Marco Perno S294424

LAB09_E01:

Per la risoluzione del problema si è ritenuto opportuno creare un grafo con rappresentazione mediante lista delle adiacenze e una tabella di simboli. Qui di seguito vengono riportate le strutture dati utilizzate:

```
typedef struct node *link;
struct node { int v; int wt; link next; } ;
struct edge { int v; int w; int wt; };
struct graph_s {int V; int E; link *ladj; ST tab; Edge * edge_array;
int DIM_edge_array; link z;} ;
```

```
typedef struct st_s* ST;
typedef struct {
    char name[31];
} Item;
struct st_s {
    Item * vet;
    int N;
};
```

Il motivo della scelta della lista di adiacenze è perchè si è ritenuto più opportuno questo tipo di struttura per problemi quali la DFS e l'implementazione dell'algoritmo di Bellman-Ford per i requisiti richiesti dal problema.
Inoltre è stato deciso di aggiungere un campo alla struttura grafo chiamato `Edge * edge_array;` nel quale sono stati salvati tutti gli archi presenti nel grafo.
Il motivo è abbastanza semplice. Infatti, per risolvere il requisito 1 si è scelto di generare un powerset di disposizioni ripetute di 0 e 1 di dimensione numero degli archi dove 1 sta per "rimuovi arco". In questo modo, facendo ogni volta una copia del grafo e scartando gli archi marcati come 1, si verificano tutte le possibilità di grafi senza certi archi alla ricerca dell'insieme di archi rimossi di cardinalità minima che renda il grafo generato un dag. Per determinare se il grafo generato è un dag si è applicata la DFS con uscita anticipata nel caso venisse trovato un arco di tipo back.
Per il punto 2 è stato solamente necessario fare la somma dei pesi di ogni insieme di archi ritornati dal punto precedente per determinare l'insieme a peso massimo.
Infine per il punto 3, dopo aver creato il dag al punto 2, è stato applicato l'algoritmo di Bellman Ford con le condizioni di relaxition invertite per ogni vertice del dag.