# Relazione Progetto Sparse Matrix

Fiorenza Gioele 851631

4/01/2022

## 1 Introduzione

Il progetto prevede l'implementazione di una Sparse Matrix (o matrice sparsa), dove solo gli elementi effettivamente inseriti, tramite apposita chiamata set, sono effettivamente memorizzati all'interno della matrice.
Per l'implementazione di questo dettaglio ho fatto uso di una lista concatenata di elementi, creata attraverso l'uso di 2 struct, la prima, chiamata "nodo" funge da rappresentazione interna della lista e contiene un attributo next che è un puntatore al nodo successivo e un'attributo della seconda struct per rappresentare il dato.
La seconda struct è "element" ed è stata implementata seguendo le direttive della consegna, quindi contiene un attributo per la x, uno per la y e il valore che è un dato templato dipendente dal dato templato passato alla classe sparse_matrix.

## 2 Descrizione della struct nodo

Ho deciso di creare questa struct, invece di posizionare l'attributo next su element, per eliminare possibili esposizioni verso l'esterno della struttura della classe, infatti se avessi implementato la sparse matrix in questo modo, quando un programmatore che fa uso della classe sarebbe andato a leggere i dati attraverso l'iteratore avrebbe avuto in mano l'element e il suo next e a quel punto avrebbe potuto chiamare una delete sul next e "rompere" la struttura interna della matrice.

### 2.1 Prestazioni

Attraverso questa struct è possibile andare a memorizzare solo che celle che vengono effettivamente inserite, infatti quando si chiama la set e non sono presenti elementi nella posizione specificata viene allocato un nuovo nodo sullo heap (con l'uso della new) e questo sarà eliminato con l'uscita di scope dell'oggetto matrice in cui è contenuto.

Un difetto di questa implementazione è la velocità di accesso ai dati, in quanto per leggere un dato nel caso peggiore devo scorrere tutta la lista di elementi.

## 2.2 Copy constructor

Anche se è un dettaglio minore in quanto la struct del nodo è privata e utilizzata dalla classe matrice, nel suo copy constructor ho deciso di non copiare il puntatore al next, in quanto non mi sembrava sensato che una copia dovesse condividere l'intero seguito della matrice di cui si sta facendo la copia, quindi nella mia implementazione ho deciso di impostare il next a nullptr, in questo modo sarà poi la matrice che deciderà come gestire la logica del next.

## 2.3 Eccezioni

Per quanto riguarda le eccezioni ho deciso di creare una classe custom di eccezioni per lanciare l'errore di out of range quando vengono sforati gli indici della matrice logica.