

Algorytmy i struktury danych - Lista z dowiązaniem

Marcin Żurowski

30 listopada 2025

Plan zajęć

- 1 Lista z dowiązaniem
- 2 Lista jednokierunkowa
- 3 Lista dwukierunkowa

Lista z dowiązaniem

Struktura, w której elementy są ułożone w liniowym porządku. Wstawianie i usuwanie elementów jest możliwe w każdym miejscu listy.

Lista z dowiązaniem zaimplementowana za pomocą wskaźników:

- lista jednokierunkowa - każdy jej element posiada wskaźnik do swojego następnika
- lista dwukierunkowa - każdy jej element posiada wskaźnik do swojego następnika, oraz wskaźnik do swojego poprzednika

Lista z dowiązaniem

Struktura, w której elementy są ułożone w liniowym porządku. Wstawianie i usuwanie elementów jest możliwe w każdym miejscu listy.

Lista z dowiązaniem zaimplementowana za pomocą wskaźników:

- lista jednokierunkowa - każdy jej element posiada wskaźnik do swojego następnika
- lista dwukierunkowa - każdy jej element posiada wskaźnik do swojego następnika, oraz wskaźnik do swojego poprzednika

Lista z dowiązaniem

Struktura, w której elementy są ułożone w liniowym porządku. Wstawianie i usuwanie elementów jest możliwe w każdym miejscu listy.

Lista z dowiązaniem zaimplementowana za pomocą wskaźników:

- lista jednokierunkowa - każdy jej element posiada wskaźnik do swojego następnika
- lista dwukierunkowa - każdy jej element posiada wskaźnik do swojego następnika, oraz wskaźnik do swojego poprzednika

Lista z dowiązaniem

Struktura, w której elementy są ułożone w liniowym porządku. Wstawianie i usuwanie elementów jest możliwe w każdym miejscu listy.

Lista z dowiązaniem zaimplementowana za pomocą wskaźników:

- lista jednokierunkowa - każdy jej element posiada wskaźnik do swojego następnika
- lista dwukierunkowa - każdy jej element posiada wskaźnik do swojego następnika, oraz wskaźnik do swojego poprzednika

Lista z dowiązaniem

Element x listy jednokierunkowej L składa się z dwóch pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$.

Lista z dowiązaniem

Element x listy jednokierunkowej L składa się z dwóch pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem

Element x listy jednokierunkowej L składa się z dwóch pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem

Element x listy jednokierunkowej L składa się z dwóch pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem

Element x listy jednokierunkowej L składa się z dwóch pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem - metody

Wyszukiwanie w liście jednokierunkowej L elementu o kluczu k:

```
procedure LIST1-SEARCH(L,k)
  x = L.head
  while x  $\neq$  NIL and x.key  $\neq$  k
    x = x.next
  return x
```

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniem - metody

Wyszukiwanie w liście jednokierunkowej L elementu o kluczu k:

```
procedure LIST1-SEARCH(L,k)
  x = L.head
  while x  $\neq$  NIL and x.key  $\neq$  k
    x = x.next
  return x
```

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniem - metody

Wyszukiwanie w liście jednokierunkowej L elementu o kluczu k:

```
procedure LIST1-SEARCH(L,k)
  x = L.head
  while x  $\neq$  NIL and x.key  $\neq$  k
    x = x.next
  return x
```

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniem - metody

Wstawianie elementu x (którego pole *key* zostało wcześniej zainicjowane) na początek listy L :

```
procedure LIST1-INSERT( $L, x$ )  
   $x.next = L.head$   
   $L.head = x$ 
```

Operacji wykonuje się w czasie $\Theta(1)$

Lista z dowiązaniem - metody

Wstawianie elementu x (którego pole *key* zostało wcześniej zainicjowane) na początek listy L :

```
procedure LIST1-INSERT( $L, x$ )  
   $x.next = L.head$   
   $L.head = x$ 
```

Operacji wykonuje się w czasie $\Theta(1)$

Lista z dowiązaniem - metody

Wstawianie elementu x (którego pole *key* zostało wcześniej zainicjowane) na początek listy L :

```
procedure LIST1-INSERT( $L, x$ )  
   $x.next = L.head$   
   $L.head = x$ 
```

Operacji wykonuje się w czasie $\Theta(1)$

Lista z dowiązaniem

Oprócz funkcji wstawiającej element x na początku listy L można zaimplementować wstawianie elementu x na przykład:

- na koniec listy L
- bezpośrednio za elementem y
- bezpośrednio przed elementem y
- na pozycji i na liście

Lista z dowiązaniem

Oprócz funkcji wstawiającej element x na początku listy L można zaimplementować wstawianie elementu x na przykład:

- na koniec listy L
- bezpośrednio za elementem y
- bezpośrednio przed elementem y
- na pozycji j na liście

Lista z dowiązaniem

Oprócz funkcji wstawiającej element x na początku listy L można zaimplementować wstawianie elementu x na przykład:

- na koniec listy L
- bezpośrednio za elementem y
- bezpośrednio przed elementem y
- na pozycji j na liście

Lista z dowiązaniem

Oprócz funkcji wstawiającej element x na początku listy L można zaimplementować wstawianie elementu x na przykład:

- na koniec listy L
- bezpośrednio za elementem y
- bezpośrednio przed elementem y
- na pozycji j na liście

Lista z dowiązaniem

Oprócz funkcji wstawiającej element x na początku listy L można zaimplementować wstawianie elementu x na przykład:

- na koniec listy L
- bezpośrednio za elementem y
- bezpośrednio przed elementem y
- na pozycji j na liście

Lista z dowiązaniem - metody

Usuwanie elementu x z listy jednokierunkowej L :

procedure LIST1-DELETE(L, x)

if $x \neq L.head$

$y = L.head$

while $y.next \neq x$

$y = y.next$

$y.next = x.next$

else

$L.head = x.next$

Operacji wykonuje się w czasie $O(n)$, należy najpierw wyznaczyć element x za pomocą procedury LIST1-SEARCH

Lista z dowiązaniem - metody

Usuwanie elementu x z listy jednokierunkowej L :

procedure LIST1-DELETE(L, x)

if $x \neq L.head$

$y = L.head$

while $y.next \neq x$

$y = y.next$

$y.next = x.next$

else

$L.head = x.next$

Operacji wykonuje się w czasie $O(n)$, należy najpierw wyznaczyć element x za pomocą procedury LIST1-SEARCH

Lista z dowiązaniem - metody

Usuwanie elementu x z listy jednokierunkowej L :

procedure LIST1-DELETE(L, x)

if $x \neq L.head$

$y = L.head$

while $y.next \neq x$

$y = y.next$

$y.next = x.next$

else

$L.head = x.next$

Operacji wykonuje się w czasie $O(n)$, należy najpierw wyznaczyć element x za pomocą procedury LIST1-SEARCH

Lista z dowiązaniem

Element x listy dwukierunkowej L składa się z trzech pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)
- $x.prev$ - wskaźnik do poprzedniego elementu x na liście (jeśli x jest pierwszy na liście to $x.prev = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem

Element x listy dwukierunkowej L składa się z trzech pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)
- $x.prev$ - wskaźnik do poprzedniego elementu x na liście (jeśli x jest pierwszy na liście to $x.prev = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem

Element x listy dwukierunkowej L składa się z trzech pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)
- $x.prev$ - wskaźnik do poprzedniego elementu x na liście (jeśli x jest pierwszy na liście to $x.prev = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem

Element x listy dwukierunkowej L składa się z trzech pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)
- $x.prev$ - wskaźnik do poprzedniego elementu x na liście (jeśli x jest pierwszy na liście to $x.prev = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem

Element x listy dwukierunkowej L składa się z trzech pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)
- $x.prev$ - wskaźnik do poprzedniego elementu x na liście (jeśli x jest pierwszy na liście to $x.prev = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem

Element x listy dwukierunkowej L składa się z trzech pól:

- $x.key$ - klucz, czyli wartość przechowywana przez element x
- $x.next$ - wskaźnik do następnego elementu x na liście (jeśli x jest ostatni na liście to $x.next = NIL$)
- $x.prev$ - wskaźnik do poprzedniego elementu x na liście (jeśli x jest pierwszy na liście to $x.prev = NIL$)

Na pierwszy element listy L wskazuje jej atrybut $L.head$. Lista L jest pusta, jeśli $L.head = NIL$

Lista z dowiązaniem - metody

Wyszukiwanie w liście dwukierunkowej L elementu o kluczu k:

```
procedure LIST2-SEARCH(L,k)
  x = L.head
  while x  $\neq$  NIL and x.key  $\neq$  k
    x = x.next
  return x
```

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniem - metody

Wyszukiwanie w liście dwukierunkowej L elementu o kluczu k:

```
procedure LIST2-SEARCH(L,k)
  x = L.head
  while x  $\neq$  NIL and x.key  $\neq$  k
    x = x.next
  return x
```

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniem - metody

Wyszukiwanie w liście dwukierunkowej L elementu o kluczu k:

```
procedure LIST2-SEARCH(L,k)
  x = L.head
  while x  $\neq$  NIL and x.key  $\neq$  k
    x = x.next
  return x
```

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniem - metody

Wstawianie elementu x (którego pole *key* zostało wcześniej zainicjowane) na początek listy L :

```
procedure LIST2-INSERT( $L, x$ )
```

```
   $x.next = L.head$ 
```

```
  if  $L.head \neq NIL$ 
```

```
     $L.head.prev = x$ 
```

```
   $L.head = x$ 
```

```
   $x.prev = NIL$ 
```

Operacji wykonuje się w czasie $\Theta(1)$

Lista z dowiązaniem - metody

Wstawianie elementu x (którego pole *key* zostało wcześniej zainicjowane) na początek listy L :

procedure LIST2-INSERT(L, x)

$x.\text{next} = L.\text{head}$

if $L.\text{head} \neq \text{NIL}$

$L.\text{head}.\text{prev} = x$

$L.\text{head} = x$

$x.\text{prev} = \text{NIL}$

Operacji wykonuje się w czasie $\Theta(1)$

Lista z dowiązaniem - metody

Wstawianie elementu x (którego pole *key* zostało wcześniej zainicjowane) na początek listy L :

procedure LIST2-INSERT(L, x)

$x.next = L.head$

if $L.head \neq NIL$

$L.head.prev = x$

$L.head = x$

$x.prev = NIL$

Operacji wykonuje się w czasie $\Theta(1)$

Lista z dowiązaniem - metody

Usuwanie elementu x z listy dwukierunkowej L :

```
procedure LIST2-DELETE( $L, x$ )
```

```
  if  $x.\text{prev} \neq \text{NIL}$ 
```

```
     $x.\text{prev}.\text{next} = x.\text{next}$ 
```

```
  else
```

```
     $L.\text{head} = x.\text{next}$ 
```

```
  if  $x.\text{next} \neq \text{NIL}$ 
```

```
     $x.\text{next}.\text{prev} = x.\text{prev}$ 
```

Operacji wykonuje się w czasie $\Theta(1)$, należy najpierw wyznaczyć element x za pomocą procedury LIST2-SEARCH

Lista z dowiązaniem - metody

Usuwanie elementu x z listy dwukierunkowej L :

procedure LIST2-DELETE(L, x)

if $x.\text{prev} \neq \text{NIL}$

$x.\text{prev}.\text{next} = x.\text{next}$

else

$L.\text{head} = x.\text{next}$

if $x.\text{next} \neq \text{NIL}$

$x.\text{next}.\text{prev} = x.\text{prev}$

Operacji wykonuje się w czasie $\Theta(1)$, należy najpierw wyznaczyć element x za pomocą procedury LIST2-SEARCH

Lista z dowiązaniem - metody

Usuwanie elementu x z listy dwukierunkowej L :

procedure LIST2-DELETE(L, x)

if $x.\text{prev} \neq \text{NIL}$

$x.\text{prev}.\text{next} = x.\text{next}$

else

$L.\text{head} = x.\text{next}$

if $x.\text{next} \neq \text{NIL}$

$x.\text{next}.\text{prev} = x.\text{prev}$

Operacji wykonuje się w czasie $\Theta(1)$, należy najpierw wyznaczyć element x za pomocą procedury LIST2-SEARCH

Lista z dowiązaniem

Inne rodzaje list:

- lista posortowana
- lista cykliczna

Lista z dowiązaniem

Inne rodzaje list:

- lista posortowana
- lista cykliczna

Lista z dowiązaniem

Inne rodzaje list:

- lista posortowana
- lista cykliczna

Lista z dowiązaniem

Przykładowe zastosowanie list z dowiązaniem:

- listowa reprezentacja stosu i kolejki, bez ograniczenia na liczbę elementów
- reprezentacja grafu za pomocą listy sąsiedztwa
- implementacja złożonych struktur danych (np. kopce Fibonacciego)

Lista z dowiązaniem

Przykładowe zastosowanie list z dowiązaniem:

- listowa reprezentacja stosu i kolejki, bez ograniczenia na liczbę elementów
- reprezentacja grafu za pomocą listy sąsiedztwa
- implementacja złożonych struktur danych (np. kopce Fibonacciego)

Lista z dowiązaniem

Przykładowe zastosowanie list z dowiązaniem:

- listowa reprezentacja stosu i kolejki, bez ograniczenia na liczbę elementów
- reprezentacja grafu za pomocą listy sąsiedztwa
- implementacja złożonych struktur danych (np. kopce Fibonacciego)

Lista z dowiązaniem

Przykładowe zastosowanie list z dowiązaniem:

- listowa reprezentacja stosu i kolejki, bez ograniczenia na liczbę elementów
- reprezentacja grafu za pomocą listy sąsiedztwa
- implementacja złożonych struktur danych (np. kopce Fibonacciego)

Lista z dowiązaniem

Listowa implementacja stosu:

- S jest listą jednokierunkową
- elementy wstawiamy na początek listy i usuwamy z początku listy
- nie występuje błąd przepełnienia

Lista z dowiązaniem

Listowa implementacja stosu:

- S jest listą jednokierunkową
- elementy wstawiamy na początek listy i usuwamy z początku listy
- nie występuje błąd przepełnienia

Lista z dowiązaniem

Listowa implementacja stosu:

- S jest listą jednokierunkową
- elementy wstawiamy na początek listy i usuwamy z początku listy
- nie występuje błąd przepełnienia

Lista z dowiązaniem

Listowa implementacja stosu:

- S jest listą jednokierunkową
- elementy wstawiamy na początek listy i usuwamy z początku listy
- nie występuje błąd przepełnienia

Lista z dowiązaniem

Listowa reprezentacja kolejki:

- Q jest listą jednokierunkową
- aby wstawić i usunąć element w czasie $\Theta(1)$, trzeba wzbogacić listę o wskaźnik na ostatni element $Q.tail$
- element wstawiamy na koniec listy i usuwamy z początku listy
- nie występuje błąd przepelnienia

Lista z dowiązaniem

Listowa reprezentacja kolejki:

- Q jest listą jednokierunkową
- aby wstawić i usunąć element w czasie $\Theta(1)$, trzeba wzbogacić listę o wskaźnik na ostatni element `Q.tail`
- element wstawiamy na koniec listy i usuwamy z początku listy
- nie występuje błąd przepełnienia

Lista z dowiązaniem

Listowa reprezentacja kolejki:

- Q jest listą jednokierunkową
- aby wstawić i usunąć element w czasie $\Theta(1)$, trzeba wzbogacić listę o wskaźnik na ostatni element `Q.tail`
- element wstawiamy na koniec listy i usuwamy z początku listy
- nie występuje błąd przepełnienia

Lista z dowiązaniem

Listowa reprezentacja kolejki:

- Q jest listą jednokierunkową
- aby wstawić i usunąć element w czasie $\Theta(1)$, trzeba wzbogacić listę o wskaźnik na ostatni element `Q.tail`
- element wstawiamy na koniec listy i usuwamy z początku listy
- nie występuje błąd przepełnienia

Lista z dowiązaniem

Listowa reprezentacja kolejki:

- Q jest listą jednokierunkową
- aby wstawić i usunąć element w czasie $\Theta(1)$, trzeba wzbogacić listę o wskaźnik na ostatni element `Q.tail`
- element wstawiamy na koniec listy i usuwamy z początku listy
- nie występuje błąd przepełnienia