Projeto e Análise de Algoritmos

Estruturas de dados elementares
Aula 11

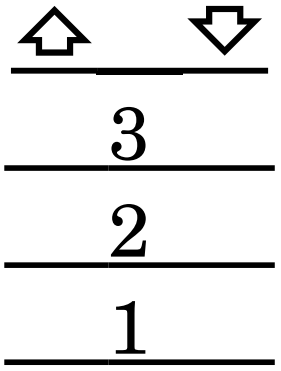
Diane Castonguay
diane@inf.ufg.br

Instituto de Informática
Universidade Federal de Goiás

Pilhas

STACK-EMPTY(S)

1. se topo[S] = 0
2. então retorna TRUE
3. senão retorna FALSE
4. fim-se



Pilhas

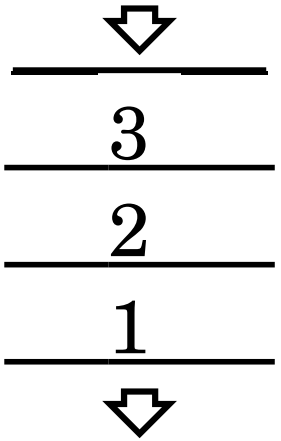
PUSH(S, x)
1. topo[S] ← topo S +1
2. S[topo[S]] ← x

POP(S)
1. se STACK-EMPTY(S)
2. então **error** “underflow”
3. senão topo[S] ← *topo* S – 1
4. retorna S[*topo*[S] + 1]
5. fim-se

Filas

ENQUEUE(Q, x)

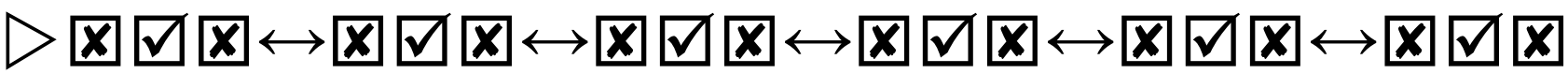
1. Q[*fim*[Q]] ← x
2. se *fim*[Q] = *comprimento*[Q]
3. então *fim*[Q] ← 1
4. senão *fim*[Q] ← *fim*[Q] + 1
5. fim-se



Filas

DEQUEUE(Q)
1. $x \leftarrow$ Q[*início*[Q]]
2. se inicio[Q] = *comprimento*[Q]
3. então *início*[Q] $\leftarrow$ 1
4. senão *início*[Q] $\leftarrow$ *início*[Q] + 1
5. fim-se
6. retorna x

Listas ligadas



LIST-SEARCH(L, k)

1. $x \leftarrow$ *início*[L]
2. enquanto $x \neq$ NIL e *chave*[x] $\neq$ k faça
3. $x \leftarrow$ *próximo*[x]
4. fim-enquanto
5. retorna x

Listas ligadas

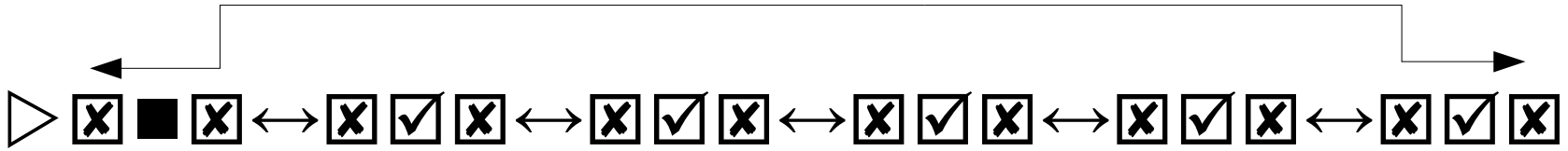
LIST-INSERT(L, x)
1. *próximo*[x] ← *início*[L]
2. se *início*[L] ≠ NIL
3. então *anterior*[*início*[L]] ← x
4. fim-se
5. *início*[L] ← x
6. *anterior*[x] ← NIL

Listas ligadas

LIST-DELETE(L, x)

1. se *anterior*[x] ≠ NIL
2. então *próximo*[*anterior*[x]] ← *próximo*[x]
3. senão *início*[L] ← *próximo*[x]
4. fim-se
5. se *próximo*[x] ≠ NIL
6. então *anterior*[*próximo*[x]] ← *anterior*[x]
7. fim-se

Sentinelas



LIST-SEARCH'(L, k)

1. $x \leftarrow$ *próximo*[*nulo*[L]]
2. enquanto $x \neq$ *nulo*[L] e *chave*[x] $\neq$ k faça
3. $x \leftarrow$ *próximo*[x]
4. fim-enquanto
5. retorna x

Sentinelas

LIST-INSERT'(L, x)

1. *próximo*[x] ← *próximo*[*nulo*[L]]
2. an*terior*[*próximo*[*nulo*[L]]] ← x
3. *próximo*[*nulo*[L]] ← x
4. *anterior*[x] ← *nulo*[L]

LIST-DELETE'(L, x)

1. *próximo*[*anterior*[x]] ← *próximo*[x]
2. an*terior*[*próximo*[x]] ← *anterior*[x]