Progetto c++

Programmazione e amministrazione di sistema

Nome e cognome: Lorenzo Di Vito

Matricola: 793128

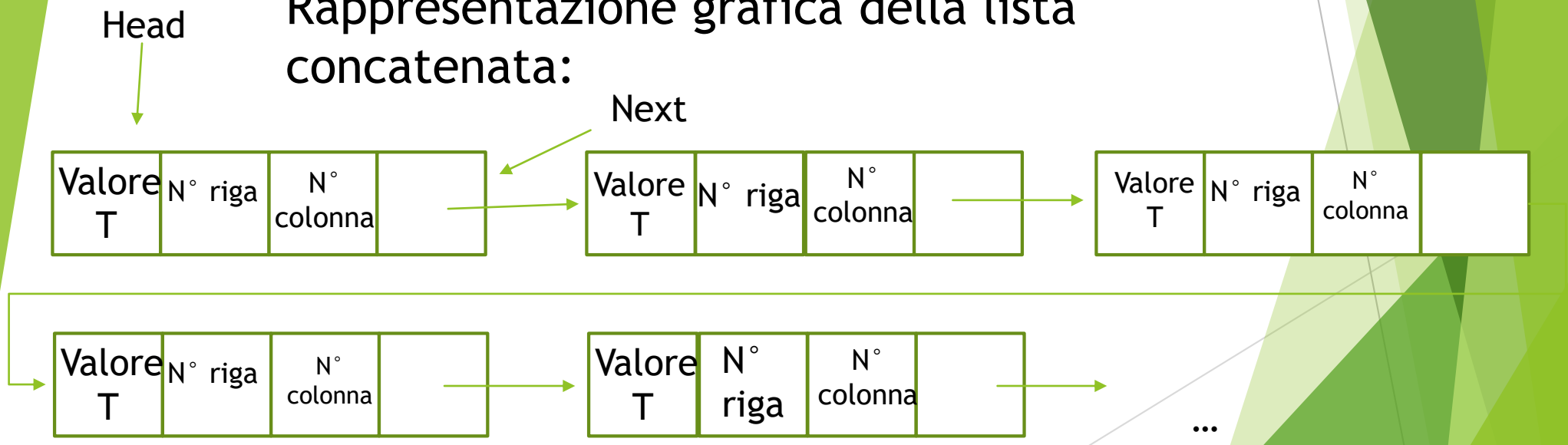
Email: l.divito1@campus.unimib.it

SparseMatrix

Matrice sparsa creata adottando una lista concatenata.

- La lista possiede una testa (head);
- Un successore (next);
- Valore di tipo T ;
- Numero della riga rappresentante il valore T;
- Numero della colonna rappresentante il valore T.

Rappresentazione grafica della lista concatenata:



Struttura interna della SparseMatrix

All'interno della matrice sparsa vi è:

- Struct node con:
  - puntatore next;
  - puntatore a struct element;
  - puntatore alla matrice sparsa a cui appartiene.

- Struct element con:
  - Valore T;
  - Numero riga;
  - Numero colonna.

Struttura interna della SparseMatrix

Node

- Contiene una sottostruttura element che conterrà il valore, il numero della riga e il numero della colonna (attributi);

- Per inserire il nuovo «nodo» istanzia la sottostruttura element interna e alloca quindi dinamicamente memoria per ospitarla (costruttore di node);

- Possiede tutti e 4 i metodi fondamentali. Tuttavia copy-costructor e copy-assignment non vengono mai usati ma inseriti comunque per possibili utilizzi futuri;

- Distruttore pulisce l'elemento contenuto nel nodo;

- Scelta di implementazione di element all'interno di node per astrarre il concetto di «contenitore» e anche per soddisfare la richiesta di restituzione di un element da parte dell'iterator.

Struttura interna della SparseMatrix

Element

- Non necessita di allocare dinamicamente memoria, è una struttura che contiene 3 attributi;

- Value è volutamente una dato e non un reference proprio per scelta di restituzione di una copia del dato e non del dato stesso presente nella sottostruttura (example: restituzione element da iteratori);

- Non possiede i 4 metodi fondamentali proprio perché quelli non presenti sono già generati di default dal compilatore (contiene solo dati) e, nonostante questo, questi non verranno mai utilizzati;

- Non possiede neanche il costruttore di default proprio perché contiene solo dati e personalmente ritengo che sia inadatto istanziare un elemento senza valori ai suoi attributi;

- E' gestito quasi interamente da node (che alloca dinamicamente la memoria e, con il distruttore, lo elimina).

Attributi della SparseMatrix

- Head: contenente il puntatore alla testa della lista rappresentante la matrice sparsa;
- Default_value: restituito come reference costante per evitare modifica dall'esterno e senza setter per evitare la modifica volontaria dell'utente (una volta inserito l'utente ha la garanzia che il default_value non venga modificato);
- Totale elementi inseriti;
- Numero totale righe;
- Numero totale colonne;

Dove possibile viene usato constness per rendere costante e immutabile variabili.

Metodi fondamentali
Sparse Matrix

- Copy-costructor che crea una matrice sparsa partendo da una come parametro;

- Copy-assignment che crea una matrice sparsa chiamando anche copy-costructor e utilizzando funzione swap;

- Costruttore default senza parametri non presente perché ritengo che la matrice sparsa debba avere per forza un valore di default inserito dall'utente (e anche per la consegna che ho interpretato);

- Distruttore che pulisce la lista avvalendosi del metodo clear_sparse_matrix che chiama clear_helper di tipo ricorsivo per l'eliminazione di ogni nodo della lista.

Copy-costructor e assignment avranno una possibile eccezione dovuta dalla add nel caso in cui non vi è memoria disponibile. Try-catch non inserito in quel punto volontariamente proprio perché vi è già la cattura nella add.

Costruttore di copia Q->T SparseMatrix

E’ un costruttore templato proprio perché riceve in ingresso un tipo Q;

Il costruttore usa lo <static_cast>(casting implicito) per il casting di una determinata primitiva di tipo Q a una di tipo T(float->int …);

Si avvale degli iteratori per scorrere la lista rappresentante la matrice sparsa Q e richiama la add per la creazione della matrice di tipo T.

Anche in questo caso copy constructor e assignment avranno una possibile eccezione causata dalla add nel caso in cui non vi sia memoria disponibile. Non gestita in questa classe volontariamente in quanto gestita dalla add.

Funzione Add

Aggiunge un elemento alla matrice adottando la seguente regola:

- Viene creato il nodo;
- Ci si sposta lungo la lista concatenata finchè la riga da inserire è minore della riga del nodo;
- Mantenendo la riga uguale ci si sposta lungo le colonne finchè la colonna da inserire è minore della colonna del nodo;
- si modificano i puntatori inserendo il nuovo elemento.

Tutto ciò viene eseguito contemplando i vari casi di errore (l'inserimento di riga e colonna minori di 0)

Quando viene allocato dinamicamente un nodo vi è la possibilità di memoria non disponibile:
In tal caso verrà pulita completamente la matrice per evitare memoria dinamica non correttamente deallocata.
Successivamente lancio dell'eccezione che farà terminare il programma in modo anomalo

Funzione Operator()

Ricerca elemento

La ricerca di un elemento avviene nel modo analogo della add:

- Si scorre la lista prima secondo le righe fino a che la riga è minore della riga da ricercare ;
- Successivamente tenendo la riga fissa, secondo le colonne finchè sono minori della colonna da ricercare;
- Si controlla se esiste un nodo con riga e colonna uguali;
- In tal caso si restituisce il valore di tale nodo;
- Altrimenti si restituisce quello di default;

In caso di riga o colonna minori di 0 oppure di riga e colonna superiore alla grandezza della matrice viene lanciata un eccezione.

Funzione Evaluate
Valutazione predicato

E’ una funzione templata che riceve in ingresso un predicato P e una matrice sparsa M;

- Vengono usati gli iteratori per verificare se i «nodi» soddisfano il predicato P;

- Totale degli elementi della matrice equivale a max(riga) * max(colonna)

- Totale elementi non di default contati nel ciclo for della verifica del predicato P

- Elementi contati che soddisfano il predicato (caso in cui anche default_value è compreso) : Totale elementi sottratti al totale dei valori NON di default sommati al numero di valori inseriti dall’utente che soddisfano il predicato P

- Elementi contati che soddisfano il predicato (caso in cui default_value non è compreso): non vengono considerati gli elementi di default

Iteratori

- Di tipo const e non const, hanno come attributi privati il nodo n;

Attributi pubblici Traits:

- Value_type : node:element;
- E gli ultimi due attributi riferiti a pointer e reference sono costanti nel caso di const_iterator proprio perché è il caso di iteratori costanti (dato non modificabile);

Oltre ai vari metodi di:

- Confronto tra iteratori(==; !=);
- Metodi fondamentali (compreso copy-costructor aggiuntivo in const_iterator per copiare un iteratore costante da uno non costante);
- Metodi begin() ed end() per restituire la testa e la coda della matrice. In caso di matrice vuota verrà lanciata una eccezione empty_sparse_matrix().

Vi sono:

- Con l'operatore di deferenziamento «*» viene ritornata la struttura element contenete il valore, la riga e la colonna (attributi);

- Con l'operatore di reference «&» viene ritornato il reference alla struttura element;

Utente

L’utente potrà :

- Creare una matrice sparsa inserendo un default_value;
- Inserire valori in determinate posizioni secondo le sue esigenze;
- Ricercare elemento in una determinata posizione;
- Stampare la matrice sparsa;
- Costruire matrice sparsa a partire da un altra (anche di tipo diverso Q);
- Verificare con evaluate quanti elementi della matrice soddisfano un determinato predicato definito;
- Avvalersi degli iteratori per scorrere la matrice e stamparla.

Main

- Creato con diversi tipi di primitive (anche stringhe) e con varie strutture e classi con all'interno attributi;

- Vi è un try- catch situato apposta in situazioni specifiche(volutamente) per verificare il corretto funzionamento delle eccezioni e la gestione dei possibili casi di errore;

- Try-catch inserito anche nel main per possibile eccezioni di allocazione memoria oppure di costruzione matrice con dimensione 0 (matrice vuota);

In ogni classe vi è anche la verifica di:
- Inserimento;
- Ricerca;
- Stampa;
- Test del costruttore di copia e assignment.