**Tredicesima lezione programmazione II**

**Gli insiemi dinamici**

Gli insiemi sono fondamentali per l’informatica e la matematica. **Mentre gli insiemi matematici sono immutabili (il loro contenuto una volta stabilito non può cambiare)**, **Gli insiemi manipolati dagli algoritmi possono crescere, ridursi o cambiare nel tempo. Per tale motivo sono detti dinamici.**

**Sequenze lineari**

**Sono insiemi finiti di elementi disposti consecutivamente**. **Gli elementi hanno un ordine**, che può essere o meno rilevante. **Ogni elemento può quindi avere associato un indice di posizione(univoco).** **L’operazione più elementare è l’accesso ai singoli elementi.**

**Quali sono le modalità di accesso alle sequenze lineari?**

* **Accesso diretto (array):** accediamo direttamente all’elemento ai senza dover attraversare la sequenza.
* **Accesso sequenziale (liste)**: raggiungiamo l’elemento attraversando la sequenza a partire da un suo estremo. Questo tipo di accesso ha costo O(i). (accedere a ai+1 (da ai) costa O(1))

**Allocazione della memoria**

Si hanno divisi modi di allocare la memoria che ricondiamo essere di due tipi (statica e dinamica)

1. **Array:** le locazioni di memoria corrispondenti ad elementi contigui sono contigue(sono locazioni contigue in memoria).

**Il nome dell’array corrisponde alla locazione del primo elemento(a[0]).** Se **x è l’indirizzo di a**, **a[0] si trova all’indirizzo x.** **a[1]** si trova all’indirizzo **x+1,** ecc.

1. **Liste:** gli elementi si trovano in locazioni di memoria non necessariamente contigue. Ogni elemento memorizza, oltre al proprio valore, l’indirizzo del valore successivo. Le liste si prestano bene a implementare sequenze dinamiche.

**È possibile ridimensionare un array?**

E’ necessario poter ridimensionare l’array(allocarne uno nuovo di diversa dimensione). L’operazione richiede O(n) tempo per ciascuna nuova allocazione.

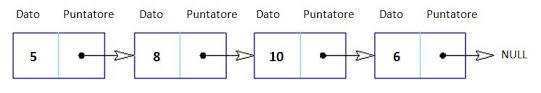
**Si può fare meglio?**

* **Costo cumulativo O(n)** per ciascun gruppo di noperazioni consecutive
* **Costo (distribuito):** O(1) per operazione.

**Cosa sono le liste concatenate?**

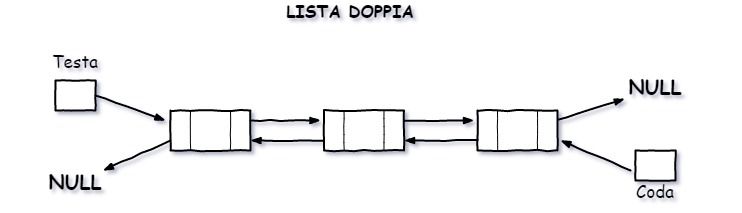
Una lista concatenata è una sequenza di elementi collegati l'uno all'altro da un puntatore. Gli elementi si compongono cioè di due parti(sono anche chiamati **nodi**):

1. **Parte:** contiene l'informazione (chiave)
2. **Parte:** costituita da un puntatore al successivo elemento

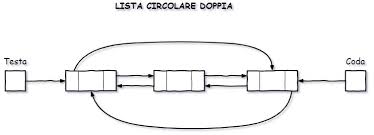


Si hanno svariati tipi di liste concatenate:

1. **Liste semplici**: ogni elemento contiene un unico puntatore che lo collega al nodo successivo. (esempio nell’immagine di sopra)
2. **Liste doppiamente concatenate:** ogni elemento contiene due puntatori, uno all'elemento precedente e l'altro al successivo



1. **Liste circolari semplici:** l'ultimo elemento si collega al primo in modo che la lista può essere attraversata in modo circolare.
2. **Liste circolari doppiamente concatenate:** l'ultimo elemento si collega al primo elemento e viceversa



**Come si dichiara un nodo della lista?**

Immagine che contiene testo, ricevuta, Carattere, schermata

Descrizione generata automaticamente

**La costruzione di una lista si può riassumere nei seguenti passi**

1. **Dichiarare il tipo del nodo ed il puntatore di testa**
2. **Allocare memoria per un nodo utilizzando l’operatore new assegnandone l'indirizzo ad un puntatore**
3. **Creare iterativamente il primo elemento ed i successivi**
4. **Ripetere finché vi siano nodi da immettere**

**Immagine che contiene testo, Carattere, ricevuta, algebra

Descrizione generata automaticamente**

**Immagine che contiene testo, schermata, diagramma, Carattere

Descrizione generata automaticamente**

**Aggiungere un elemento in cima, in mezzo o in coda**

Per aggiunge un elemento alla lista bisogna decidere dove inserirlo

* **In cima:** per inserire un elemento bisogna prima crearlo, poi spostarsi  al inizio della lista, nuovo deve puntare al indirizzo della cima, cosi la nuova cima diventa il nuovo elemento.
* **In mezzo**: per inserire un elemento bisogna prima crearlo, poi spostarsi   nella posizione in cui si vuole aggiungere l’elemento(nuovo). Supponiamo che si deve andare nella terza posizione. Il 3° elemento deve puntare a nuovo e nuovo deve andare alla terza posizione.
* **In coda**: per inserire un elemento bisogna prima crearlo, poi spostarsi  nella coda e far puntare la coda al nuovo elemento, a sua volta il nuovo elemento punterà a NULL

**Contare gli elementi di una lista**

Per eseguire questa operazione bisogna partire dalla cima e man mano controllare se è presente un elemento se è così incrementare un contatore che va inizializzato a 0.

**Eliminazione di un elemento dalla lista**

* **Per eseguire questa operazione (dalla cima)** bisogna prima far puntare elemento a cima e la cima punterà al successivo.
* **Per eseguire questa operazione (dalla coda)** bisogna posizionarsi all’ultimo nodo della lista e conservare il puntatore al nodo precedente in cui si deve inserire la costante NULL

**Ricerca di un elemento nella lista**

È molto simile alla ricerca in array. Semplicemente alla fine viene restituito un puntatore all’elemento trovato.

**Lista doppiamente concatenata**

**Aggiungere un nodo alla lista doppiamente concatenata in testa, in coda e in mezzo**

Per aggiunge un elemento alla lista bisogna decidere dove inserirlo:

* **In testa:** per aggiungere un elemento in testa bisogna eseguire i seguenti passaggi:

1. **Allocare dinamicamente un nuovo nodo, assegnarne l'indirizzo ad un puntatore nuovo e copiarvi dentro l’elemento che si vuole inserire.**
2. **Assegnare il campo succ del nuovo nodo al puntatore di testa p, ed il campo ant del primo nodo, se esiste, al nodo nuovo. Se la lista è vuota non fare alcunché.**
3. **Far puntare p al nuovo nodo.**

* **In mezzo:** per aggiungere un elemento **in mezzo o anche in qualsiasi posizione** bisogna seguire i seguenti passaggi:

1. **cercare la posizione dove bisogna inserire il nodo**
2. **allocare dinamicamente memoria al nuovo nodo puntato dal puntatore nuovo, e copiarvi il contenuto informativo.**
3. **far puntare il campo succ del nuovo nodo al nodo pos che va dopo la posizione del nuovo nodo (oppure a NULL in caso non vi sia alcun nodo dopo la nuova posizione); l'attributo ant del nodo successivo a quello che occupa la posizione del nuovo nodo che è pos, deve puntare a nuovo se esiste; se non esiste non fare alcunché**
4. **far puntare l'attributo suc del puntatore ant al nuovo nodo; l'attributo ant del nuovo nodo, farlo puntare ad ant**

* **In coda:** si applica lo spesso procedimento dell’inserimento in mezzo o in qualsia posizione

**Rimuovere un nodo dalla lista doppiamente concatenata**

1. **ricerca del nodo che contiene il dato; si deve avere l'indirizzo del nodo da rimuovere e l'indirizzo dell'antecedente (ant).**
2. **l'attributo suc del nodo anteriore (ant) deve puntare all'attributo suc del nodo da rimuovere, pos (se non è il primo nodo della lista); se è il primo della lista l'attributo p deve puntare all'attributo suc del nodo da rimuovere pos.**
3. **l'attributo ant del nodo successivo a quello da cancellare deve puntare all'attributo ant del nodo da rimuovere (se non è l'ultimo nodo della lista);**
4. **se è l'ultimo nodo della lista non fare alcunché si libera la memoria occupata dal nodo rimosso pos**

**Aggiungere un nodo alla lista circolare in testa, in coda e in mezzo**

1. **allocare memoria al nodo nuovo e riempirlo di informazione**
2. **se la lista è vuota far puntare il succ di nuovo al nodo stesso, e far puntare il puntatore di lista al nuovo**
3. **se la lista non è vuota si deve decidere dove collocare il nuovo, conservando l'indirizzo del nodo antecedente ant; collegare l'attributo suc di nuovo con l'attributo suc del nodo antecedente ant; collegare l’attributo suc del nodo antecedente ant con il nuovo.**

**Rimozione di un nodo dalla lista circolare**

**1. cercare il nodo ptrnodo che contiene il dato conservando un puntatore all'antecedente ant**

**2. far puntare il campo suc del nodo antecedente ant dove punta il campo suc del nodo da cancellare; se la lista conteneva un solo nodo si mette a NULL il puntatore p della lista.**

**3. se il nodo da rimuovere è quello puntato dal puntatore d'accesso alla lista, p, e la lista contiene più di un nodo, si modifica p per mandarlo dove punta il campo suc del nodo puntato da p (se la lista rimane vuota fare prendere a p il valore NULL).**

**4. da ultimo??, si libera la memoria occupata dal nodo eliminato**

**Lista circolare con sentinella**

In una lista con sentinella è presente un nodo speciale, detto "**sentinella", che non contiene alcuna informazione utile ma serve solo per marcare l'inizio (o la fine) della lista.**

La sentinella consente d’accedere in tempo O(1) al primo o ultimo elemento della lista: il primoelemento della lista è il successore della sentinella, mentre l'ultimoelemento della lista è il predecessore.