Relazione progetto Algoritmi e Struttere Dati

Gruppo 1
Giacomelli Riccardo
Giacomo Pandini

Struttere dati:

Struttura graph_struct:
E' la struttura principale del progetto, in quanto con essa identificheremo la rete di interconnessione.
La struttura graph_struct conterrà un intero che indicherà il numero dei nodi presenti in un certo istante nel grafo, un puntatore di tipo node_struct alla sentinella della lista degli aeroporti ed un intero rappresentante il contatore id dei nodi.
Questa struttura viene utilizzata tramite un puntatore *graph ad essa.

Struttura node_struct:
Questa struttura viene utilizzata per rappresentare gli aeroporti, nonché i nodi del grafo.
La struttura contiene un intero contenente l'id del nodo, un stringa per il codice IATA del aeroporto, un puntatore di tipo struct_edge alla lista contenente la stella uscente del nodo, un puntatore di tipo node_struct all' elemento successivo della lista dei nodi ed un intero indicante il flag visitato.
Questa strutta verrà utilizzata mediante un puntatore *airport ad essa.

Struttura edge_struct:

Questa struttura identifica gli archi di collegamento tra i nodi del grafo.
La struttura contiene due valori interi indicanti costo e distanza,un puntatore di tipo edge_struct al elemento successivo della lista di archi e un puntatore di tipo node_struct al nodo di destinazione.
Questa struttura verrà utilizzata mediante un puntatore *route ad essa.

Struttura coda:

La struttura e' di supporto alle elaborazioni delle funzioni più complesse, essa gestisce una coda circolare.
La struttura contiene due puntatori di tipo coda indicanti l'elemento precedente e successivo della coda rispetto al elemento considerato e un puntatore di tipo node_struct ad un nodo del grafo.
Questa struttura verrà utilizzata mediante un puntatore *tail ad essa.

Struttura tree_struct

Questa struttura e' utilizzata per gestire un albero utile al esecuzione di una funzione.
La struttura contiene un puntatore di tipo node_struct ad un nodo del grafo, contiene due puntatori di tipo tree_struct uno che punta al elemento successivo e uno che punta all'elemento padre ed infine il campo valore di tipo intero.

Funzioni del progetto

Creagrafo:
Questa funzione crea il grafo che rappresenterà la rete di interconnessione. Alloca lo spazio necessario per una struttura graph, inizializza il numero dei nodi e id a 0, crea la sentinella della lista contenente gli aeroporti. La funzione ritorna il puntatore al grafo creato.

Addnode:
La funzione aggiunge un nodo al grafo. Riceve in input il grafo in cui verrà inserito il nodo e il codice IATA del nodo inserito. Il nodo e' stato creato allocando la memoria necessaria, inizializzando i valori dei campi in base ai valori ricevuti in input e creando la sentinella della lista degli archi uscenti.

Addege:
La funzione aggiunge un arco non orientato tra due nodi del grafo.
Riceve in input i codici IATA dei nodi tra cui verrà creato l'arco e i valori interi di costo e distanza.
Alloco in memoria lo spazio necessario per un arco, inserisco i parametri presi in input e inserisco l'arco nella stella uscente di uno dei due nodi. Ripeto l'operazione per l'altro nodo.

deleteNode:
La funzione elimina dal grafo il nodo desiderato.
Riceve in input il codice IATA del nodo da eliminare dal grafo, lo ricerca nel grafo e lo elimina. Successivamente aggiorna il grafo. Il nodo eliminato deve essere isolato.

delEdge:
La funzione elimina l'arco compreso tra due nodi del grafo.
Riceve in input i codici IATA dei nodi e richiama due volte la sottofunzione helpdeledge.
Helpdeledge trova i nodi nel grafo ed elimina dalle stelle uscenti dei due nodi gli archi che li collegano fra di loro.

listAirport:

La funzione stampa l'elenco dei nodi partendo da un nodo effettuando una visita DFS.
Riceve in input il codice IATA del nodo da cui partirà la visita, lo ricerca nel grafo e richiama la funziona DFS su di esso, infine richiama la funzione resetvisitato per riportare il flag visitato di tutti i nodi a 0.
La funzione DFS riceve un nodo in input, lo stampa e lo marca come visitato poi per ogni suo adiacente controlla se e' visitato e se non lo e' richiama la funzione DFS su di esso.
Partendo da un nodo desiderato chiama una visita DFS che mi stampa collegati.

listDestinations:
La funzione stampa la lista di nodi raggiungibili da un nodo tramite una visita BSF del grafo.
Riceve in input il codice IATA del nodo da cui inizieremo la visita e cerca il nodo nel grafo. Crea due code utili alla risoluzione dell'algoritmo della funzione e inserisce in una delle due il valore del nodo trovato. Ora finchè entrambe le code non saranno vuote controllerà quali delle due non contiene elementi ed eseguira la funzione helpBFS passando come primo parametro di input la coda vuota e come secondo la coda piena. Quando entrambe le code saranno vuote richiamerà la funzione resetvisitato.
La funzione helpBFS riceve in input le due code, finchè quella piena non si svuota del tutto estrae ogni elemento della coda marcandolo come visitato e inserisce nella coda vuota i suoi adiacenti non ancora visitati e non in coda. Alla fine dell'elaborazione avremo la coda piena che ora è vuota e la coda vuota che risulterà piena degli adiacenti non ancora visitati dei nodi che abbiamo estratto. Così facendo è stato possibile la stampa dei nodi in relazione alla distanza unitaria dal nodo preso come input dalla funzione listDestinations, infatti la stampa dei nodi avverrà prima per i figli del nodo radice poi per i figli di quei nodi e cosi via.

findMinPath:
Funzione di ricerca del cammino minimo tra due nodi del grafo adottando l'algoritmo di Dijkstra.
La funzione ricevera in input i codice IATA dei nodi di partenza e arrivo del cammino e una stringa indicante se l'algoritmo dovrà essere eseguito in base al costo o in base alla distanza. Inizialmente creerò una coda di supporto alla funzione e un albero che verrà utilizzato come albero di copertura minimo del grafo. L'albero verrà gestito attraverso una lista di padri, ogni elemento della lista conterrà un nodo del grafo, un riferimento all'elemento successivo e un riferimento al padre dell'elemento nell'albero. Successivamente inserisco tutti i nodi del grafo nell'albero organizzati come figli del nodo di partenza del cammino tramite la funzione listapadri. Inserirò nella coda il nodo di partenza e finchè la coda non sara vuota richiamerò prima minFuoricoda ,che troverà l'elemento minimo della coda, successivamente richiamerò o adiacenti costo o adiacenti distanza dipendentemente se l'algoritmo verrà eseguito sul costo o sulla distanza. Quando la coda sarà vuota avrò l'albero di copertura minimo con radice il nodo di partenza del cammino, non mi resterà che cercare il nodo di arrivo del cammino. Stamperò infine il cammino tramite la funzione stampaCammino.
Adiacenticosto od adiacenti distanza sono le funzioni principali di findMinPath in quanto realizzano l'algoritmo di Dijkstra, dopo aver ricevuto in input il nodo di valore minimo della coda visitano la sua

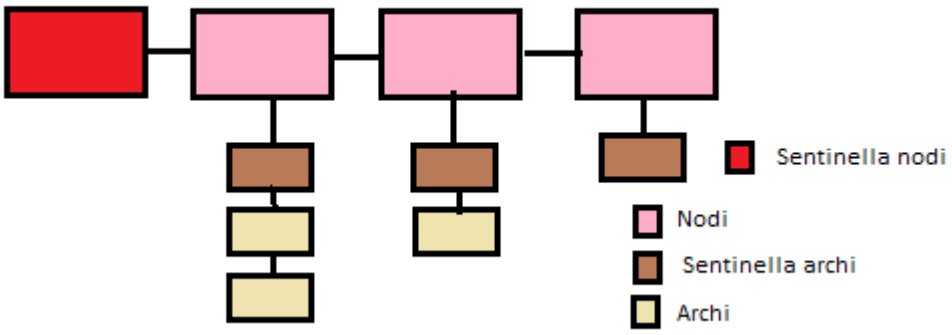
stella uscente e per ogni elemento di essa confrontano i valori presenti nell'albero con i valori di distanza degli archi del grafo. Se gli archi minimizzano la distanza dell'albero aggiornerò l'abero e inserirò il nodo nella coda nel caso non sia già presente.

Struttura generale del progetto:

La rete di interconnessioni è gestita attraverso un grafo , costituito da una lista di liste di figli.
Ogni nodo del grafo e' un elemento della lista ed identificherà un aeroporto della rete.
Ogni nodo a sua volta conterrà la lista degli archi colleganti gli archi adiacenti. Gli archi

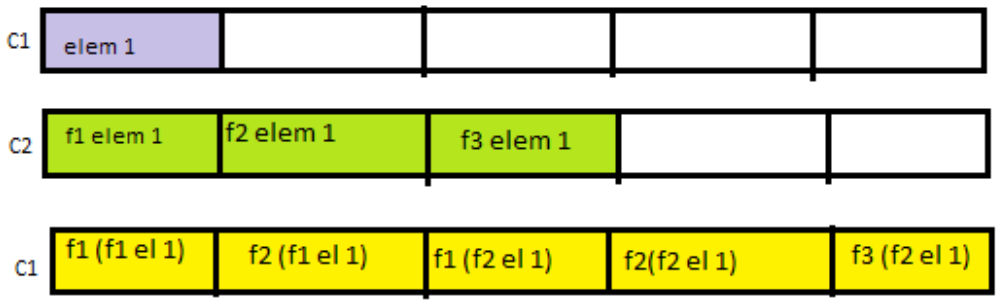
Scelte del progetto e soluzioni adottate

La struttura principale grafo è dunque gestita tramite una lista di figli in cui ogni elemento ha oltre i valori standard di ogni nodo anche un puntatore all'elemento successivo. La lista è monodirezionale e a inizio lista troviamo una sentinella mentre il puntatore dell'ultimo della lista puntera a NULL. Ogni nodo inoltre ha un puntatore alla sentinella di un ulteriore lista, la lista degli archi. Quest'ultima è gestita in modo analogo alla lista dei figli, ogni arco avrà oltre ai valori standard un puntatore all'elemento successivo della lista e come nella precedente lista a capo vi sarà una sentinella e alla fine un elemento NULL.



Gli inserimenti dei nodi e degli archi vengono effettuati immediatamente dopo la sentinella in modo da non dover scorrere tutta la lista. Le eleminazioni invece consisono solo nell'aggiornare il puntatore dell'elemento precedente deallocando la memoria del nodo o arco liberato quindi si limiteranno alla ricerca del nodo o dell'arco prestabilito.
La funzione di stampa dei nodi tramite visita DFS presuppone che per stampare tutti i nodi il grafo sia connesso. E' stato inoltre necessario aggiungere ai nodi una variabile visitato in modo da tener traccia dei nodi visitati dalla funzione.
Nella funzione listDestinations è stato necessario utilizzare due code gestite come liste circolari per riuscire a stampare i nodi in base alla distanza unitaria dal nodo iniziale. Infatti per riuscire a spampare

esattamente gli elementi è stato necessario creare un alternarsi tra le due code. Allo svuotamento di una corrisponderà il riempimento dell'altra finchè entrambe non saranno vuote.



La funzione findminpath utilizza l'algoritmo di Dijkstra per creare un albero di copertura minimo formato da una lista di padri da cui si potra ricavare il cammino minimo tra due nodi. Per arrivare a compimento dell'algoritmo si avvale di una coda utilizzata come una normale lista. Ad inizio dell'esecuzione della funzione verra creato l'albero ponendo il nodo di inizio cammino come padre di tutti gli altri e inizializzando le loro distanze da esso a MAX INT. Preso il nodo di partenza verranno confrontate le distanze dei nodi nell'albero rispetto alle distanze dei nodi nel grafo aggiornando l'albero nel caso che quest'ultime ne minimizzino i valori. Se la distanza di un nodo verrà cosi modificata il nodo sarà inserito nella lista dalla quale successivamente verrà estratto il nodo con distanza minore e su cui si ripetranno le operazioni di minimizzazione delle distanze. L'algoritmo continuera finche la coda non sarà vuota, al termine di ciò si avrà l'albero di copertura minimo che permetterà di trovare il valore del cammino minimo tra i due nodi passati alla funzione. L'ultilizzo della struttura lista dei padri per creare l'albero è utile in quanto ogni nodo avra un solo padre e ciò facilita molto le operazioni di aggiornamento dell'albero.