Enrico Bravi 31/01/19

Matricola: 244780

RELAZIONE ALGORITMI E PROGRAMMAZIONE

**Strutture Dati**:

Per la memorizzazione del grafo ho utilizzato la struttura Grafo che contiene: numero di vertici, numero di archi, matrice delle adiacenze e tabella di simboli. La struttura ST, per la tabella di simboli, è composta da un vettore di stringhe e la dimensione della tabella (numero di stringhe contenute nel vettore).

**Strategie risolutive**:

Il caricamento del grafo avviene attraverso più letture del primo file. La prima permette il conteggio degli archi in modo da poter inizializzare la tabella di simboli (sovradimensionandola). La seconda serve per il caricamento dei vertici nella tabella di simboli, in modo da poter determinare il numero di vertici, per poter allocare la matrice delle adiacenze. L’ultima lettura permette di aggiornare la matrice delle adiacenze.

La verifica del kernel salvato sul secondo file avviene attraverso una prima lettura del file per dimensionare il vettore (di indici) che conterà la soluzione, con la seconda lettura si salva la soluzione. Una volta salvata la soluzione si chiama una funzione che deve verificare se è effettivamente un kernel valido. La funzione di check del kernel effettua una prima verifica sulla non adiacenza tra i nodi del kernel e (non presente nel compito) una seconda verifica che per ogni vertice non appartenente al nucleo ci sia un arco uscente da un nodo del kernel e entrante nel vertice in questione.

L’identificazione di un kernel massimo per il grafo viene fatta attraverso una funzione per il calcolo del powerset dei vertici attraverso le combinazioni semplici di cardinalità decrescente. In questo modo appena viene trovata un insieme che risulta essere un kernel si può abbandonare l’esplorazione dello spazio delle soluzioni. L’abbandono dell’esplorazione viene fatto attraverso un flag, passato per riferimento alla funzione ricorsiva powerset\_R, dove, nel caso venga trovata una soluzione, viene settata opportunamente per interrompere le chiamate ricorsive e il ciclo che determina la cardinalità degli insiemi. La verifica delle soluzioni trovate viene fatta attraverso la funzione richiamata per l’insieme letto dal secondo file.

Il calcolo della lunghezza del cammino semplice che attraversa il maggior numero di vertici del kernel avviene tramite una funzione ricorsiva per il calcolo dei cammini semplici. Un ciclo esterno seleziona un vertice di partenza e la funzione ricorsiva calcola tutti i cammini semplici con partenza dal vertice selezionato. La funzione ricorsiva procede scansionando la matrice delle adiacenze selezionando i vertici raggiungibili dal vertice considerato, verificando che il vertice non sia già stato visitato e ricorrendo su di esso. Per ogni cammino vengono registrati il numero di vertici in comune con il kernel e la dimensione, in modo da poter verificare, al termine di ogni cammino, se la soluzione migliora quella già salvata oppure no.

**Differenze tra compito cartaceo e codice allegato**:

In Grafo.c:

In carica\_Grafo id1 e id2 sono dichiarati come vettori di puntatori a caratteri invece che come vettori di caratteri. La funzione Test\_Kernel\_A è scritta sotto la funzione Test\_Kernel. Le funzioni print\_sol e powerset\_R sono scritte sotto la funzione Max\_Kernel. Nella funzione Test\_Kernel la variabile v è dichiarata come intero invece che come puntatore a intero. Nella funzione cammini\_sempliciR manca la dichiarazione delle variabili i e j. Nella funzione cammini semplici manca la dichiarazione di mark e k\_dim nella chiamata della funzione cammini\_sempliciR.

Nella funzione carica\_Grafo manca il ciclo per determinare il numero di vertici, in cui vengono caricati i nodi nella tabella di simboli. Nella funzione Test\_Kernel\_A manca la verifica del secondo punto della definizione di kernel. Nella funzione powerset\_R manca la verifica del flag prima della chiamata ricorsiva. Nel calcolo dei cammini semplici, nella funzione cammini\_semplici manca la variabile per il salvataggio della lunghezza del cammino con il massimo numero di vertici in comune con il nucleo. Nella funzione cammini\_sempliciR manca la scansione della matrice delle adiacenze e l’aggiornamento del vettore mark per segnare il vertice come visitato e il backtrack una volta calcolati tutti i cammini semplici da quel vertice. Il controllo della presenza del vertice di riferimento nel nucleo è errato in quanto il controllo viene fatti su i e non su v. Manca il controllo alla fine di ogni cammino per l’aggiornamento della lunghezza del cammino soluzione.

In ST.c:

Nella funzione ST\_insert dichiarazione della variabile j all’interno del for al posto dell’utilizzo della variabile i.

Manca la funzione ST\_size per poter conoscere il numero di vertici e allocare la matrice delle adiacenze nella funzione carica\_Grafo.

**Nota**:

Nel codice allegato è presente il codice del compito cartaceo, commentato.