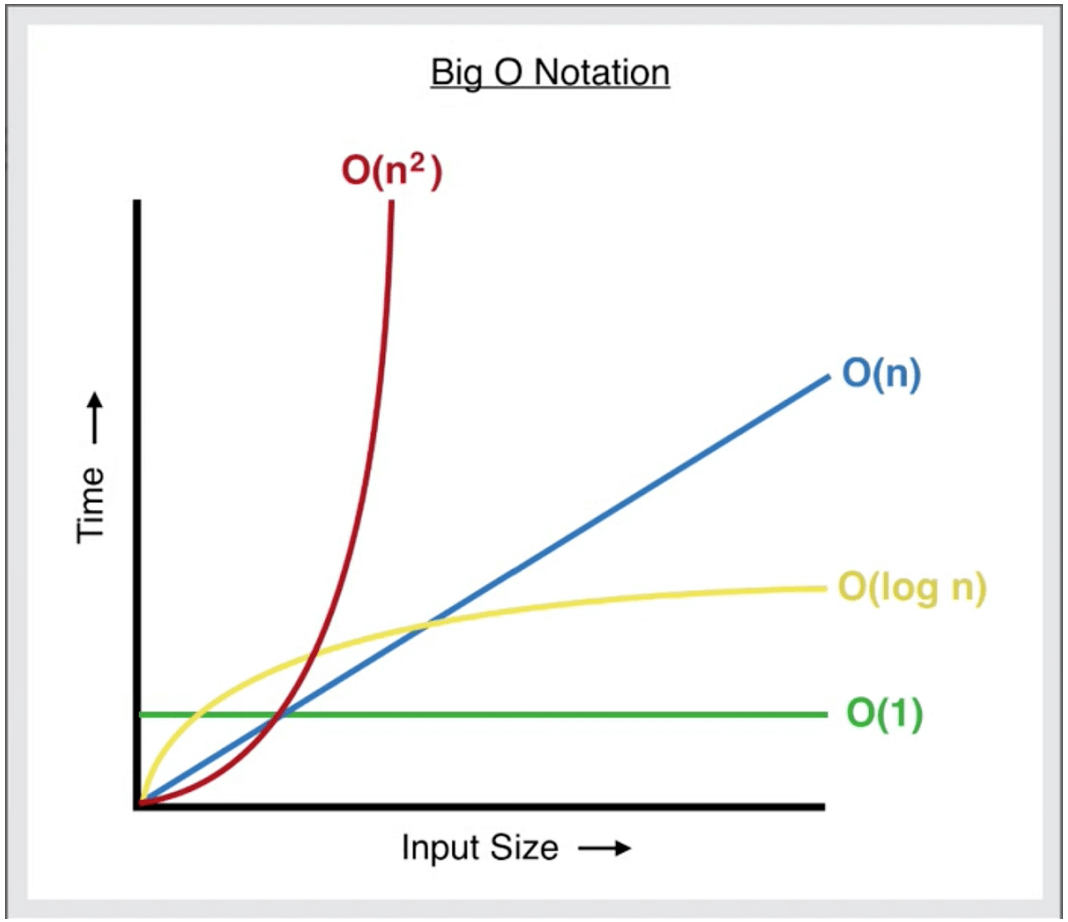
cphbusiness

Datastrukturer

Kompleksitet og performance

O-notationen bruges til at angive kompleksitet



Hvad betyder det?

`O(n)` – Med en samling af `n` elementer skal vi foretage `n` operationer.
Fx: hvis vi skal søge efter et element i et array med 10 pladser, skal vi maksimalt søge 10 elementer igennem.

`O(log n)` – I stedet for at kigge på alle `n` elementer, kan vi halvere problemet for hvert trin.

Fx: hvis vi skal søge efter et element i et træ med 16 elementer, skal vi søge gennem fire niveauer.

`O(1)` – ligegyldigt hvor stor vores samling er, skal vi altid bruge det samme antal operationer. Vi kalder det også konstant tid.

Fx: når vi slår op på index i et array, tager det altid samme tid uanset hvor stort arrayet er.

ArrayList og LinkedList

	ArrayList	LinkedList
Indsæt i slutning af liste (`add(element)`):	O(1)	O(1)
Indsæt et vilkårligt sted (`add(index, element)`)	O(n)	O(n)
Indsæt i starten af liste (`add(0, element)`)	O(n)	O(1)
Søgning (`contains(element)`)	O(n)	O(n)
Opslag på index (`get(index)`)	O(1)	O(n)
Fjerne et vilkårligt sted (`remove(index)`)	O(n)	O(n)
Fjerne første objekt (`remove(0)`)	O(n)	O(1)
Fjerne sidste objekt (`remove(list.size()-1)`)	O(1)	O(1)

Test det af

- Kig på metoden `searchTimes()` og se hvordan man kan måle tid i java
- Lav nogle eksperimenter med henholdsvis `ArrayList` og `LinkedList` som du fylder med objekter
- Undersøg hvordan de to lister performer når du indsætter, sletter eller søger. Prøv med forskellige antal elementer i listerne
- Vær systematisk og skriv resultaterne op i en tabel fx i Excel

Tree og Hash

	TreeSet/TreeMap	HashSet/HashMap
Indsæt (`add(element)`)	O(logn)	O(1)
Søge (`contains(element)`)	O(logn)	O(1)
Fjerne (`remove(element)`)	O(logn)	O(1)

Test det af

- Lav nogle eksperimenter med henholdsvis `TreeSet` og `HashSet` som du fylder med objekter
- Undersøg hvordan de to set performer når du indsætter, sletter eller søger. Prøv med forskellige antal elementer i settene
- Vær systematisk og skriv resultaterne op i en tabel
- Sammenlign med resultaterne for lister

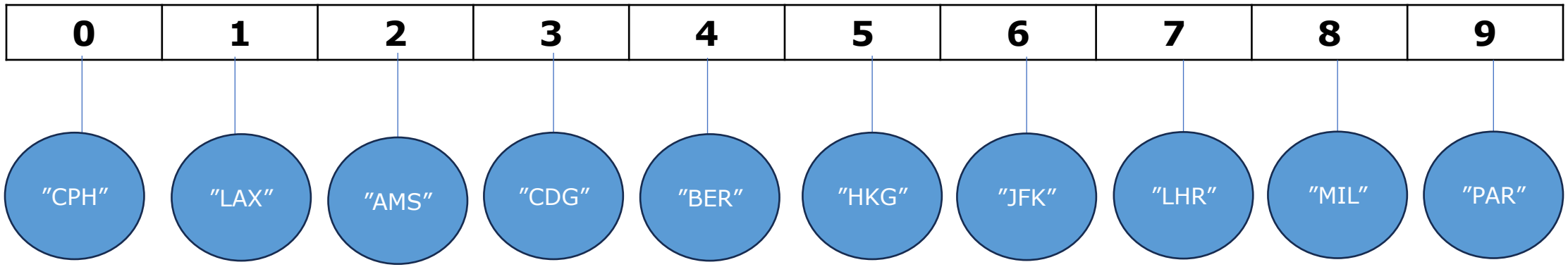
Opsamling

- Hvilken datastruktur skal vi bruge?
  - Kig først på egenskaber og vælg interface
  - Kig dernæst på hvad du skal bruge den til og vælge konkret klasse

ArrayList

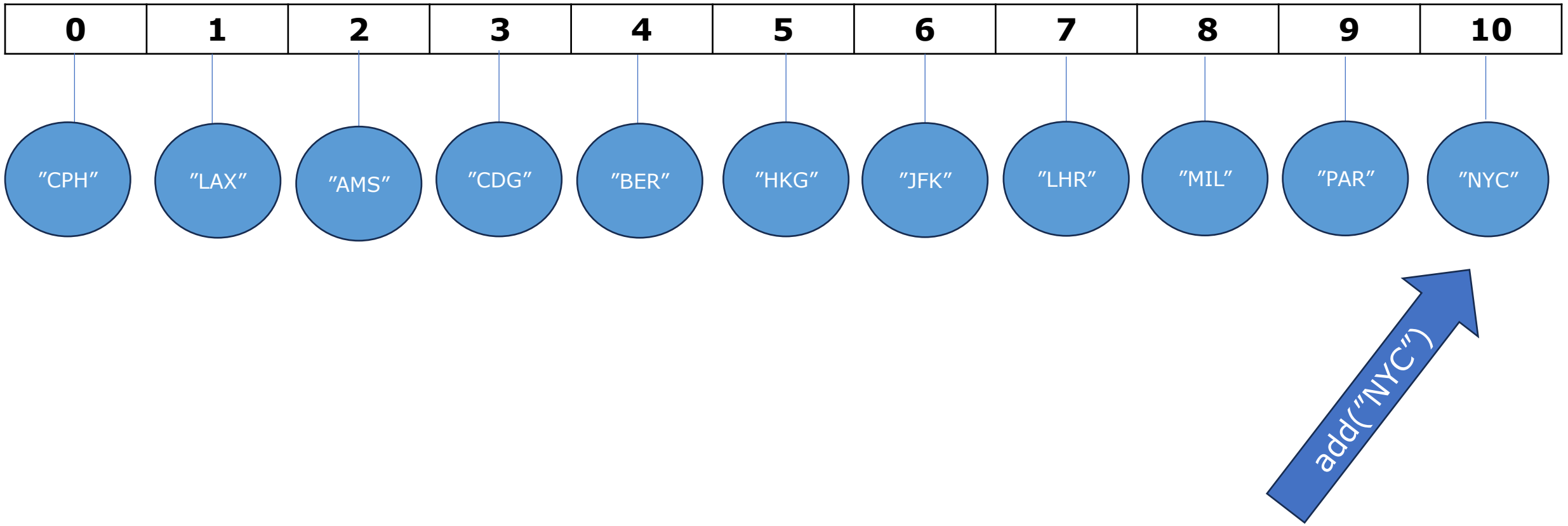
Hvad er kompleksiteten for

- Indsæt
  - `add(”NYC”)`
  - `add(0, ”NYC”)`
- Slet
  - `remove (0)`
  - `remove (”BER”)`
- Søg
  - `get(8)`
  - `indexOf(”MIL”)`



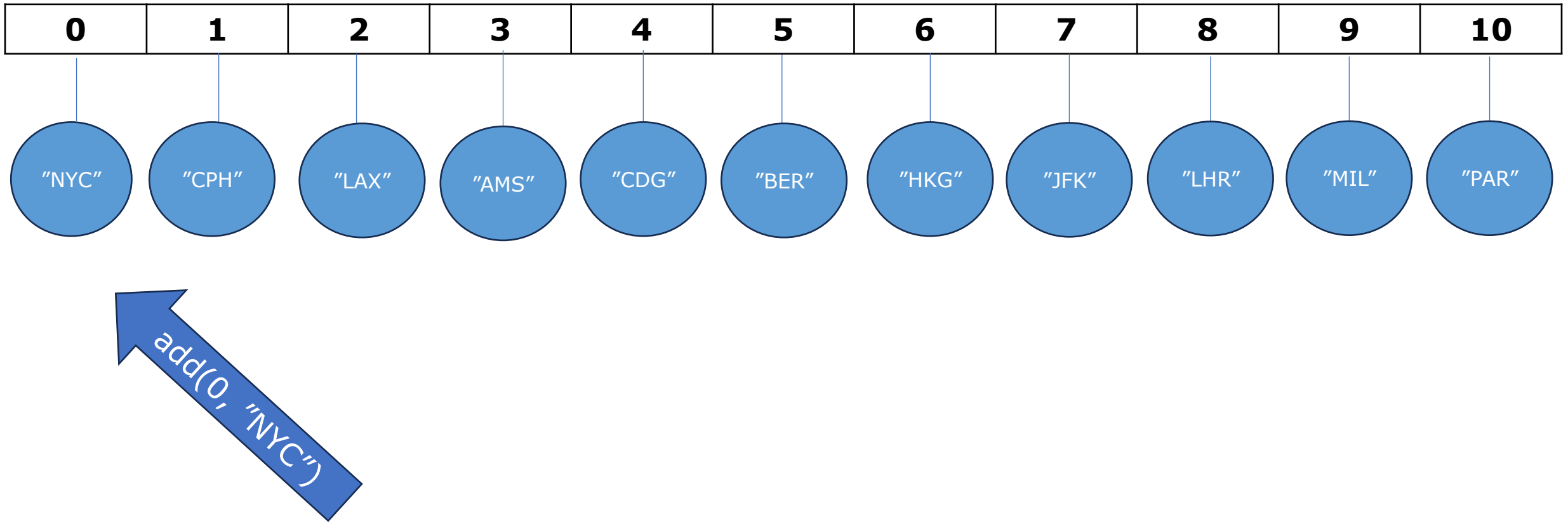
cphbusiness

ArrayList – add(element)



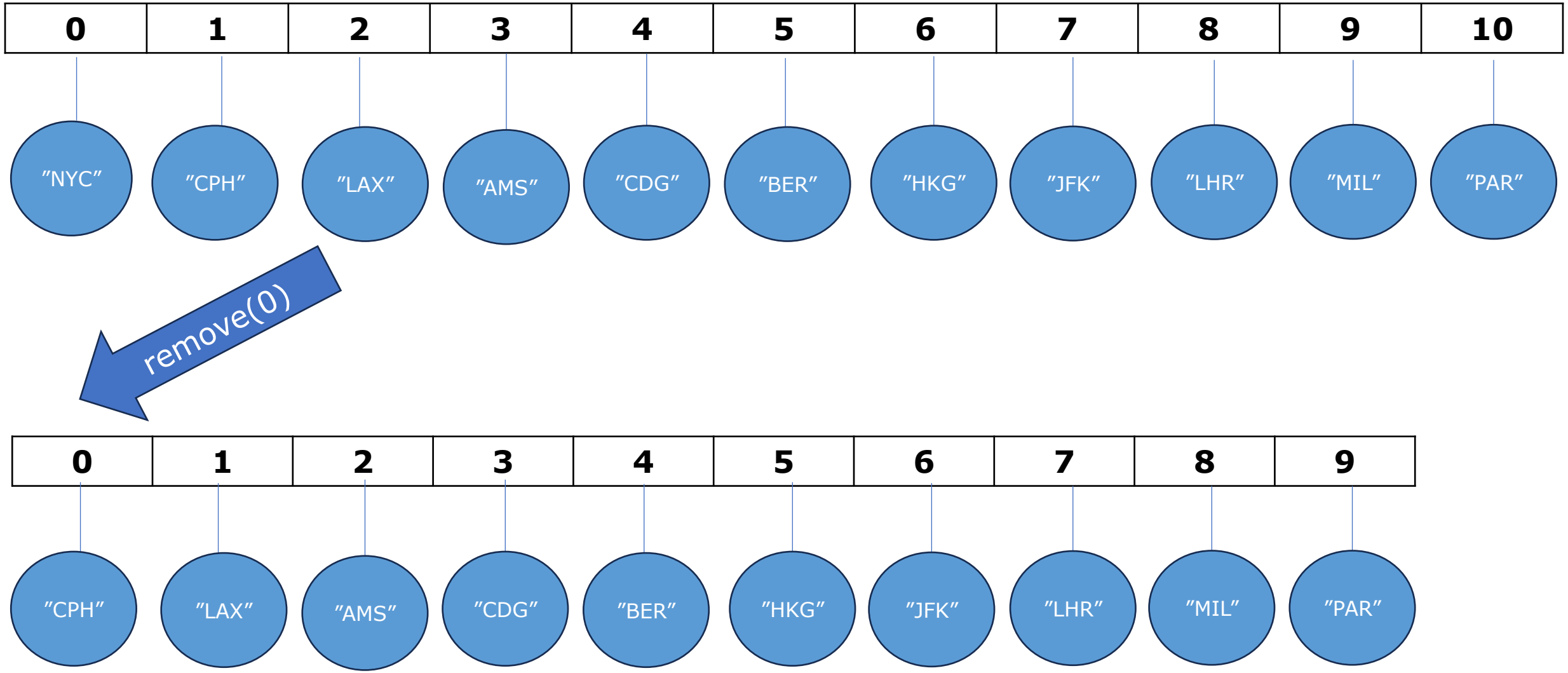
cphbusiness

ArrayList – add(index, element)

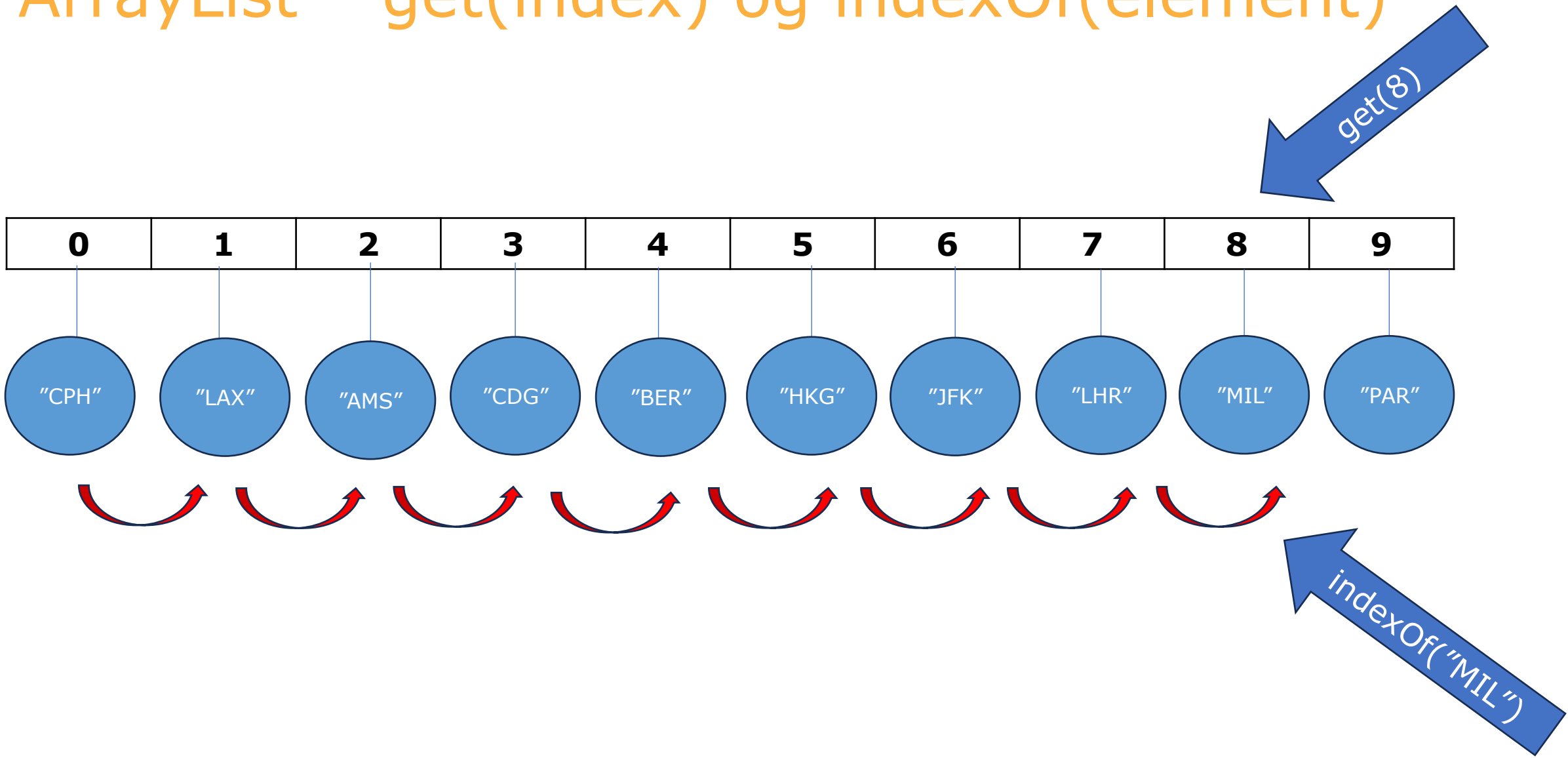


cphbusiness

ArrayList – remove(index)



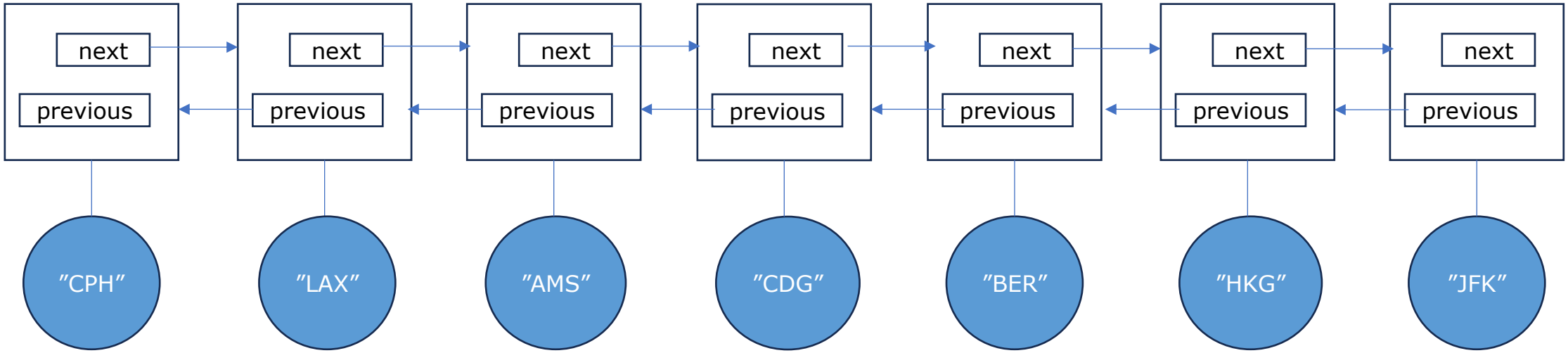
ArrayList – get(index) og indexOf(element)



LinkedList

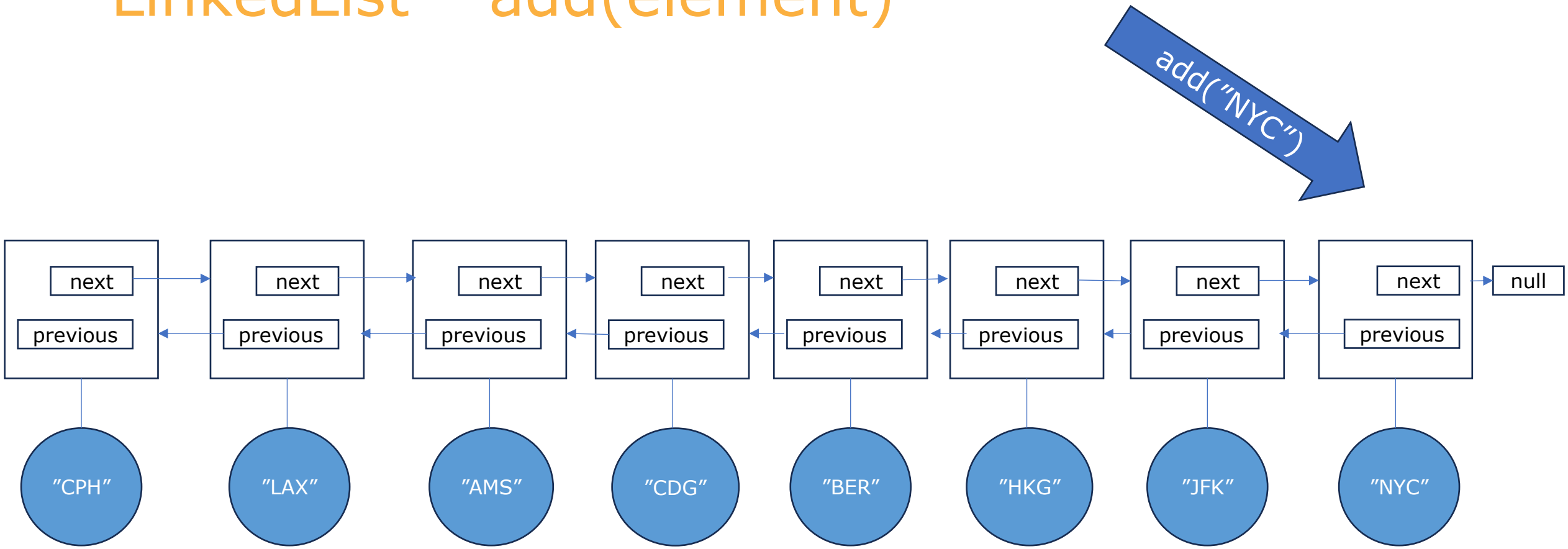
Hvad er kompleksitet for

- Indsæt
  - `add("NYC")`
  - `add(0, "NYC")`
- Slet
  - `remove (0)`
  - `remove ("BER")`
- Søg
  - `get(6)`



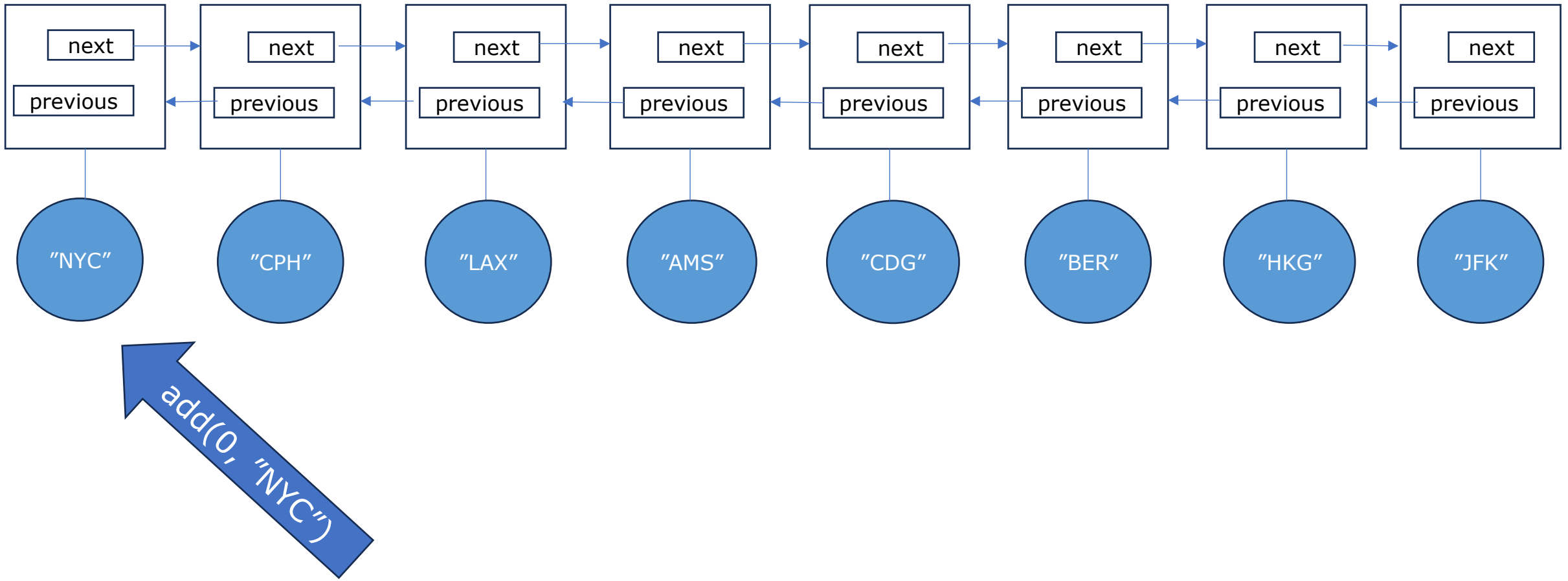
cphbusiness

LinkedList – add(element)



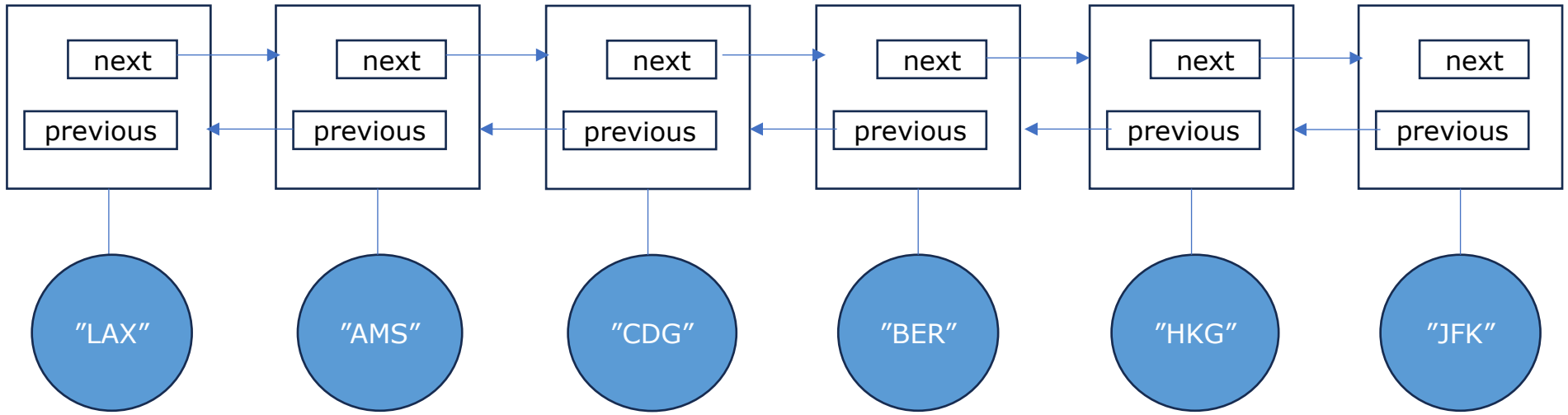
cphbusiness

LinkedList – add(index, element)



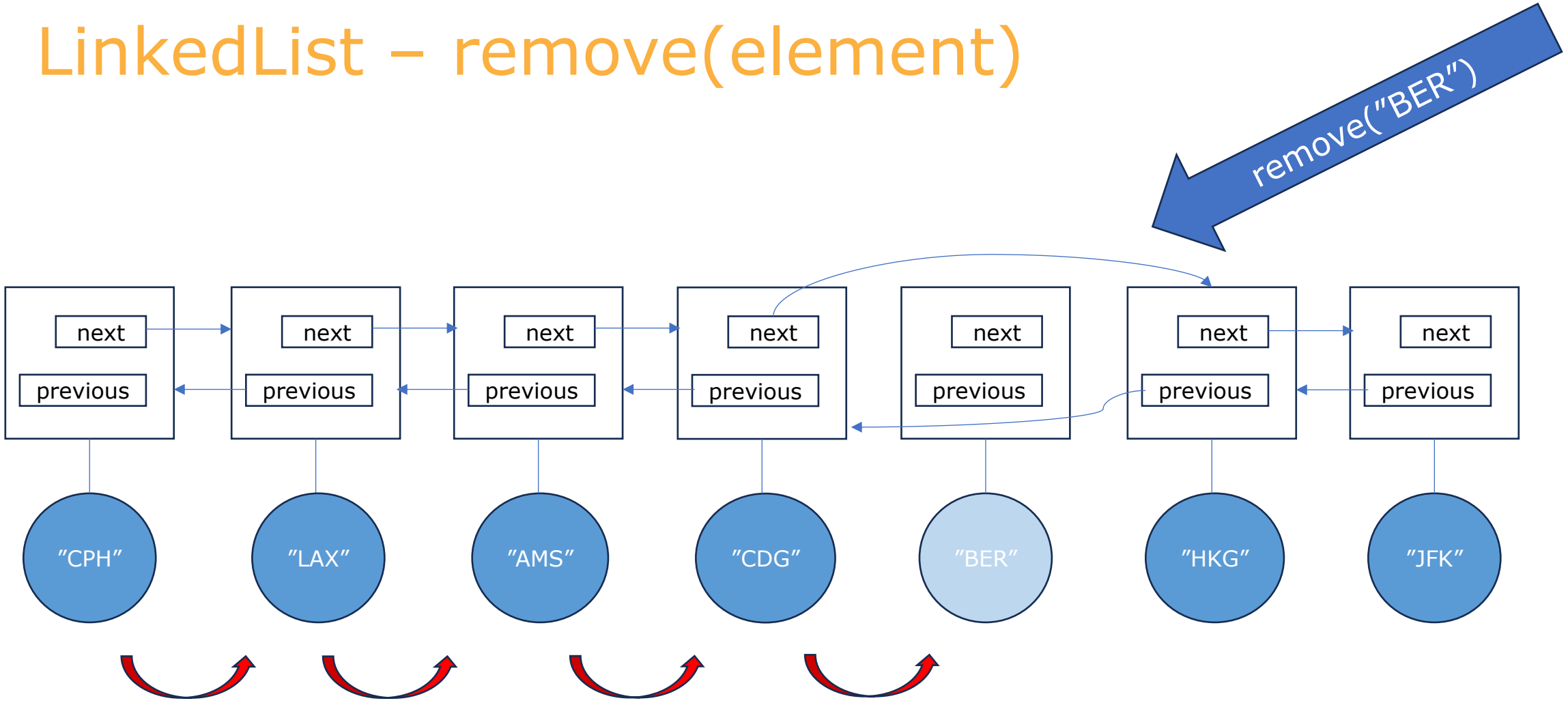
cphbusiness

LinkedList – remove(0)



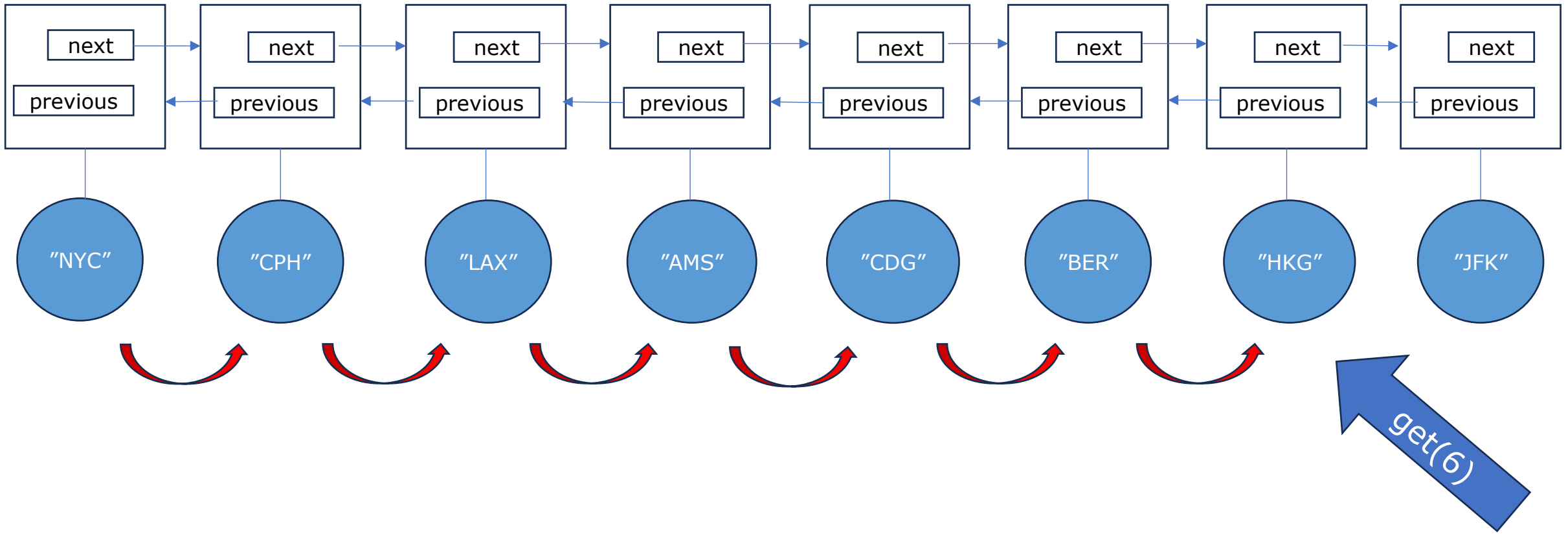
cphbusiness

LinkedList – remove(element)



cphbusiness

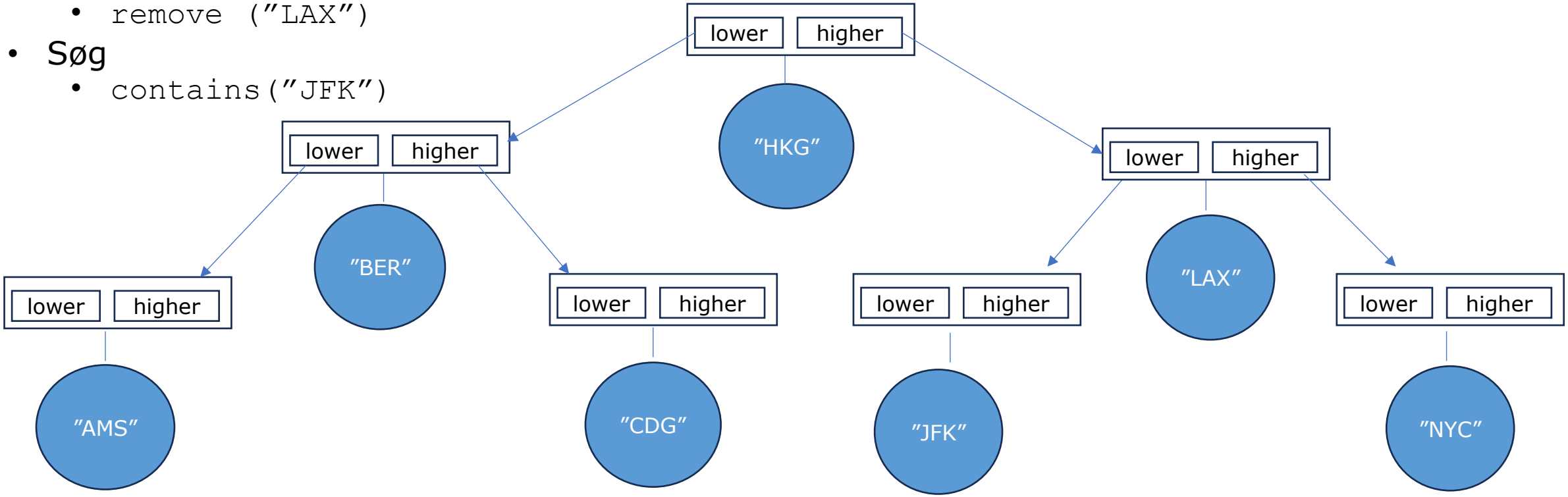
LinkedList – get(index)



TreeSet

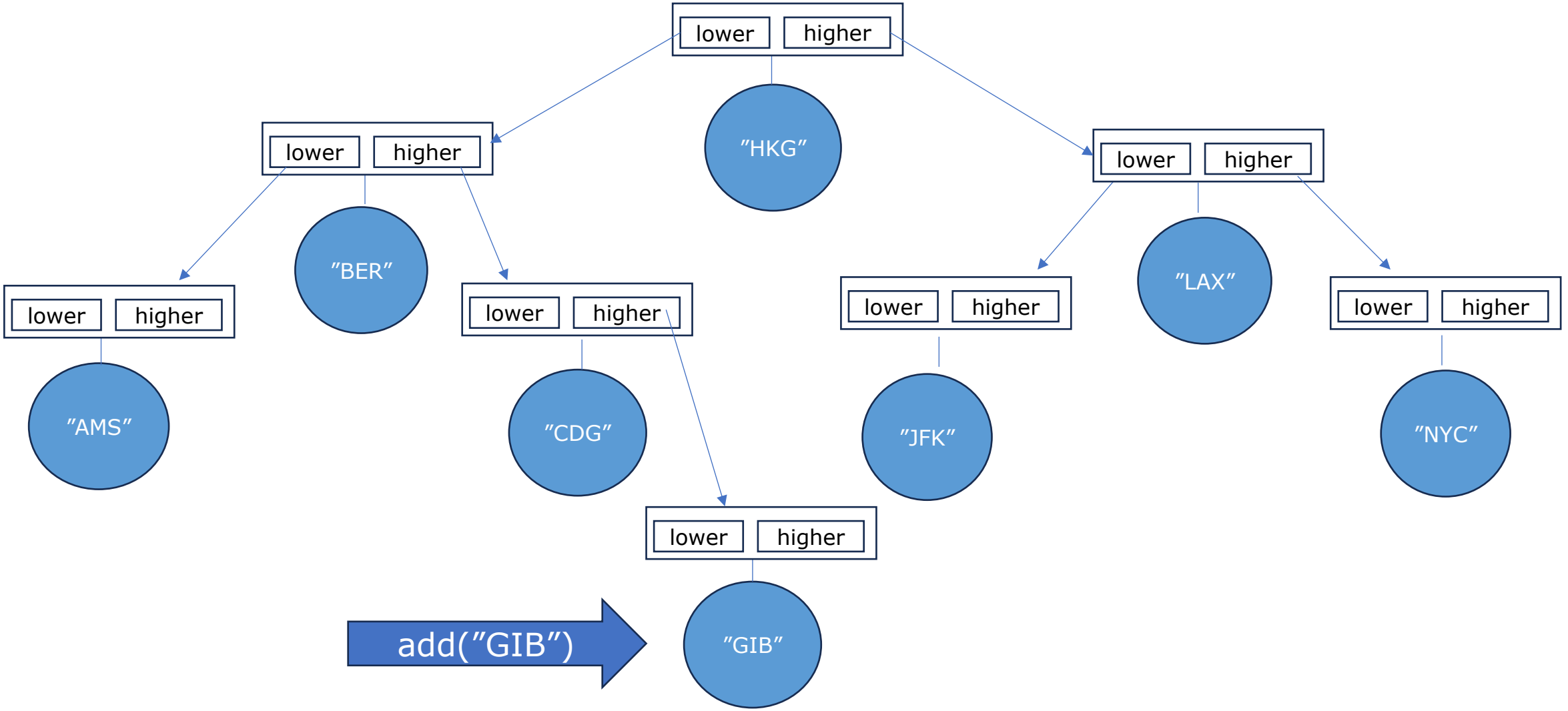
Hvad er kompleksiteten for

- Indsæt
  - `add("GIB")`
- Slet
  - `remove ("LAX")`
- Søg
  - `contains("JFK")`

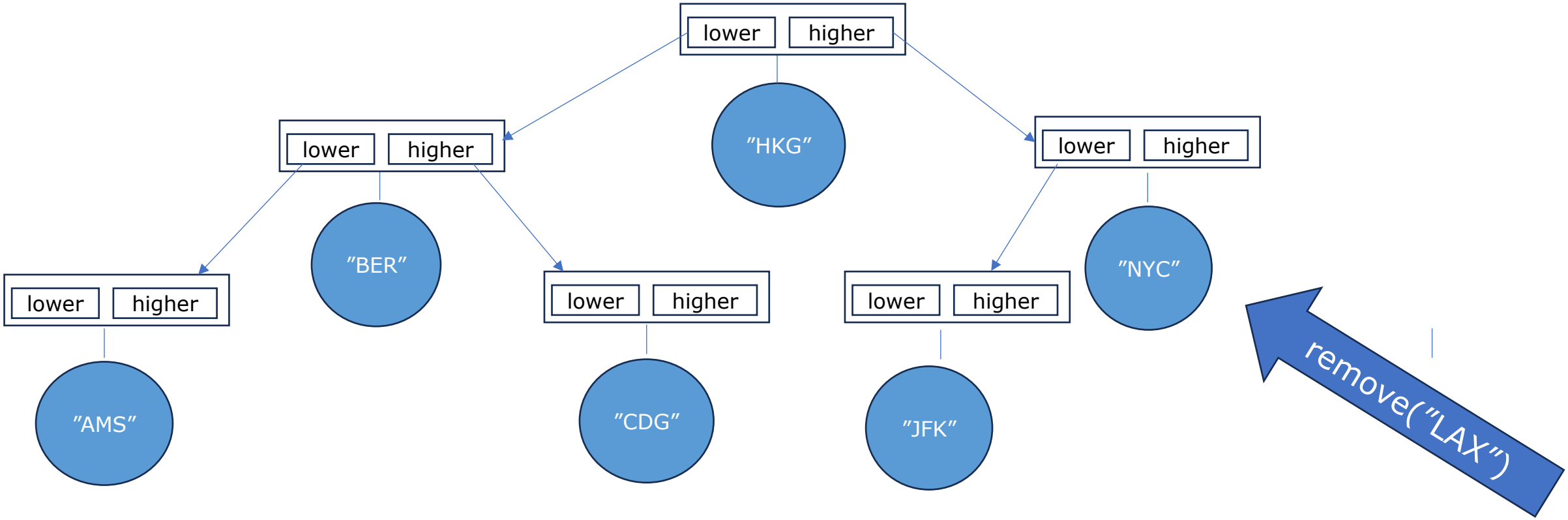


cphbusiness

TreeSet – add(element)

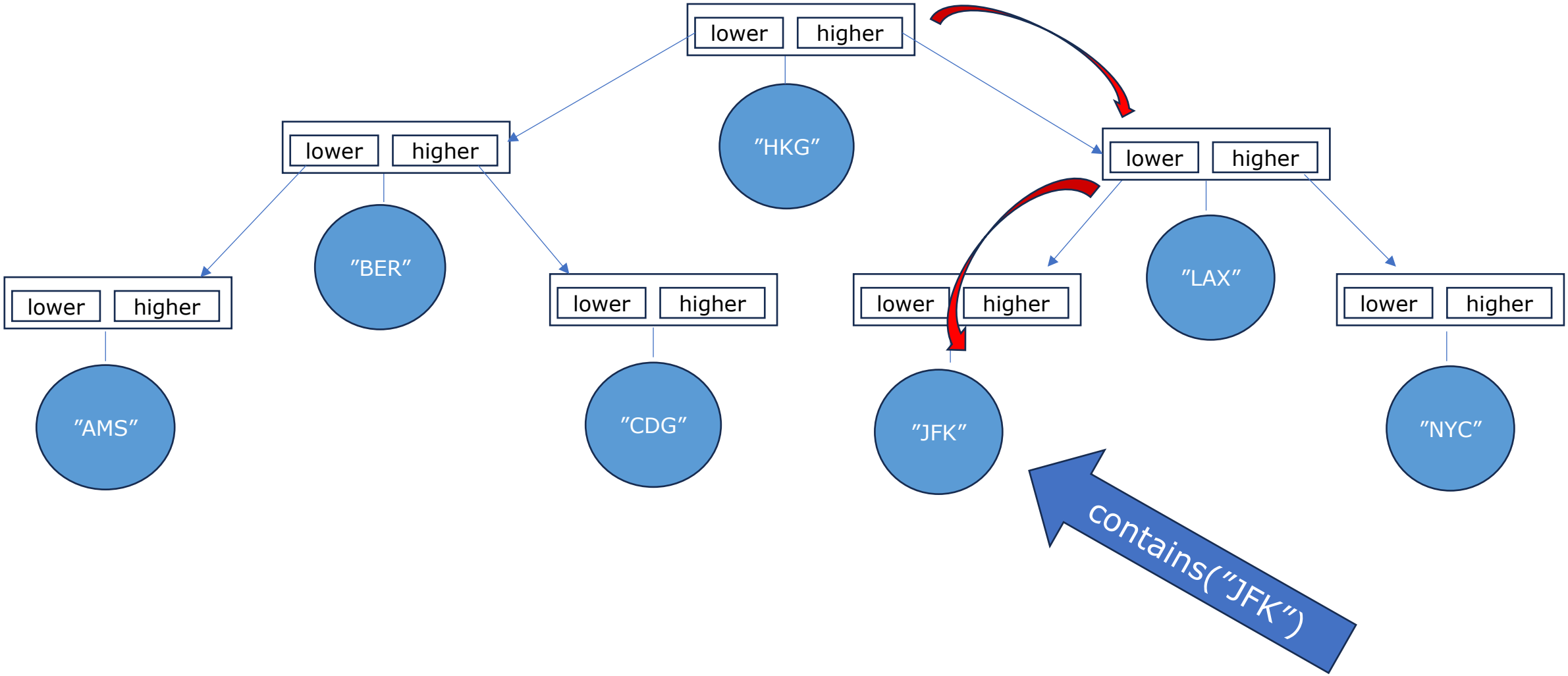


TreeSet – remove(element)



cphbusiness

TreeSet – get(element)

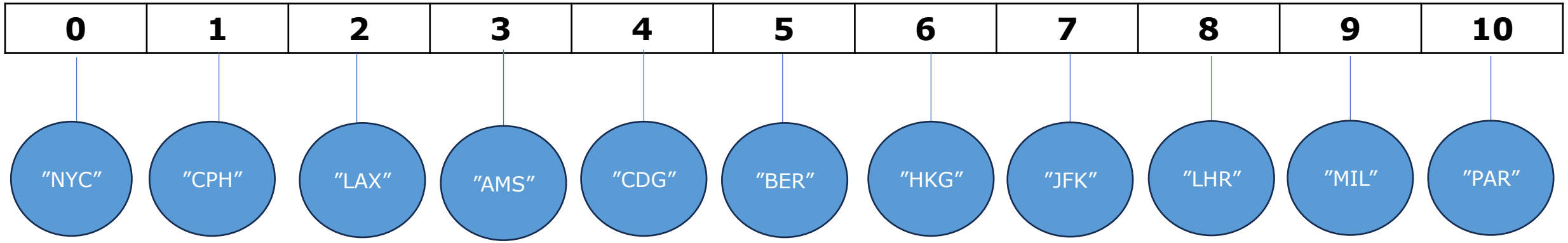


cphbusiness

HashSet

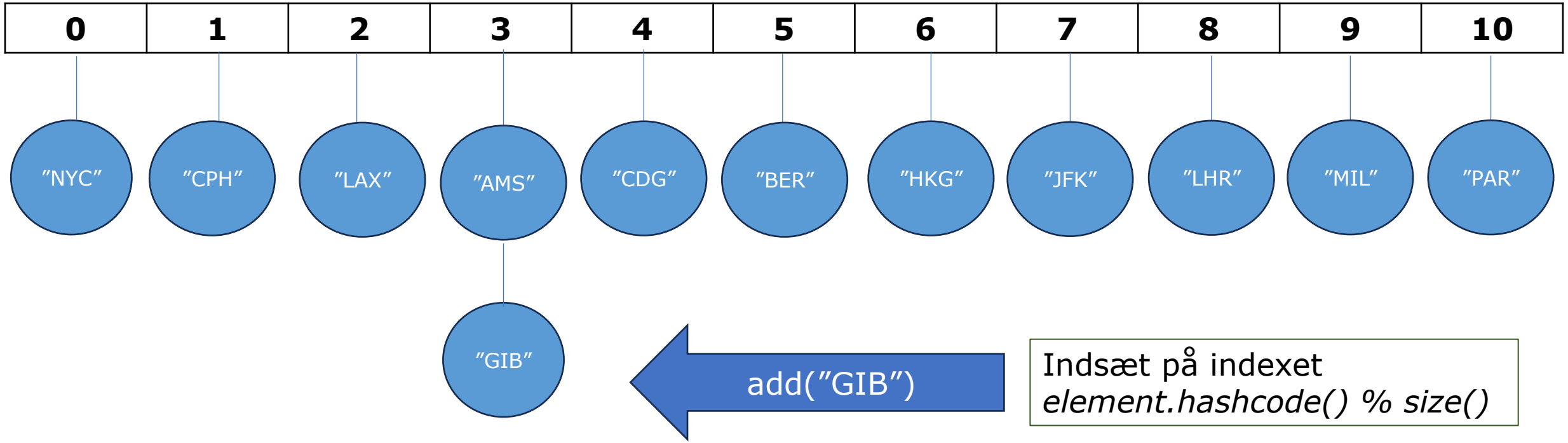
Hvad er kompleksiteten for

- Indsæt
  - `add("GIB")`
- Slet
  - `remove ("LAX")`
- Søg
  - `contains("JFK")`

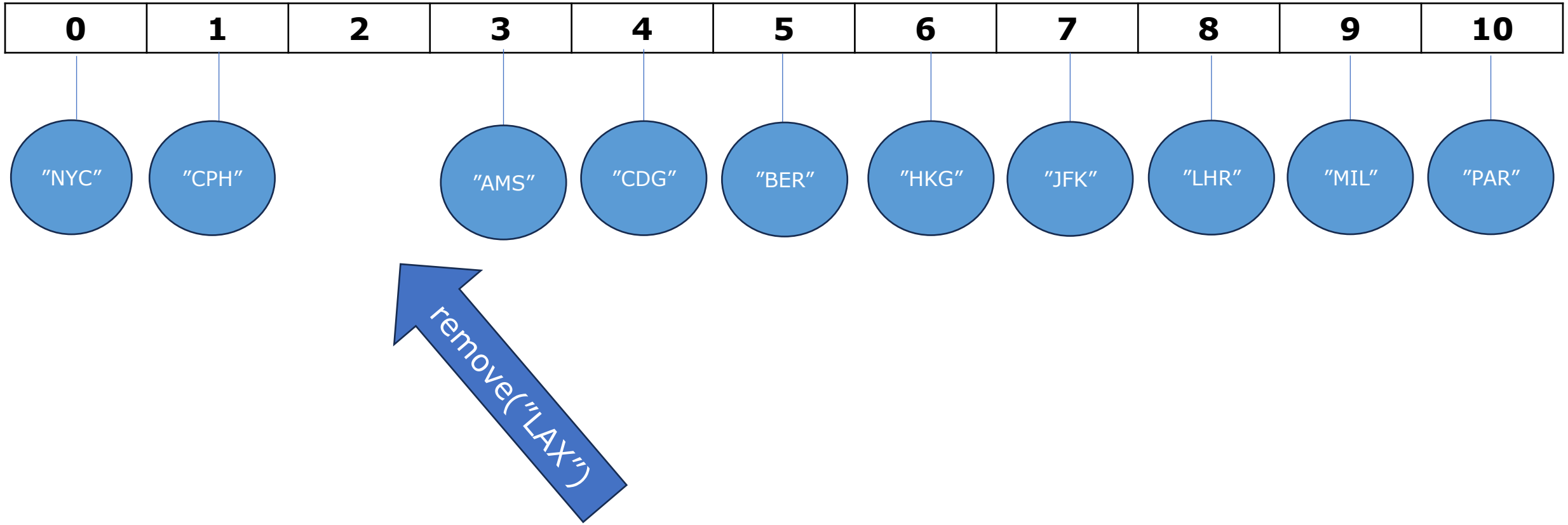


cphbusiness

HashSet – add(element)



HashSet – remove(element)



HashSet – contains(element)

Slå op på indexet
element.hashcode() % size()