Programmazione ed Amministrazione di Sistema
Relazione Progetto C++

Leonardo Fraquelli
Matricola: 820651
Mail: l.fraquelli@campus.unimib.it

19 aprile 2019

Parte I

Implementazione di un Grafo Orientato

Capitolo 1

Introduzione

In seguito ad una attenta revisione del problema ho identificato i seguenti problemi da risolvere e affrontare:

1. Decidere come implementare la matrice.
2. Decidere come relazionare nodi e matrice di adiacenza.

La scelta di utilizzare una matrice booleana dinamica attraverso puntatori è stata pressoché immediata vista la proibizione sull'utilizzo di liste, l'utilizzo di valori booleani semplifica e riduce la dimensione dei dati allocati. [1].
Un nodo nella matrice si trova a un indice fissato, in base alla dimensione della matrice durante l'inserimento.
Non vi sono nodi duplicati, in quanto sarebbe impossibile[2] fare distinzioni.
Il grafo ha purtroppo problemi di efficienza sia nell'aggiunta che nella rimozione di un nodo, ricreando ad ogni operazione una matrice identica.

[1] La size di boolean è 1 byte, La size di int è 4 byte
[2] O estremamente difficile

Capitolo 2

Dati

Essendo una struttura dati, il Grafo necessita di essere templato per consentire un corretto inserimento dei dati.
Il grafo è composto da un insieme di nodi, una matrice di adiacenza, e un valore che ne indica la dimensione.
Include inoltre una templatura di Equals, necessaria per il confronto dei dati.
I dati sono organizzati come segue:

Nodes è l'insieme dei nodi templati, rappresentano gli identificatori che il grafo utilizzerà per rappresentarli;

Matrix è la matrice di adiacenza, di tipo booleano, rappresenta la presenza di archi tra gli identificatori

Equals è un funtore, viene utilizzato per il confronto dei nodi .

Size è un valore di tipo unsigned int che rappresenta la dimensione della matrice.

Innestiamo inoltre una classe che rappresenta il const-iterator di tipo forward sul grafo, che permette di iterare sui valori di Nodes.

Capitolo 3

Scelte Implementative

L'implementazione della struttura dati ha richiesto:

1. Generalizzare le funzioni per funzionare sui valori templati
2. Metodi per modificare,aggiungere,rimuovere dei dati, da cui segue che non deve essere possibile eseguire queste operazioni dall'esterno della struttura[1]

Da ciò deriva la scelta di utilizzare una classe, per la quale sono stati implementati i metodi fondamentali necessari:

Costruttore Vuoto che inizializza un grafo vuoto, di dimensione 0

Copy Constructor che inizializza un grafo identico a quello passato in argomento

Distruttore che elimina le allocazioni in memoria attraverso una funzione **clear**, rimuovendo nodi,matrice, e settando la dimensione a 0

Assegnamento che permette di assegnare i valori di un grafo ad un altro

La classe inoltre utilizza delle eccezioni che ereditano dalla gerarchia delle eccezioni standard per consentire una migliore identificazione dei problemi, nel caso in cui dovessero insorgere.
Le eccezioni non dispongono di costruttori in questo caso,pertanto sono state implementate come struct.

[1] Ad esempio la modifica dell'identificaore di un nodo è possibile solamente attraverso setNode

Capitolo 4

Funzioni

Le seguenti funzioni sono state implementate:

init	Che inizializza un grafo, senza assegnare valori agli archi, alloca la memoria necessaria al grafo.
clear	Che de-alloca il grafo, utilizzando delete su matrix e nodes, e setta la dimensione a 0.
operator()	Che permette di accedere agli elementi della matrice dal grafo. è stato ridefinito () e non [][] in quanto non è possibile eseguire questa operazione in C++
operator[ ]	Che permette di accedere agli identificatori del grafo
swap	Che ridefinisce swap della libreria standard.
getSize	Che ritorna la dimensione della matrice
hasEdges	Che ritorna il numero di archi esistenti
hasEdge	Che ritorna l'esistenza di un arco
exists	Che ritorna l'esistenza di un nodo
add	Che aggiunge un nodo. Lancia un'eccezione nel caso in cui questi sia già presente
remove	Che rimuove un nodo. Lancia un'eccezione nel caso in cui questi non sia presente
getNode	Che restituisce la posizione di un nodo.
setNode	Che cambia l'identificatore di un nodo. Lancia un eccezione nel caso in cui questi non sia presente, oppure il nuovo valore
setEdge	Che cambia il valore di un arco. Lancia un eccezione nel caso in cui i nodi utilizzati per identificare l'arco non esistano

Capitolo 5

Main

Il main del progetto è utilizzato unicamente a scopo di test.
I test testano tutte le funzioni elencate nel capitolo precedente.
Il grafo è stato testato su:

1. Grafo di tipo Int,con funtore "Equal-Int"
2. Grafo di tipo Char, con funtore "Equal-char"
3. Grafo di tipo Voce, una classe custom utilizzata nel precdente progetto Rubrica, con funtore "Equal-voce"

Il progetto non presenta leak o errori su Virtual-Box con OS: "Ubuntu 18.04.2 LTS"