Relazione sul Terzo Esercizio

Ciò che si chiedeva in questo esercizio era di scegliere se implementare grafi orientati o meno. La differenza tra queste due scelte è notevole in quanto scegliendo i grafi NON orientati, si possono risolvere gli stessi problema evitando un discreto carico di lavoro per l'implementazione di alcuni algoritmi quali:

- **Tralazione del grafo**: in un grafo non orientato il rispettivo grafo traslato è identico all' originale per cui non si rende necessaria la traslazione;
- **Il Topological Sort**: il metodo di scoperta delle parti fortemente connesse non lo richiede;
- **Il calcolo dell CFC**: si può effettuare mediante una semplice DFS.

Nonostante questa premessa ho deciso di implementare grafi Orientati perché in fase di lettura da file non mi piaceva l'idea di "creare archi virtuali" che non fossero realmente scritti su file. Comunque il mio codice supporta entrambi i tipi di grafi, e può fornire entrambi i risultati corretti mediante l'utilizzo di una variabile usata come switch.

Inoltre per quanto riguarda l'algoritmo di ricerca di cammini minimi ho scelto di implementare una **Coda di Priorità Heap** basata sull' algoritmo di sorting visto nell' esercizio 1.

Limiti: il mio algoritmo di creazione del grafo non è ottimo. Per rappresentare un grafo ho scelto una struttura semplice: array di liste. In questo modo la ricerca di un vertice costa tempo O(n), e quindi la creazione avrà un costo complessivo di $O(n^2)$. Per ottenere una ricerca in tempo O(logn) si sarebbe potuto utilizzare una Tabella Hash, ma ho preferito evitarle in quanto volevo riutilizzare gli alberi rosso-neri implementati nell' esercizio 2. L' idea era di creare un dizionario parallelo all' array di vertici e quindi fare l'inserimento duplice di ogni vertice: uno nell' array e uno nell' albero, il che a conti fatti sarebbe stato comunque un enorme guadagno di tempo. Per mancanza di tempo però, non ho avuto modo di implementarli, anche se il codice è strutturato per essere agevolmente aggiornato.
Tempistiche: Qui di seguito sono riportate le varie tempistiche degli algoritmi:

(+) Loading data from file: Completed in 0.03241 seconds

(+) Create Graph: Completed in 28.77578 seconds

(+) Finding Strongly Connected Components:
There are 3 connected components:
This cfc have 2 elements
This cfc have 2 elements
This cfc have 18636 elements

(-)Completed in 0.01913 seconds

(+) Searching min path: Completed in 1.50386 seconds

(+)Minimum Distance: 1207.68 km

(#) Unit_Test Graph: Successfully Performed.

NB: i risultati proposti sono quelli che si hanno con un grafo NON orientato.