

Esercizio 1. Riscrivete le procedure ENQUEUE e DEQUEUE per una coda viste a lezione in modo da rilevare underflow e overflow.

Esercizio 2. L'operazione INSERT per gli insiemi dinamici può essere implementata per una lista singolarmente concatenata nel tempo $O(1)$? E l'operazione DELETE?

Esercizio 3. Implementate uno stack utilizzando una lista singolarmente concatenata. Le operazioni PUSH e POP dovrebbero richiedere tempo $O(1)$.

Esercizio 4. Implementate una coda utilizzando una lista singolarmente concatenata. Le operazioni ENQUEUE e DEQUEUE dovrebbero richiedere tempo $O(1)$.

Esercizio 5. Considerate una tavola hash con 9 celle e la funzione hash $h(k) = k \bmod 9$. Partendo dalla tavola vuota, descrivete cosa succede quando si inseriscono in sequenza le chiavi

$$5, 28, 19, 15, 20, 33, 12, 17, 10,$$

risolvendo le collisioni mediante concatenamento.

Esercizio 6. Dovete memorizzare un insieme di n chiavi in una tavola hash di dimensione m . Dimostrate che, se le chiavi appartengono ad un universo U con $|U| > (n - 1)m$, allora esiste un sottoinsieme $U' \subseteq U$ di cardinalità n formato da chiavi che hanno tutte lo stesso valore di hash (i.e., esiste $i \in \{0, \dots, m - 1\}$ tale che $h(k) = i$ per ogni $k \in U'$), e quindi il tempo di ricerca nel *caso peggiore* per l'hashing con concatenamento è $\Theta(n)$.

Esercizio 7. Il professor Del Caso dichiara di aver scoperto un modo semplicissimo per definire una funzione hash che distribuisce uniformemente le chiavi in una tavola:

$$h(k) = \text{numero a caso in } \{0, \dots, m - 1\}.$$

Una tavola hash basata su questa funzione randomizzata si comporterebbe correttamente?

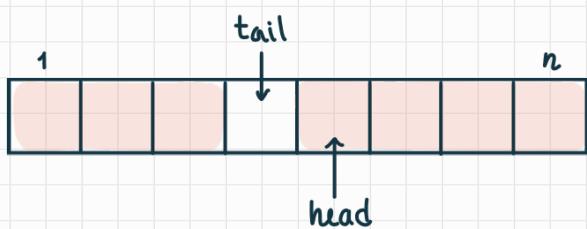
Esercizio 1. Riscrivete le procedure ENQUEUE e DEQUEUE per una coda viste a lezione in modo da rilevare underflow e overflow.

o CONTESTO : coda implementata tramite array (circolari)

- teniamo da parte informazione aggiuntiva :

- $Q.\text{head}$: l'elemento nella lista
- $Q.\text{tail}$: prossima posizione libera

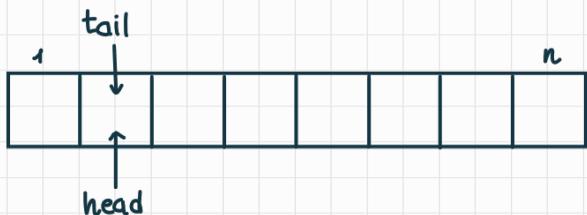
o caso 1: coda piena , $Q.\text{head} = (Q.\text{tail} + 1) \bmod Q.\text{dim}$



(*)

Possiamo memorizzare n elementi con questo sistema ?

o caso 2: coda vuota , $Q.\text{head} = Q.\text{tail}$



quindi ...

ENQUEUE (Q, x)

```
if FULL ( $Q$ )
    raise "overflow"
// ...
```

DEQUEUE (Q)

```
if EMPTY ( $Q$ )
    raise "underflow"
// ...
```

FULL (Q)

```
return ( $Q.\text{head} = Q.\text{tail} + 1$ )  $\bmod Q.\text{dim}$ 
```

EMPTY (Q)

```
return  $Q.\text{head} = Q.\text{tail}$ 
```

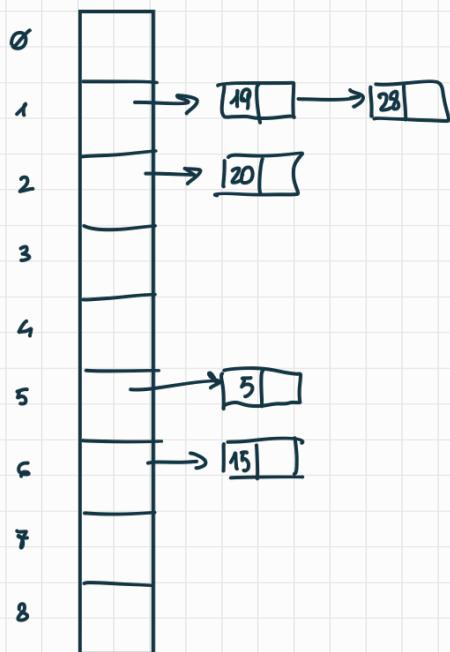
Esercizio 5. Considerate una tavola hash con 9 celle e la funzione hash $h(k) = k \bmod 9$. Partendo dalla tavola vuota, descrivete cosa succede quando si inseriscono in sequenza le chiavi

5, 28, 19, 15, 20, 33, 12, 17, 10,

risolvendo le collisioni mediante concatenamento.

o quante celle in T ?

→ op. "mod 9" al più 9 risultati → insieme $[0..8] \rightarrow |T| = 9$



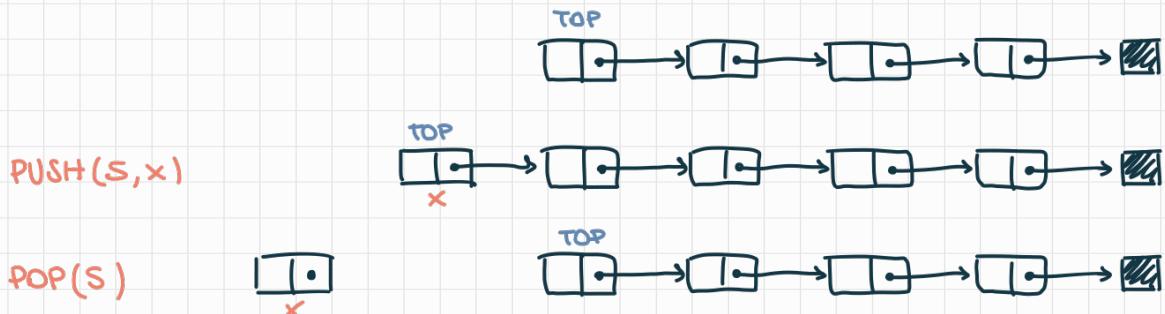
$x \quad 5 \ 28 \ 19 \ 15 \ 20 \ 33 \ 12 \ 17 \ 10$

$h(x) \quad 5 \ 1 \ 1 \ 6 \ 2 \ 6 \ 3 \ 7 \ 1$



Esercizio 3. Implementate uno stack utilizzando una lista singolarmente concatenata. Le operazioni PUSH e POP dovrebbero richiedere tempo $O(1)$.

o STACK : struttura dati LIFO



POP(S)



o elementi di base : nodi con dato + puntatore next

o vuoto se $S.\text{top} = \text{NIL}$

node
↓
PUSH(S, x) :

$x.\text{next} = S.\text{top}$
 $S.\text{top} = x$



$e \leftarrow S.\text{head}$

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

POP(S) :

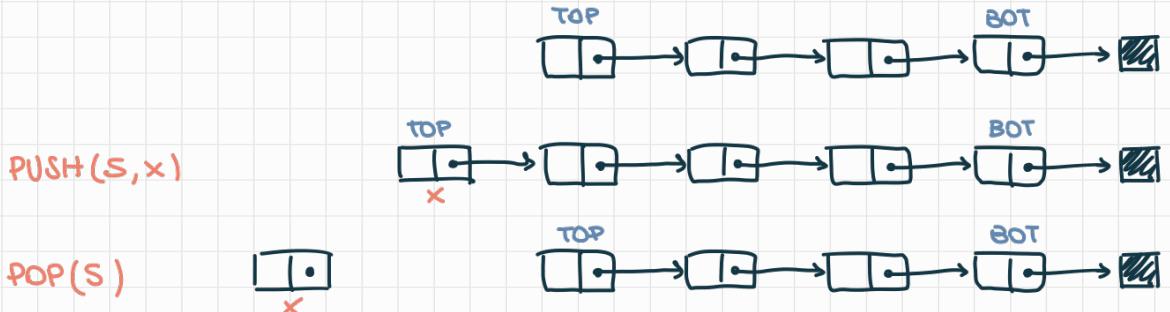
if $S.\text{top} = \text{NIL}$
raise "underflow"

$x = S.\text{top}$
 $S.\text{top} = S.\text{top}.\text{next}$

return x

Esercizio 3. Implementate uno stack utilizzando una lista singolarmente concatenata. Le operazioni PUSH e POP dovrebbero richiedere tempo $O(1)$.

o STACK : struttura dati LIFO



o elementi di base : nodi con dato + puntatore next

o vuoto se $S.\text{top} = \text{NIL}$

PUSH(S, x)

// supponiamo illimitato ...

$x.\text{next} = S.\text{top}$

if $S.\text{top} = \text{NIL}$ // stack vuoto

$S.\text{top} = x$
 $S.\text{bot} = x$

POP(S)

if ($S.\text{top} = \text{NIL}$)
raise "underflow"

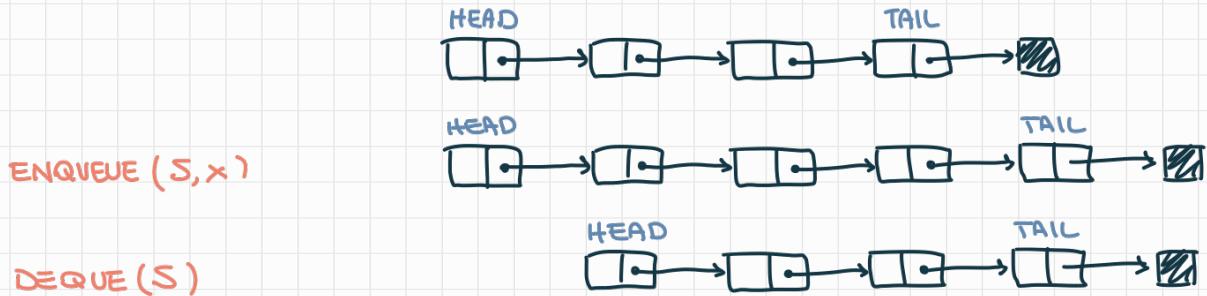
$e = S.\text{top}$
 $S.\text{top} = S.\text{top}.\text{next}$

if ($S.\text{top} = \text{NIL}$)
 $S.\text{bot} = \text{NIL}$

return e

Esercizio 4. Implementate una coda utilizzando una lista singolarmente concatenata. Le operazioni ENQUEUE e DEQUEUE dovrebbero richiedere tempo $O(1)$.

o CODA : struttura dati FIFO



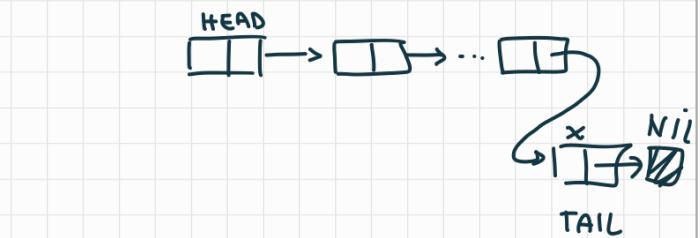
o elementi di base : nodi con dato + puntatore next

o vuota se $S.\text{head} = \text{NIL}$

ENQUEUE (S, x)

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

if $S.\text{head} = \text{NIL}$
 $S.\text{head} = x$



// primo elemento

else
 $S.\text{tail}.next = x$

$S.\text{tail} = x$

DEQUE (S)

if $S.\text{head} = \text{NIL}$
raise "underflow"

$e = S.\text{head}$

$S.\text{head} = S.\text{head}.next$

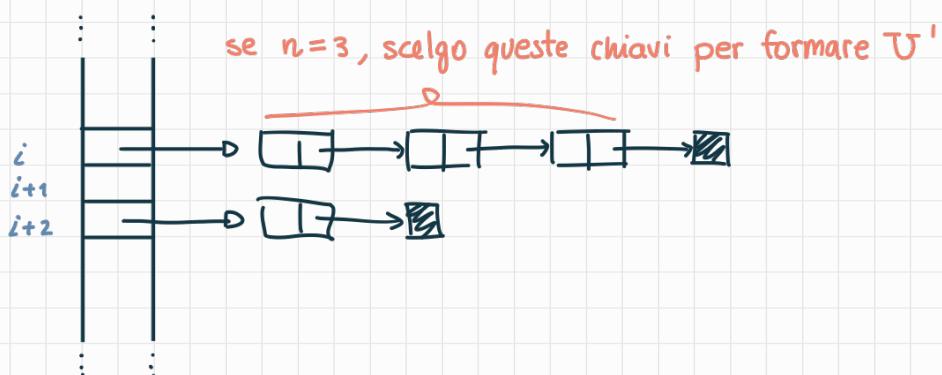
if $S.\text{head} = \text{NIL}$
 $S.\text{tail} = \text{NIL}$

return e

$$\alpha \geq n \quad \longleftrightarrow \quad \alpha > \frac{(n-1) \cdot m}{m} = (n-1)$$

Esercizio 6. Dovete memorizzare un insieme di n chiavi in una tavola hash di dimensione m . Dimostrate che, se le chiavi appartengono ad un universo U con $|U| > (n-1)m$, allora esiste un sottoinsieme $U' \subseteq U$ di cardinalità n formato da chiavi che hanno tutte lo stesso valore di hash (i.e., esiste $i \in \{0, \dots, m-1\}$ tale che $h(k) = i$ per ogni $k \in U'$), e quindi il tempo di ricerca nel caso peggiore per l'hashing con concatenamento è $\Theta(n)$.

- o supp. di memorizzare tutte le chiavi di U nella tavola hash, risolvendo le collisioni con liste concatenate.
- o se in almeno una lista sono collocate almeno n chiavi, allora banalmente scelgo n elem. di quella lista per formare U' .



→ per far verificare la negazione della proprietà cercata, è necessario che tutte le liste siano più corte di n .

- o Supponiamo quindi per assurdo che la PROPRIETÀ non sia vera. Abbiamo affermato che questo può succedere se ogni lista contiene meno di n elementi.
- o Sommiamo il num. di elementi indicizzati nella tavola

$$|U|$$

$$a < n \Leftrightarrow a \leq n-1$$

$$\begin{aligned}
 |U| &= T[0].len + T[1].len + \dots + T[m-1].len \\
 (*) &\leq (n-1) + (n-1) + \dots + (n-1) \\
 &= m \cdot (n-1)
 \end{aligned}$$

ossia stiamo dicendo che

$$|U| \leq m(n-1)$$

- o Assurdo! Infatti per hp $|U| > m(n-1)$

□

Esercizio 7. Il professor Del Caso dichiara di aver scoperto un modo semplicissimo per definire una funzione hash che distribuisce uniformemente le chiavi in una tavola:

$$h(k) = \text{numero a caso in } \{0, \dots, m-1\}.$$

Una tavola hash basata su questa funzione randomizzata si comporterebbe correttamente?

- quindi come funziona la funzione hash $h()$?
→ restituisce RANDOM, per ogni input, un numero a caso tra [0 .. m-1]
- e se devo poi cercare l'oggetto inserito ... ?
→ il risultato dell'hashing NON è DETERMINISTICO → devo scorrere tutta la tavola!

BONUS: caso dell' HASH UNIFORME SEMPLICE

- o per ogni chiave K , $P(h(K) = j) = \frac{1}{m}$, $j \in [0..m-1]$
- \Rightarrow ogni lista ha la stessa probabilità $1/m$ di essere scelta.
- o su n lanci, mi aspetto, in media, di trovare $\frac{n}{m}$ elementi in ognuna:
 - poiché ogni lista è equiprobabile e in particolare la prob. di essere scelta è $1/m$, significa che

$$\frac{\text{# elem. lista } j\text{-esima}}{n} = \frac{1}{m} \Leftrightarrow \text{# elem. lista } j\text{-esima} = \frac{n}{m}$$

caso favorevoli
caso totali

- più formalmente, possiamo associare il numero di elementi di ogni lista dopo n tentativi ad una

VARIABILE ALEATORIA BINOMIALE

(ELEM. PROB. E STATISTICA)

di parametri $\text{bin}(n, \frac{1}{m})$.

- non a caso, il VALORE ATTESO di questa VAR. ALEAT. è $\frac{n}{m}$.

CURIOSITÀ

- o quindi, il FATTORE DI RIEMPIMENTO α vale n/m .
- \rightarrow se $n = O(m)$ \Rightarrow operazioni in $O(1)$!
 - calcolo $h(\cdot)$ in $\Theta(1)$
 - ricerca in $O(1)$ poiché $\frac{n}{m} = O(1)$
 - inserimento $\Theta(1) \rightarrow$ in testa
 - cancellazione $O(1) \rightarrow$ arco e stacco l'elemento