Liste

giovedì 2 giugno 2022 16:27

- Dinamica: L'area di memoria può variare
- Vantaggio: Spazio
- Svantaggio: tempo (Accesso sequenziale)
- Come funziona:
  Una lista è composta da 1 componente statico, che viene chiamata
  Testa della lista, e questa fa riferimento al primo valore della lista.
  Ogni valore ha in sé il riferimento per il prossimo
- Sentinella : elemento che fa parte della lista però contiene valore fittizio
  -> Serve per avere una lista che mai è vuota

Tipologie:

- Lista dinamica semplice
  Puntatore Head[L]
  E punta ad una casella che è una nuova variabile
  Che ha le varie informazioni + puntatore chiamato "next"
- Lista dinamica doppiamente concatenate
  Oltre alla casella next, ha anche la casella previous
- Lista circolare
  Head[l].prev = ultimo elemento
  Tail[l].next = primo elemento

Prev	Key	next

```
Pointer Search(L, k):
    P = head[L]
    While p.key != k and p != null:
        P = p.next
    Return p
```

$T(n)_M = \Omega(1)$
$T(N)_p = O(n) \rightarrow k \neq L$

```
# Bellaaa questa soluzione
Void Insert(L, x):
    X.prev = null
    X.next = head[L]
    If Head[L] != null:
        Head[L].prev = x
    Head[L] = x

Delete(L, x):
    If x.prev != null:
        X.prev.next = x.next
    Else:
        Head[L] = x.next

    If x.prev != null:
        X.next.prev = x.prev
    Free(x)

Pointer listMin(L):
    If head[L] == null:
        Return null;
    pmin = head[L]
    patt = head[L].next
    While patt != null:
        If patt.key < pmin.key:
            Pmin = patt
        Patt = patt.next
    Return pmin


Pointer successor(L, k):
    If head[l]==null:
        Return null
    Patt = head[l]
```

```
        While patt.key <= k and patt != null:
                Patt = patt.next
        If patt = null:
                Return null
        Pmin = patt
        Patt = patt.next
        While patt != null:
                If patt.key < pmin.key and patt.key > k:
                        Pmin = patt
                Patt = patt.next
        Return pmin

Pointer Unire(l1, l2):
        If head[l1] == null:
                Head[l3] = head[l2]
                Head[l2] = null
        Else if head[l2] ==  null:
                Head[l3] = head[l1]
                Head[l1] = null
        Else:
                P1 = head[l1]
                While p1.next != null:
                        P1 = p1.next
                P1.next = head[l2]
                Head[l3] = head[l1]
                Head[l2] = null
                Head[l1] = null
        Return head[l3]
```