STRUTTURA DATI → INTERFACCIA

<T> GENERICO = OGNI TIPO

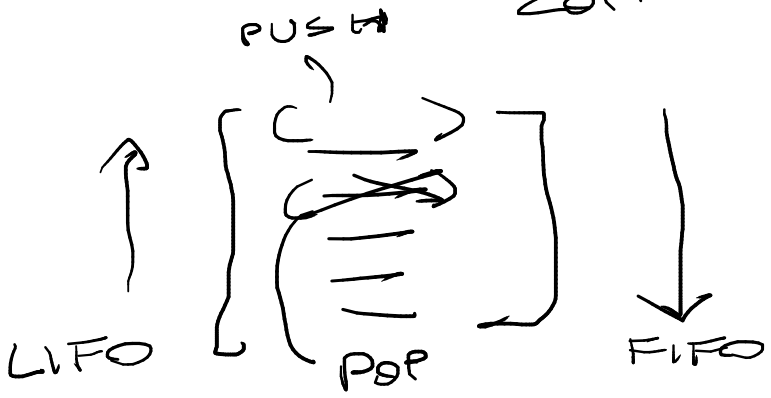
<S> SUCCESSIVO

CONTENITORE → INTERFACE

- AGGIUNTA = ADD()
- RIMOZIONE = REMOVE()
- QUANTI ELEMENTI HA = SIZE()
- SE È VUOTO = ISEMPTY()

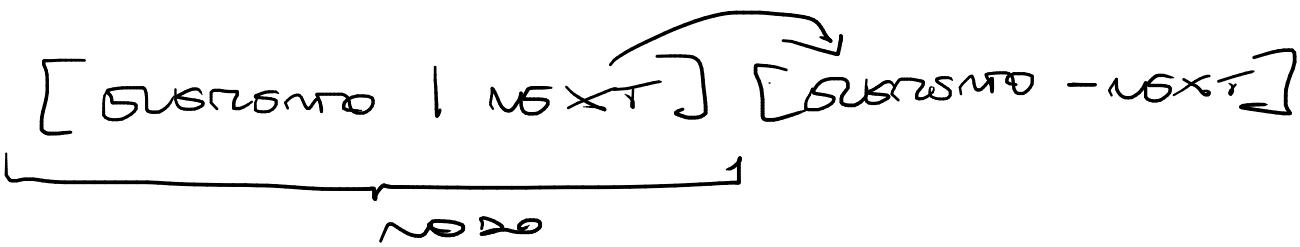
→ ARRAYLIST (INSIEME DI ELEMENTI A CODICE)

→ PILA (STACK)



PILA <T> (GENERICO)
- PILA <INTEGER>
- PILA <DOUBLE>
} ALTRO CHE VUOI ...

→ LISTA (LINKED LIST) → LISTA<T>



- CODA = QUEUE → PRIORITÀ (PRIORITY QUEUE)
  → DOPPIA (DEQUE) → COPPIA DI ELEMENTI

CODA

[- - - - - - - - - -]

- INSIEME (SET) → TREESET (NON EFFICIENTE COPPIA DI ELEMENTI)
  → HASHSET → ≠
  → ACCESSO EFFICIENTE AD ELEMENTI SPARSI
  MIGLIORE/PIÙ EFFICIENTE

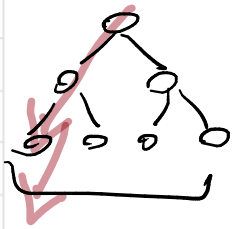
- MAPPE (MAP)

[KEY / VALUE]

HASHMAP = (HASHCODE) → COMPARAZIONE ELEMENTI

VAR X = 1;

Struttura Dati	Accesso	Ricerca	Inserimento	Cancellazione	Spazio
ArrayList	O(1)	O(n)	O(1)/O(n)	O(n)	O(n)
Stack (ArrayList)	O(1)	O(n)	O(1)	O(1)	O(n)
Queue (LinkedList)	O(1)	O(n)	O(1)	O(1)	O(n)
Circular Queue	O(1)	O(n)	O(1)	O(1)	O(n)
Deque (ArrayDeque)	O(1)	O(n)	O(1)	O(1)	O(n)
Priority Queue	O(1)*	O(n)	O(log n)	O(log n)	O(n)
LinkedList	O(n)	O(n)	O(1)**	O(1)**	O(n)
HashSet	N/A	O(1)	O(1)	O(1)	O(n)
TreeSet	N/A	O(log n)	O(log n)	O(log n)	O(n)
HashMap	N/A	O(1)	O(1)	O(1)	O(n)
TreeMap	N/A	O(log n)	O(log n)	O(log n)	O(n)



SOTTOPADO = BUCHI

STACK<T> S;

S.GET(2) = 3 → O(1)

↓

TO CERCHIAMO DIRETTAMENTE

O(n)

ARRAY[100];

RICERCA

ALLA PEGGIO -> 100 SISTEMI

FOR(i = 1 : 100) - ~~~