Grafo

Descrizione Intestazione Classe:

Il programma contiene un file main, una libreria grafo2.h, con la sua implementazione nel file grafo2.cpp.

La classe è composta da un puntatore a lista con membri interi, composto da, rispettivamente: numero nodo, costo e capacità.

La classe possiede anche una dimensione fissata e scelta dall’utente, che sarà necessario per istanziare il vettore di liste, dove l’indice rappresenta il nodo padre dei figli.

Per fare un esempio:

[0]-> (1,3,5)->(2,8,5)

[1]->(2,4,11)

[2]

L’esempio mostra un output riprodotto dal programma void stampa\_grafo(), da cui possiamo dedurre informazioni dal grafo. Dal nodo 0 si arriva al nodo 1 e 2, con coppia di valori (costo,figlio) pari, rispettivamente, a (3,5) e (8,5); dal nodo 1 si arriva al nodo 2 con coppia (4,11) e il nodo 2 è solamente un punto di arrivo. Come si può notare,i nodi partono da 0 fino a n-1(non vi confondete, vi raccomando!)

Il limite di questo codice è che la lista sarà di tipo semplice, e non a doppio verso e non si farà uso di matrici di adiacenza/incidenza(si sacrifica l’efficienza per scrivere un codice più semplice).

Oltre al costruttore e distruttore della classe e la funzione di stampa, abbiamo un costruttore di copia, varie funzioni di utilità, e le funzioni per applicare gli algoritmi dei cammini minimi di Dijkstra e l’algoritmo del taglio minimo di Ford Furkersord.

Funzioni Pubbliche:

Ecco un elenco delle funzioni pubbliche che l’utente potrà utilizzare:

* Grafo(int), costruttore che alloca dinamicamente il vettore di liste, come spiegato nella sezione precedente. Non viene gestito il caso di ingressi negativi. Mi affido al buonsenso dell’utente che dovrebbe sapere che non esistono grafi con un numero di nodi negativi.

All’utente verrà chiesto, oltre alla dimensione del grafo, di inserire, per ciascun nodo, il numero di nodi figli da inserire e poi tutte le triplette necessarie per ricoprire tutti i valori, seguendo l’ordine(nodo,costo,capacità).Non vi preoccupate dell’ordine delle triplette, perché, attraverso il selection sort(inefficiente, ma più conveniente per la stesura del codice), saranno inseriti i nodi del vettore di liste in ordine crescente per ciascun indice del grafo.

* ~Grafo(), distruttore;
* void stampa\_grafo, funzione che stampa i grafi, come mostrato nella sezione precedente
* void Cammini\_Minimi(int partenza,int arrivo, bool& errato), funzione che applica l’algoritmo di Dijkstra e che ci illustra passo dopo passo(andando a capo per ciascun passo):

1) i nodi scelti(N={…}),

2) l’insieme stella dell’ultimo nodo scelto(CF(nodo)={…}),

3)i predecessori(p=(…))

4)l’etichetta di ciascun nodo(Pi=(…)).

Ciò che è scritto tra parentesi sono gli output che l’utente deve aspettarsi passo dopo passo nell’output di questa funzione. L’input è sanitizzato e viene segnalato l’errore al main attraverso errato;

* void Ford\_Furkersord(int partenza, int arrivo,bool &arrivo) , funzione che applica l’omonimo algoritmo, che ci produce, passo dopo passo(andando a capo per ciascun passo):

1)il percorso dei cammini minimi(Percorso: “…->…”),

2)il delta trovato(delta = )

3)il valore del taglio ricavato iterativamente(v =).

Ci illustra, alla fine del passo: Nt e Ns, che sono i nodi, rispettivamente, che non sono arrivati all’ultimo passo e quelli che, invece, ce l’hanno fatta. L’input è sanitizzato e viene segnalato l’errore al main attraverso errato.

**Attenzione:** **I nodi partono da 0 a N-1, con N numero di nodi, facendo uscire le posizioni decrementate di 1 rispetto all’output atteso rispetto ai nodi che partono da 1**

Main:

Il main consente all’utente di poter ripetere le seguenti operazioni, nel caso in cui avesse commesso l’errore nei seguenti casi,secondo lo stesso ordine con cui sono esposti:

* l’utente ha digitato per sbaglio le dimensioni sbagliate
* l’utente ha digitato la combinazione di triplette sbagliate(che vengono stampate a video per questo specifico motivo)
* l’utente ha sbagliato a digitare partenza e arrivo del nodo
* l’utente non si accorge di aver sbagliato a digitare partenza e arrivo(valori negativi o troppo grandi)

Per ciascuno di queste possibilità viene chiesto all’utente di digitare 1 per andare avanti o digitare 0 per ricominciare tutto da capo, ad eccezione degli ultimi 2 casi, in cui risulterebbe veramente scomodo ricominciare da capo. Negli ultimi 2 casi ci si limita a ripetere la digitazione di partenza e arrivo.

**Attenzione:** **Il caso partenza == arrivo non è considerato un errore dell’utente, ma un caso particolare da cui si esce, poiché abbiamo già raggiunto il cammino minimo**