Algorytmy i struktury danych - Lista z dowiązaniami

Marcin Żurowski

07 maja 2026

Plan zajęć

Lista z dowiązaniami

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

Lista z dowiązaniami 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 dowiązaniami

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

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

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

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami - metody

Wyszukiwanie w liście jednokierunkowej L elementu o kluczu k:

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

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniami - metody

Wyszukiwanie w liście jednokierunkowej L elementu o kluczu k:

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

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniami - metody

Wyszukiwanie w liście jednokierunkowej L elementu o kluczu k:

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

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniami - 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 dowiązaniami - 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 dowiązaniami - 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 dowiązaniami

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami - metody

Usuwanie elementu x z listy jednokierunkowej L:

```
procedure LIST1-DELETE(L,x)
 if x ≠ L.head
  y = L.head
  while y.next ≠ 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 dowiązaniami - metody

Usuwanie elementu x z listy jednokierunkowej L:

```
procedure LIST1-DELETE(L,x)
 if x ≠ L.head
  y = L.head
  while y.next ≠ 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 dowiązaniami - metody

Usuwanie elementu x z listy jednokierunkowej L:

```
procedure LIST1-DELETE(L,x)
 if x ≠ L.head
  y = L.head
  while y.next ≠ 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 dowiązaniami

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami

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 dowiązaniami - metody

Wyszukiwanie w liście dwukierunkowej L elementu o kluczu k:

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

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniami - metody

Wyszukiwanie w liście dwukierunkowej L elementu o kluczu k:

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

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniami - metody

Wyszukiwanie w liście dwukierunkowej L elementu o kluczu k:

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

Operacji wykonuje się w czasie $O(n)$

Lista z dowiązaniami - 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 ≠ NIL
    L.head.prev = x
  L.head = x
  x.prev = NIL
```

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

Lista z dowiązaniami - 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 ≠ NIL
    L.head.prev = x
  L.head = x
  x.prev = NIL
```

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

Lista z dowiązaniami - 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 ≠ NIL
    L.head.prev = x
  L.head = x
  x.prev = NIL
```

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

Lista z dowiązaniami - metody

Usuwanie elementu x z listy dwukierunkowej L:

```
procedure LIST2-DELETE(L,x)
 if x.prev ≠ NIL
  x.prev.next = x.next
 else
  L.head = x.next
 if x.next ≠ NIL
  x.next.prev = x.prev
```

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

Lista z dowiązaniami - metody

Usuwanie elementu x z listy dwukierunkowej L:

```
procedure LIST2-DELETE(L,x)
 if x.prev ≠ NIL
  x.prev.next = x.next
 else
  L.head = x.next
 if x.next ≠ NIL
  x.next.prev = x.prev
```

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

Lista z dowiązaniami - metody

Usuwanie elementu x z listy dwukierunkowej L:

```
procedure LIST2-DELETE(L,x)
 if x.prev ≠ NIL
  x.prev.next = x.next
 else
  L.head = x.next
 if x.next ≠ NIL
  x.next.prev = x.prev
```

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

Lista z dowiązaniami

Inne rodzaje list:

- lista posortowana
- lista cykliczna

Lista z dowiązaniami

Inne rodzaje list:

- lista posortowana
- lista cykliczna

Lista z dowiązaniami

Inne rodzaje list:

- lista posortowana
- lista cykliczna

Lista z dowiązaniami

Przykładowe zastosowanie list z dowiązaniami:

- 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 dowiązaniami

Przykładowe zastosowanie list z dowiązaniami:

- 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 dowiązaniami

Przykładowe zastosowanie list z dowiązaniami:

- 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 dowiązaniami

Przykładowe zastosowanie list z dowiązaniami:

- 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 dowiązaniami

Listowa implementacja stosu:

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

Lista z dowiązaniami

Listowa implementacja stosu:

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

Lista z dowiązaniami

Listowa implementacja stosu:

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

Lista z dowiązaniami

Listowa implementacja stosu:

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

Lista z dowiązaniami

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 dowiązaniami

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 występuję błąd przepełnienia

Lista z dowiązaniami

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 występuję błąd przepełnienia

Lista z dowiązaniami

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 występuję błąd przepełnienia

Lista z dowiązaniami

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 występuję błąd przepełnienia