Strutture dati

Strutture dati dinamiche

Le strutture di dati dinamiche sono un modo per organizzare i dati di cui a priori non è nota la dimensione.
Il problema nel gestire questi dati sorge perché non esiste un numero prefissato indicante la loro quantità e durante l'esecuzione questo numero può aumentare e diminuire continuamente. Per poter trattare questo tipo di dati servono quindi strutture apposite, che vengono definite strutture di dati dinamiche.
Tipicamente le operazioni di queste strutture si dividono in:

- **Inserimento** di un elemento
- **Cancellazione** di un elemento
- **Modifica** di un elemento
- **Ricerca** di un elemento

Array dinamici

Le strutture di dati dinamiche possono essere pensate in termini di oggetti, in quanto sono composte da dati e da un insieme di operazioni che agiscono sui dati. Per implementare le strutture di dati dinamiche non possono essere usate strutture statiche come gli array, poi che la loro dimensione è fissa.
Per implementare le strutture di dati dinamiche si usano strutture dati che possono cambiare dimensione durante l'esecuzione del programma. Le strutture dati dinamiche più semplici sono gli array dinamici, che sono array che possono cambiare dimensione durante l'esecuzione del programma.
Java rende disponibile una struttura dati di questo tipo attraverso la classe **Vector** inclusa nel package **java.util**.

Vector

La classe **Vector** è una classe generica che permette di creare array dinamici di oggetti di un tipo specificato.
Questa classe implementa un array di oggetti dinamico nel senso che è basata su un array le cui dimensioni possono aumentare o diminuire in base ai dati che contiene.
L'array si adatta dinamicamente, aumentando quando vengono aggiunti nuovi dati e diminuendo quando i dati vengono eliminati. L'operazione di adattamento è eseguita automaticamente: viene allocato nuovo spazio quando serve e deallocato quando non è più utilizzato.

Vector

Come per gli array, gli elementi di un Vector sono indicizzati da un numero intero che va da 0 a n-1, dove n è il numero di elementi contenuti nel Vector. Per accedere agli elementi di un Vector si usa il metodo `elementAt(int index)`, che restituisce l'elemento in posizione index.

Per dichiarare un `Vector` ci sono 3 diversi costruttori:

```
Vector v = new Vector();
Vector v = new Vector(int initialCapacity);
Vector v = new Vector(int initialCapacity, int capacityIncrement);
```

Vector

Ogni situazione che richiede un array può essere risolta con un Vector, guadagnandone in flessibilità e memoria.

I metodi più importanti della classe Vector sono:

- `addElement(Object obj)` : aggiunge l'oggetto obj alla fine del Vector
- `removeElementAt(int index)` : rimuove l'elemento in posizione index
- `size()` : restituisce il numero di elementi contenuti nel Vector
- `elementAt(int index)` : restituisce l'elemento in posizione index

Vector

```
import java.util.Vector;
class Poligono {
  private Vector punti;
  public Poligono() {
    punti = new Vector();
  }
  public void aggiungiPunto(Punto p) {
    punti.addElement(p);
  }
  public void rimuoviPunto(int i) {
    punti.removeElementAt(i);
  }
  public int numeroPunti() {
    return punti.size();
  }
  public Punto getPunto(int i) {
    return (Punto)punti.elementAt(i);
  }
}
```

Vector

```
class Punto {
  private int x, y;
  public Punto(int x, int y) {
    this.x = x;
    this.y = y;
  }
  public int getX() {
    return x;
  }
  public int getY() {
    return y;
  }
}
```

La pila

Una pila è una struttura dati dinamica che permette di inserire e rimuovere elementi secondo una politica detta LIFO (Last In First Out), ovvero l'ultimo elemento inserito è il primo ad essere rimosso.
Gli elementi vengon aggiunti tramite un operazione detta **push**, mentre vengono rimossi tramite un'operazione detta **pop**.
Per interrogare una pila si usa l'operazione **top**, che restituisce l'elemento in cima alla pila senza rimuoverlo.

La coda

Una coda è una struttura dati dinamica che permette di inserire e rimuovere elementi secondo una politica detta FIFO (First In First Out), ovvero il primo elemento inserito è il primo ad essere rimosso. Gli elementi vengon aggiunti tramite un operazione detta **enqueue**, mentre vengono rimossi tramite un'operazione detta **dequeue**. O meglio **aggiungi** e **togli**.

La lista concatenata

Una lista concatenata o linked list è una struttura dati dinamica che permette di inserire e rimuovere elementi in qualsiasi posizione. Una lista concatenata è composta da una sequenza di nodi, dove ogni nodo contiene un elemento della lista e un riferimento al nodo successivo. Il primo nodo della lista è detto **testa**, mentre l'ultimo nodo è detto **coda**.
In C e C++ le liste concatenate sono implementate tramite puntatori, mentre in Java sono implementate tramite riferimenti agli oggetti.
I nodi vengono aggiunti tramite un'operazione detta **insert**, mentre vengono rimossi tramite un'operazione detta **remove**. Per interrogare una lista concatenata si cerca l'elemento desiderato tramite un'operazione detta **search** o **contiene**.

Gli alberi

Un albero è una struttura dati dinamica che permette di organizzare i dati in modo gerarchico. Un albero è composto da un insieme di nodi, dove ogni nodo può avere un numero arbitrario di figli. Il nodo senza figli è detto **foglia**, mentre il nodo che ha almeno un figlio è detto **nodo interno**. Il nodo senza padre è detto **radice**.
I collegamenti tra i nodi non formano cicli, ovvero non è possibile raggiungere un nodo partendo da se stesso.
Un albero è detto **binario** se ogni nodo ha al massimo due figli. Un albero binario è detto **completo** se tutti i livelli dell'albero sono pieni, ovvero ogni nodo ha esattamente due figli, tranne le foglie che non hanno figli.
Un albero binario di ricerca è un albero binario in cui ogni nodo ha un valore maggiore o uguale al valore del figlio sinistro e minore o uguale al valore del figlio destro.

Gli alberi

Le operazioni di modifica di un albero sono inserimento e cancellazione di un nodo. Le operazioni di ricerca di un albero sono ricerca di un nodo e visita dell’albero.