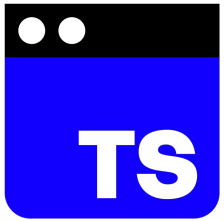
Code Academy

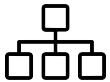


Duomenų struktūros
Sąrašas

TypeScript

7 paskaita

Paskaitos eiga



Sąrašo duomenų struktūra

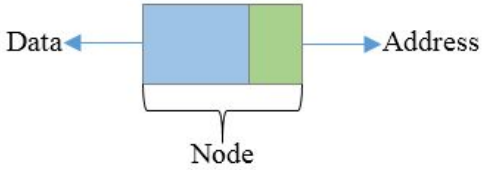
Sąrašo tipai

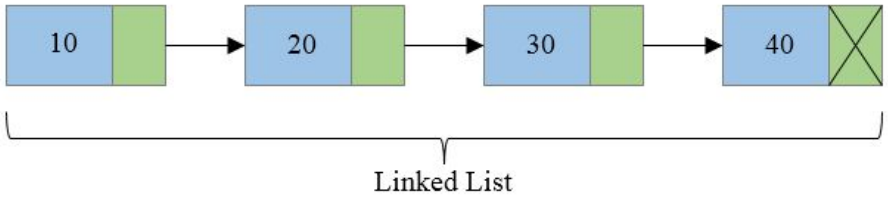
Vienpusio sąrašo manipuliacijos veiksmai

Sąrašo duomenų struktūra

Kas yra sąrašas?

Sąrašas, tai duomenų struktūra kaupti duomenims. Sąrašas sudaromas iš duomenų vienetų - mazgų. Kiekvienas mazgas veda į sekantį ir tokiu būdu sudaro *grandinėlę*.



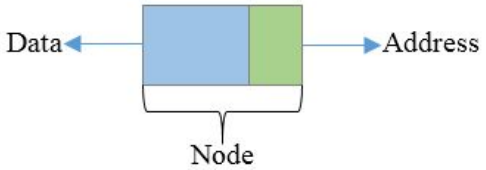


Primityvus mazgas

Primityvus mazgas, tai sąrašui skirta duomenų struktūra.

Primityvus mazgas susideda iš 2 savybių:

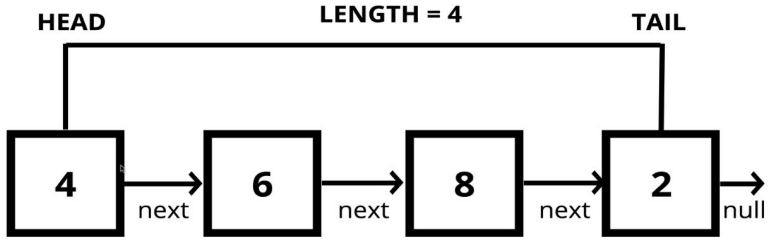
- Data - savybė saugoti duomenims
- Address - savybė saugoti nuorodą į sekantį elementą



Sąrašo savybės

Pats sąrašas turi papildomas savybes, reikalingas gauti informacijai ir atlikti sąrašo manipuliacijos veiksmus.

- length - elementų kiekis
- head - nuoroda į sąrašo pradžią
- tail - nuorodą į sąrašo pabaigą



Sąrašo tipai

Sąrašo tipai

Yra keli sąrašų tipai. Dažnai atliekant pakartotinius veiksmus su duomenimis gali trūkti kompiuterio resursų. Todėl svarbu pasirinkti reikiamą sąrašo tipą, kuris labiausiai tiks algoritmo veiksmams atlikti.

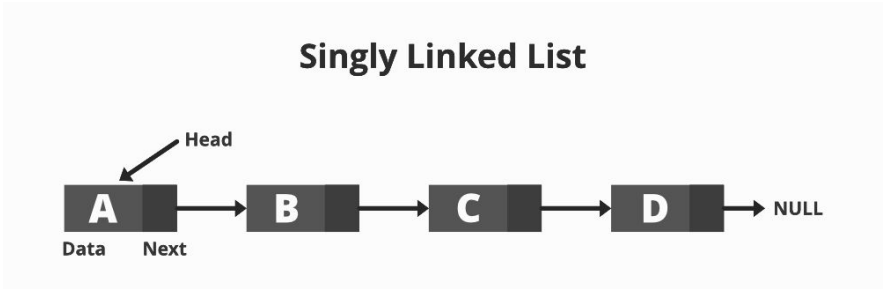
Sąrašo tipai

- **Vienakryptis sąrašas** - naudojamas dažnam duomenų panaudojimui. Pavyzdžiui postų atvaizdavimui.
- **Dvikryptis sąrašas** - naudojamas dažnam duomenų panaudojimui, kur reikia pakeisti rikiavimo kryptį. Arba gauti prieš tai buvusį elementą. Pavyzdžiui puslapio naršymo navigacija, ar lentelės duomenys.
- **Begalinis vienkryptis sąrašas** - naudojamas *swiperiam.*
- **Begalinis dvikryptis sąrašas** - naudojamas *swiperiam* kurie gali būti slenkami į dvi puses.

Vienakryptis sąrašas

Tai paprasčiausia ir pagrindinė sąrašo versija. Skirta dažnam duomenų atvaizdavimui iš eilės.

Taip paprasčiausi duomenų pridėjimo ir trynimo veiksmai.

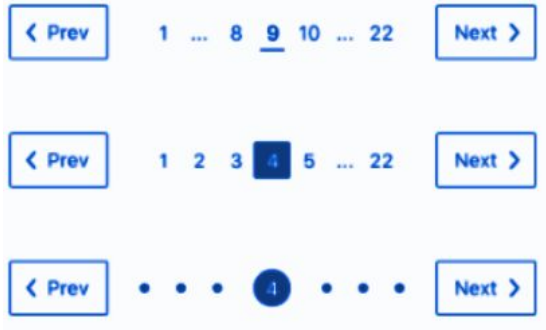


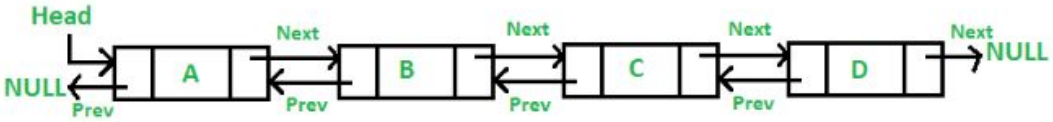
Dvikryptis sąrašas

Dvikryptis sąrašas nuo vienakrypčio skiriasi tuo, kad jo mazgai turi nuorodas ne tik į sekantį elementą, bet ir į prieš tai buvusį.

Iš vienos pusės tai privalumas, nes galima iteruoti į abi puses.

Iš kitos pusės minusas, nes sąrašo redagavimas tampa sudėtingesnis. Trinant ar pridedant elementus reikia rūpintis nuorodų “next” ir “prev” tvarkymu.

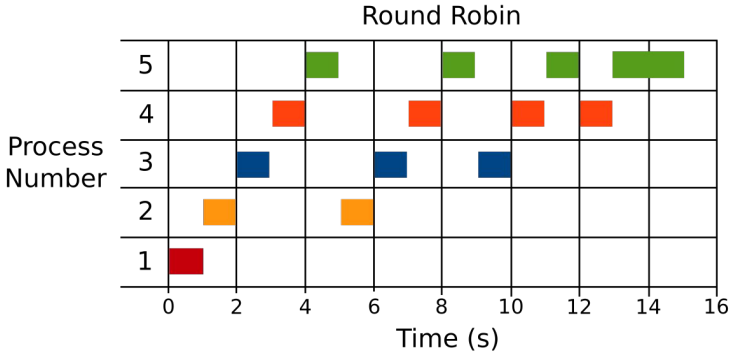


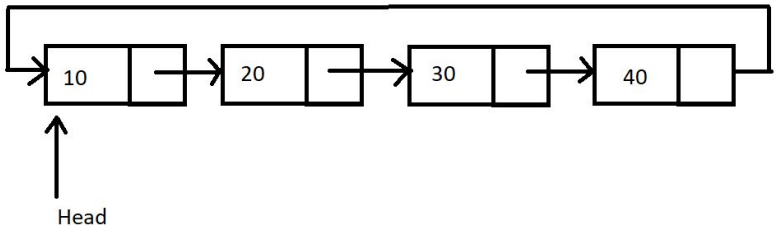


Begalinis vienkryptis sąrašas

Tai tokia sąrašo atmaina, kuomet paskutinis elementas rodo į pirmąjį. Tai naudinga kuomet iteravimas turi būti begalinis.

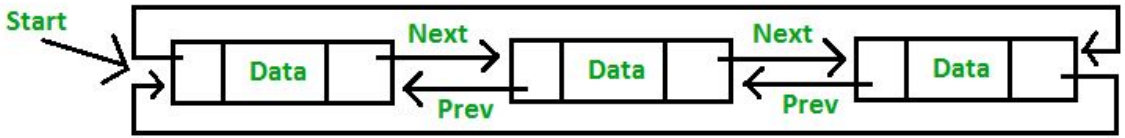
Pavyzdžiui: Algoritmams, kurie paskirsto resursus atliekant begalinę iteraciją (Round-Robin resource sharing).





Begalinis dvikryptis sąrašas

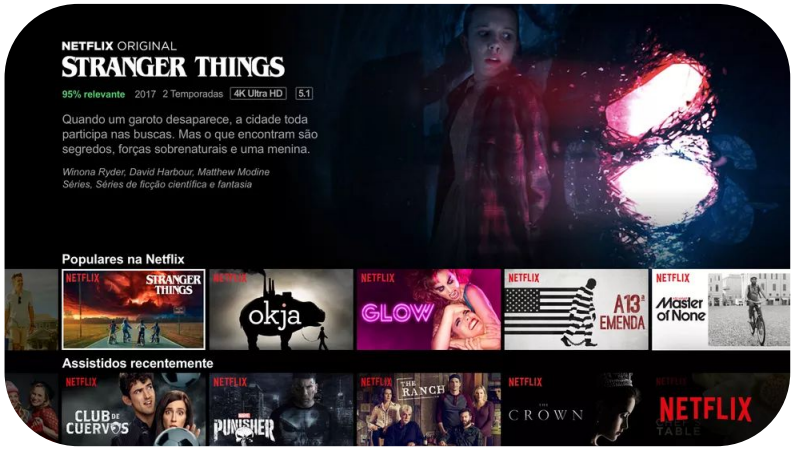
Begalinis dvikryptis sąrašas tai tokią sąrašo atmaina, kur pirmas elementas rodo į paskutinį, o paskutinis į pirmąjį.



Begalinis dvikryptis sąrašas

Šis sąrašas naudojamas, kuomet reikia grįžti atgal iteruojamu ratu:

- Muzikos grojaraštis
- Nuotraukų galerijos peržiūra
- Filmų meniu ir t.t.



Vienpusio sąrašo manipuliacijos veiksmai

Vienpusio sąrašo manipuliacijos veiksmai

Panagrinėsime vienpusio sąrašo manipuliacijos veiksmus:

- Iteravimas
- Pridėjimas
- Reiškmės suradimas sąraše/patikrinimas
- Trynimas

Įsisavinus šiuos metodus vienpusiui sąrašui, nesunkiai suprasite ir sudėtingesnių sąrašų metodų papildymus.

forEach - Iteravimas

Šį metodą jau pažįstate iš kitos duomenų struktūros - masyvo. Jis skirtas tam, kad programuotojai galėtų atlikti veiksmus su kiekvienu duomenų struktūros elementu.

Šiuo atveju - su kiekvienu sąrašo elementu.

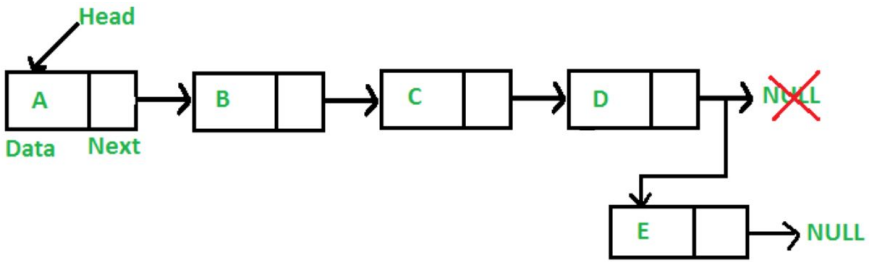
```
class List<T> {
  head: ListNode<T> | null;

  forEach(callback: (data: T) => void): void {
    let current = this.head;

    while (true) {
      if (current === null) break;
      callback(current.data);
      current = current.next;
    }
  }
}
```

```
class ListNode<T> {
  constructor(
    public val: T,
    public next: ListNode<T> | null = null
  ) { }
}
```

push - pridėjimas į sąrašo galą

Pridedant reikia įvertinti ar sąrašas yra tuščias. Jeigu sąrašas tuščias, pridėtas elementas taps pirmuoju ir paskutiniu vienu metu.

Jeigu sąrašas nėra tuščias, po pridėjimo į galą turime pakeisti paskutinio mazgo nuorodą *tail*, kad ji rodytų į naujai pridėta elementą.




```
class List<T> {
  head: ListNode<T> | null;
  tail: ListNode<T> | null;

 push(data: T): void {
    if (this.tail === null) {
      this.head = new ListNode(data);
      this.tail = this.head;
    } else {
      this.tail.next = new
ListNode(data);
      this.tail = this.tail.next;
    }
  }
}
```

include - patikrinimas, ar yra reikšmė sąraše

Iteravimas atliekamas nuo pirmo sąrašo elemento head.

Jeigu iteravimo metu randamas mazgas, kurio reikšmė lygi tikrinamai reikšmei data, grąžinama true ir automatiškai nutraukiamas tiek ciklas, tiek visa funkcija.

Jeigu praiteravus visus sąrašo elementus neradome mazgo, kurio reikšmė būtų lygi ieškomai reikšmei data, grąžiname false.

```
class List<T> {
  head: ListNode<T> | null;
  tail: ListNode<T> | null;

  includes(data: T): boolean {
    let current = this.head;

    while (true) {
      if (current === null) break;
      if (current.data === data) return true;
      current = current.next;
    }

    return false;
  }
}
```

pop - pašalinimas iš galo

Bandant pašalinti paskutinį elementą gali būt 3 atvejai:

1. Sąrašas tuščias - turi būt grąžinta *null*
2. Sąraše yra vienintelis mazgas - grąžiname mazgo duomenis, ir nuostatome *tail* ir *head* nuorodas į *null*
3. Sąraše yra daug elementų - grąžiname paskutinio elemento duomenis ir prieš paskutinį elementą nustatome paskutiniuoju.

```
pop(): T | null {
   let current = this.head;

   while (true) {
     if (current === null) return null;
     if (current.next === null) {
       const returnVal = current.data;
       this.head = null;
       this.tail = null;

       return returnVal;
     } else if (current.next.next === null) {
       const returnVal = current.next.data;
       current.next = null;
       this.tail = current;

       return returnVal;
     }
     current = current.next;
   }
 }
```

Klausimai?

Paskaitos darbas

Paskaitos darbas

Paskaitoje atliksime užduotis, tokia eiga:

1. Sprendžiame užduotis savarankiškai
2. Po savarankiško sprendimo laiko (10-30 min.) dėstytojas išsprendžia 1 užduotį argumentuodamas sprendimą
3. Studentai užduoda klausimus apie sprendimą
4. Sprendimų palyginimas
5. Atliekama sekanti užduotis

Jeigu išsprendėte užduotį anksčiau nei kiti, spręskite sekančias užduotis.

Užduoties aptarimo metu, nesidrovėkite klausti kuo daugiau klausimų. Nebūtinai jūsų sprendimas yra prastesnis. Galbūt net geresnis?

Iki kito karto!