(n) < O(n\ lg(n)) < O(n^2) < O(n^3) \ldots$

Comportamento das Classes

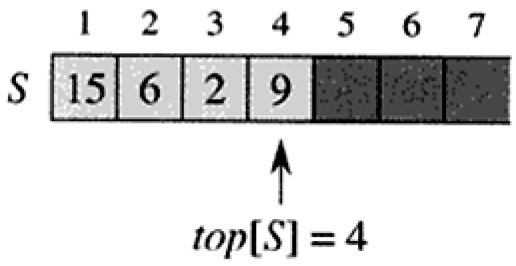


Exemplo: Pilhas

- O elemento eliminado é o mais recentemente inserido.
  - Implementa a norma: *último a entrar, primeiro a sair.*
- Operações básicas de uma pilha:
  - PUSH (inserção)
  - POP (remoção)
    - Alusão à pratos de restaurantes...

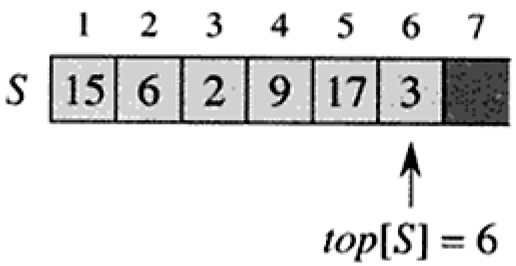
Pilhas

- Podemos implementar uma pilha de no máximo *n* elementos em um conjunto *S* de *n* posições *de* memória.



(a)

- Pilha de no máximo 7 elementos.
- A pilha tem 4 elementos.
- *top*[S] marca o topo da pilha.
- O elemento do topo é 9.

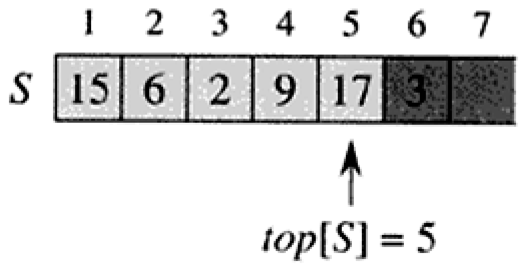


(b)

- PUSH(S,17)
- PUSH(S,3)

Pilhas

- Podemos implementar uma pilha de no máximo n elementos em um conjunto S de n posições *de* memória.



(c)

- POP(S) retorna 3
- Apesar de 3 não ter sido apagado do conjunto, não é mais possível acessar tal elemento em memória.

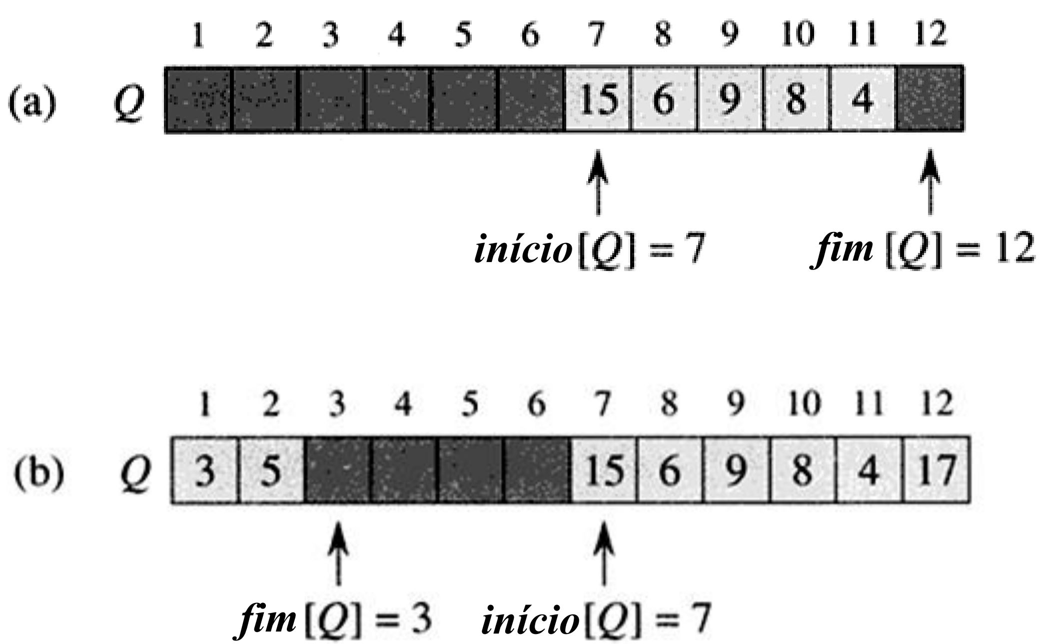
- Perceba que, as operações PUSH e POP têm tempo de execução constante = *O(1)*

Exemplo: Filas

- O elemento eliminado é sempre o que estiver pelo tempo mais longo.
  - Implementa a norma: *primeiro a entrar, primeiro a sair*.
- Operações básicas de uma fila:
  - ENQUEUE (ENFILEIRAR)
  - DEQUEUE (DESINFILEIRAR)
    - Alusão à filas de bancos...

Filas

- Podemos implementar uma fila de no máximo *n-1* elementos em um conjunto *Q* de *n* posições de memória.

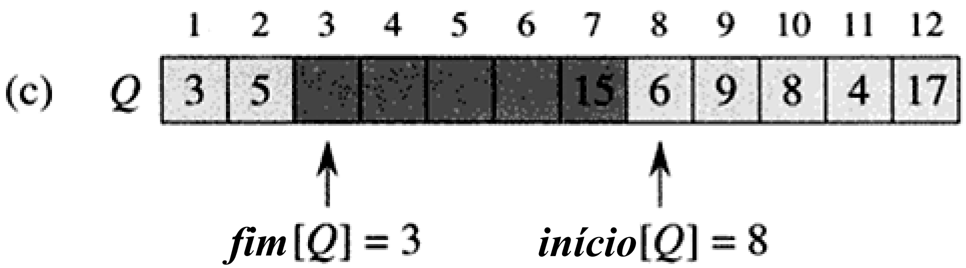


- *início*[Q] marca o início da fila
- *fim*[Q] marca o fim da fila
- Se *início*[Q] = *fim*[Q] a fila está vazia, inicialmente, temos que *início*[Q] = fim[Q] = 1.
- Se início[Q] = fim[Q] + 1, então a fila está cheia.

- ENFILEIRAR(Q,17)
- ENFILEIRAR(Q,3)
- ENFILEIRAR(Q,5)

Filas

- Podemos implementar uma fila de no máximo *n-1* elementos em um conjunto *Q* de *n* posições de memória.

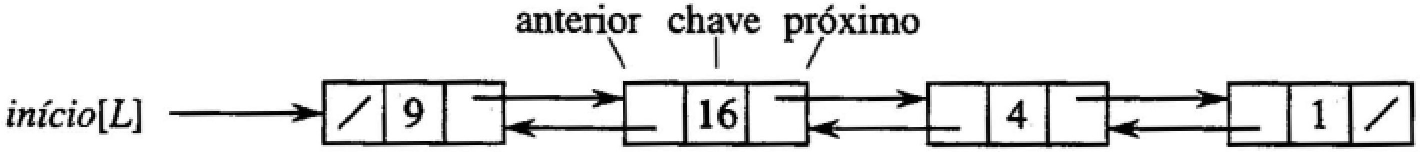


- DESINFILEIRAR(Q) retorna 15.
- Novo *início*[Q] marca 8.
- Apesar de 15 não ter sido apagado do conjunto, não é mais possível acessar tal elemento em memória.

- Perceba que, as operações ENFILEIRAR e DESINFILEIRAR têm tempo de execução constante = *O(1)*

Exemplo: Lista Ligada

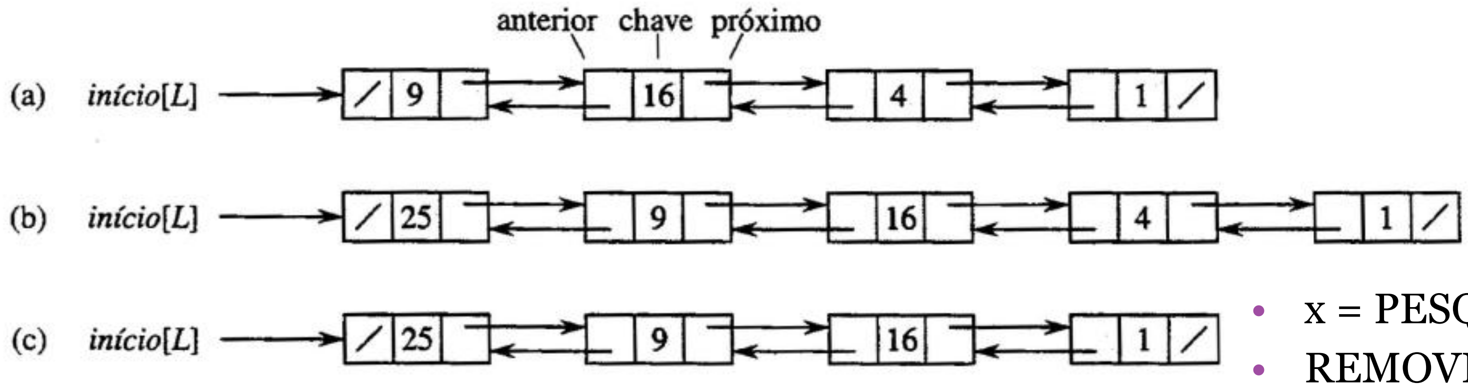
- Estrutura de dados em que os objetos estão organizados em uma ordem linear.
  - A ordem de uma lista ligada é determinada por um ponteiro em cada objeto.
- Uma lista duplamente ligada *L* contém um campo *chave* que armazena um dado e dois ponteiros.
  - *próximo*[x] aponta para o elemento sucessor de x.
  - *anterior*[x] aponta para o elemento predecessor de x.
  - *início*[L] aponta para o primeiro elemento da lista L.



- Vale ressaltar que, filas e pilhas também poderiam ser implementadas utilizando listas.

Lista Ligada

- Operações básicas de uma lista:
  - PESQUISAR, INSERIR, REMOVER



- INSERIR(L,25)
- x = PESQUISAR(L,4)
- REMOVER(L,x)

- Se a inserção de um novo objeto é feita no início da lista, o tempo de inserção é constante = O(1).
- No pior caso, o tempo de pesquisa depende da quantidade *n* de objetos da lista = O(n).
- A operação de remoção (sem pesquisa) executa ajustes de ponteiros de um objeto, logo, tem tempo constante = O(1).