RELAZIONE ALGORITMI E PROGRAMMAZIONE

NOME: SALVATORE

COGNOME: MALLEMACI

MATRICOLA: S261358

ESERCIZIO SCELTO PER LA VALUTAZIONE: LAB12 ES 01

STRUTTURE DATI:

- Utilizzo un ADT di prima classe per la tabella di simboli (standard), al fine di poter "tradurre" i nomi dei singoli vertici costituenti il grafo, sfruttando un vettore di stringhe, in modo da identificarli tramite interi per identificarli ai fini dell'algoritmo e come stringhe per identificarli ai fini dell'utente. Uso dunque la STsearchByIndex, che dato l'indice, ritorna la chiave. La tabella di simboli è esterna al grafo, in modo tale da poter essere utilizzata sia dal main che dall'ADT grafo.

- Utilizzo un ADT di prima classe per la costruzione del singolo grafo, (che successivamente sarà manipolato al fine di renderlo un DAG). Per la gestione degli archi, ho preferito implementare una matrice delle adiacenze. Seguono le funzioni standard, con aggiunta di funzioni extra, implementate in base alle richieste del programma.

SCELTE ALGORITMICHE:

- Al fine di ottenere un DAG, la prima cosa da fare è verificare se ci siano dei cicli nel grafo. Segue dunque la GRAPHdfsSearchCycle, tramite una ricerca in profondità, individuando archi Tree e Back. Trovato un arco Back, si trova un ciclo. In questo caso, segue la generaR, dove si adotta un approccio PowerSet di tipo Combinazioni Semplici, generando gli insiemi di cardinalità minima la cui rimozione porta ad un DAG. Se vi è la presenza di un ciclo dunque, al vettore di interi "search" viene assegnato 0, e si procede alla somma dei pesi di ciascun vertice di tale natura, confrontandone la valenza ottimale con la bestWt, aggiornando ove necessario la bestSol.
  Per rimuovere determinati archi, utilizzo le funzioni standard viste nei grafi.
- Per generare i DAG utilizzo la generaGraphToDag, rimuovendo gli archi back che mi imponevano la ciclicità del grafo, richiamando dunque la generaR.
- Per il calcolo delle distanze massime da ogni nodo sorgente verso ogni nodo del DAG costruito al passo precedente, seguo un ragionamento relativo alla ricerca dei cammini massimi su DAG pesati. L'assenza di archi B permette dunque, effettuata la ricerca in profondità, un ordinamento topologico (TSdfsR presente nel modulo Graph). Seguendo tale ordinamento, è possibile applicare la Relaxation <invertita>, che segue il seguente approccio:

```
if(d[v]<d[u]+w(u,v)){
  d[v]=d[u]+w(u,v);
}
```

COMMENTI EXTRA:

- Sfrutto la linea di comando semplicemente per inserire più nomi di file, essendo quattro quelli a disposizione. Utilizzo dunque un vettore di grafi, gestendo per ogni casella del vettore un singolo grafo.