Mohamed Amine Hamdi Matricola: 260167 28/01/2020

# ALGORITMI E PROGRAMMAZIONE

## Relazione esame del 28/01/2020

I moduli usati per la realizzazione dell'esercizio sono i seguenti:

- GRAPH.H: libreria classica, che utilizza un ADT di prima classe “Graph” ed un quasi ADT “Edge”
- ST.H: libreria classica per l'utilizzo di una tabella di simboli.
- MAIN.C: In questo esercizio nel main vengono solo usate le funzioni di allocazione/deallocazione e altre operazioni sul grafo

In questo caso ho preferito la lettura del grafo fornito da file mediante una matrice delle adiacenze. Poichè il file contenente le informazioni relative al grafo era standard è stata usata solamente una funzione di libreria, la GRAPHload, che riceve un file e legge alla prima riga il numero di vertici, chiamando così la GRAPHinit (funzione di allocamento di un grafo), in seguito legge g->V (numero di vertici appena letto) e memorizza gli indici in una tabella di simboli, in seguito legge gli archi e li inserisce nella matrice di adiacenze.

Il problema richiede successivamente l'individuazione del k-core, questa operazione viene svolta mediante l'utilizzo della funzione “kcore”, ovvero una funzione wrapper che legge un valore k da tastiera che raffigura il grado ricercato e calcola il grado di partenza di tutti i vertici con un vettore degree, infine chiama la funzione ricorsiva che genera l'insieme delle parti dei vertici, e grazie alla presenza del vettore degree non vengono mai presi nella soluzione i vertici che originariamente avevano grado inferiore a k.
La funzione “cntrlKcore” controlla se e' una soluzione è valida andando a calcolare il grado di ogni vertice preso nella soluzione (tenendo conto della presenza di archi con vertici non presi nella nostra soluzione possibile), nel caso tale grado sia minore del valore k letto da tastiera viene ritornato il valore 0.
Nel caso in cui la funzione di controllo ritorna 1 viene calcolato il numero di vertici presi, e se tale soluzione è massimale al momento viene salvata nel vettore best e viene aggiornato il massimo, nel codice dell'esame non era presente questa parte poiché per una mia svista pensavo servisse solamente una soluzione qualsiasi quindi adoperavo di un'uscita non strutturata.

Infine per l'ultima richiesta dell'esercizio viene chiamata la funzione “jEdge”, ovvero una funzione wrapper che legge il valore j di archi da rimuovere da tastiera, alloca dinamicamente un vettore soluzione di archi, un vettore di archi, che viene poi riempito grazie alla funzione di libreria “GRAPHedges”, un vettore di interi che serve da marcatore degli archi ed infine chiama la funzione ricorsiva per la ricerca del sottoinsieme voluto.
Inizialmente la funzione ricorsiva era strutturata con il modello delle disposizioni, poiché l'ordine non è influente sulla generazione di una soluzione è stata invece utilizzato un modello di combinazioni semplici, vengono generati g->E-j archi (numero di archi meno il numero di archi da togliere), entrando nel caso terminale viene effettuato un controllo sulla connessione del grafo generato e grazie alla funzione wrapper “cntrlEdge” vengono allocati dinamicamente il vettore visited ed il vettore touched che hanno una funzionalità simile ai tempi di scoperta e tempi di fine elaborazione di una visita in profondità, nel codice consegnato all'esame nella funzione ricorsiva non veniva utilizzato il vettore touched e veniva controllato solo il vertice di partenza di un arco sebbene sia un grafo non orientato.
Terminata la visita viene effettuato un controllo sul vettore visited e se è presente un vettore non visitato viene ritornato 0, e nel caso contrario 1.

Se il controllo della possibile soluzione ritorna 0, ovvero il grafo è stato sconnesso vengono stampati gli archi non presi dalla soluzione, ovvero gli archi che hanno il marcatore a 0.

La ricerca del nome del vertice per poi stamparlo è effettuata grazie alla funzione di libreria STsearchByIndex.

Al termine della stampa viene usata un'uscita non strutturata per terminare la generazione di nuove soluzioni.

Riepilogando le differenze significative tra il compito cartaceo ed il codice allegato sono:

- Presenza di un vettore contenente il grado di partenza per evitare di prendere i vertici inutili, diminuendo i casi pessimi
- Calcolo del k core massimale attraverso un vettore best e non di un k core qualsiasi
- Variazione del modello di calcolo combinatorio nella funzione “jEdgeR”
- Terminazione della funzione “cntrlEdge” con l'aggiunta del vettore touched e il controllo anche dei vertici di arrivo