RELAZIONE – LUCA MAZZUCCO S203372_230215
PROGRAMMAZIONE 18 PUNTI

Indice:

1. STRUTTURE DATI

Nello sviluppo di questo programma sono state adottate le seguenti strutture dati:

1 – struct graph per memorizzare numero vertici, archi, vettore nomi dei vertici e matrice delle adiacenze;
2 – struct Lista per memorizzare i nomi dei vertici alla prima lettura del file;
3 – vettori path1, path2 e commons per la memorizzazione dei cammini e dei vertici in comune;

```
typedef struct graph{
   int v;
   int e;
   char **name;
   int **adj;
}*G;
```

```
struct Lista{
      char name[21];
      struct Lista *next;
   }*lista;
```

2.COMPOSIZIONE DEL PROGRAMMA

Il programma è stato suddiviso nelle seguenti funzioni:

```
void listinsert(struct Lista *lista, char *str);
int listlength(struct Lista *lista);
char *listgetname(struct Lista *lista, int i);
int getname(G grafo, char* str);

void path(G grafo, int x, int y, int k, int p, int *mark, int *sol, int totweight, int pos);

void stampasol(int *sol, G grafo, int n);
int num(G grafo, int *mark);
int sum(G grafo, int *mark);

int main(int argc, char* argv[]);
```

MAIN:

- Prima lettura del file contenente gli archi al fine di identificare e memorizzare il numero e nome di vertici diversi in una lista mediante l'utilizzo di funzioni dedicate quali listinsert e listlength;
- Allocazione della struttura graph e relativa inizializzazione della matrice delle adiacenze;
- Caricamento dei nomi dei vertici nel vettore name della struttura graph mediante la funzione dedicata listgetname;
- Seconda lettura del file per il caricamento delle adiacenze mediante l'utilizzo della funzione getname che ritorna l'indice del nome vertice corrente;

- Apertura secondo file, lettura e caricamento cammini nei vettori path1 e path2 appena allocati;
- Ricerca dei vertici in comune confrontando ad uno ad uno i vertici del primo cammino con tutti quelli del secondo cammino. I vertici in comune saranno copiati sul vettore commons.
- Spezzettamento cammini in sottocammini: Leggendo un cammino per volta, per ogni vertice si controlla l'eventuale intersezione con il confronto di tutti i vertici salvati in commons segnandola in un flag. Successivamente in base alla condizione sul flag si genera un a capo e un incremento del numero dei sottocammini;
- Richiesta all'utente dei valori per lo sviluppo dell'ultimo punto;
- Dichiarazione, allocazione, iniziallizzazione variabili utili alla funzione ricorsiva;
- Chiamata della funzione ricorsiva path per lo sviluppo dell'ultimo punto;
- Stampa soluzione mediante funzione stampasol;

PATH:
Path è la funzione ricorsiva che genera tutti i cammini possibili dal vertice x a y. Presenta i seguenti attributi:

- x = vertice di partenza corrente;
- y = vertice di arrivo;
- grafo = struttura del grafo con tutti i relativi dati;
- k = numero vertici riattraversabili;
- p = numero riattraversamenti possibili;
- mark = vettore marcatori dei vertici;
- sol = soluzione corrente;
- totweigth = peso corrente del cammino;
- pos = posizione corrente per il vettore soluzione.

La funzione viene richiamata sotto opportune condizioni:
la somma degli attraversamenti per ogni vertice -1 (vedere funzione sum) deve essere minore di p;
il numero dei vertici con mark>1 (vedere funzione num) deve essere minore di k;
La funzione salva la soluzione nel vettore bestsol solo se x==y e il peso corrente è maggiore del massimo.

LISTINSERT: Scandaglia la lista e tramite un flag segna l'eventuale esistenza del vertice nella lista. Se il flag resta = 0 alloca una nuova struttura lista e la accoda.

LISTLENGTH: Scandaglia la lista e ritorna la lunghezza;

LISTGETNAME: Ritorna il nome di un vertice in una posizione richiesta;

GETNAME: Ritorna l'indice di un vertice scandagliando il vettore name della struct graph confrontando le stringhe;

STAMPASOL: Stampa a video il vettore soluzione migliore bestsol;

NUM: Ritorna il numero di vettori attraversati più di una volta;

SUM: Ritorna il numero di riattraversamenti;

3. MODIFICHE APPORTATE RISPETTO ALLO SVILUPPO IN SEDE D'ESAME

- Correzzioni varie di virgolette, parentesi, omissioni dovute alla fretta;
- Aggiunto vettore bestsol per il salvataggio della soluzione migliore, la sovvrascrizione su sol generava errori e soluzioni errate;
- Corrette le funzioni sum e num che calcolavano in base agli attraversamenti e non riattraversamenti;
- Implementate le funzioni di lista che mancavano dalla versione d'esame;
- Il marcamento e smarcamento dei vertici attraversati nella funzione path è stato spostato all'esterno di una delle condizioni interne al ciclo di chiamata per poter tenere meglio conto nella condizione interna del marcamento corrente;
- E' stato precaricato nel vettore sol in posizione 0 il vertice di partenza che veniva omesso;
- La struct Lista è stata spostata dal main a globale per la visibilità necessaria per le altre funzioni.