Obiettivo
L'obiettivo di questo elaborato è quello di risolvere, con una buona approssimazione e in un tempo ragionevole, il problema del commesso viaggiatore attraverso l'utilizzo dell'algoritmo di ricerca A*, facendo uso dell'euristica basata sul Minimum Spanning Tree.

Descrizione del problema
Dato un insieme di città disposte in una griglia di dimensione unitaria, il problema consiste nel trovare il cammino di costo minimo che ci permette di visitare tutte le città una sola volta e di tornare alla città di partenza.

Formalizzazione del problema
Si formalizza il problema rappresentando le città come coppie di interi, si genera il grafo completo calcolando la distanza euclidea tra le città e lo si memorizza con una matrice di adiacenza. Questo grafo è da non confondere con il grafo su cui poi si muoverà l'algoritmo di ricerca A*.
Uno stato è composto da un insieme di città visitate e dalla città attuale. Si definisce lo stato iniziale del problema come lo stato in cui l'insieme delle città visitate è composto solo dalla città di partenza e la città attuale è quella di partenza. Si definisce lo stato goal del problema come lo stato in cui l'insieme delle città visitate corrisponde con l'intero insieme delle città presenti e la città attuale è quella di partenza.
Un nodo del grafo su cui si muove l'algoritmo di ricerca è composto da uno stato, da un riferimento al nodo padre e dal costo del cammino rappresentato dal quel nodo.

Sviluppo della soluzione
L'algoritmo di ricerca A* utilizza principalmente due strutture dati. La prima è una coda con priorità in cui vengono messi i nodi di frontiera, cioè quei nodi non ancora esaminati che sono il risultato dell'espansione di nodi già visitati. La seconda è un dizionario, che ha come chiavi gli stati e come valori i nodi, che ci dà indicazioni sugli stati che abbiamo già raggiunto e con quale costo li abbiamo raggiunti. La priorità con cui vengono presi ed esaminati i nodi dalla frontiera è la somma del costo del cammino rappresentato da quel nodo e dal valore assunto dalla funzione euristica in quel nodo.
La funzione euristica trova il minimum spanning tree che collega tutte le città non ancora visitate e ne calcola il costo, sommandoci poi il costo minimo che serve per andare dall'insieme delle città non visitate alla città iniziale. Questa euristica è ammissibile poiché il minimum spanning tree è il collegamento di costo minimo tra le città, tale costo viene poi sommato al costo minimo per tornare alla città iniziale.

Conclusioni
Si è eseguito l'algoritmo su problemi con città generate randomicamente ottenendo i risultati riportati nella tabella (1) per la velocità e nella tabella (2) per l'ottimalità. (le soluzioni esatte per il confronto sono state ottenute con la libreria python-tsp https://pypi.org/project/python-tsp/.)

(1)

Number of cities	map size	Average Time(20 random runs)
4	10x10	0,000
5	10x10	0,000
6	10x10	0,001
7	10x10	0,002
8	10x10	0,004
9	10x10	0,016
10	10x10	0,034
11	10x10	0,052
12	10x10	0,220
13	10x10	0,377
14	10x10	0,314
15	10x10	1,551
16	10x10	2,699
17	10x10	7,086
18	10x10	11,141
19	10x10	28,542
20	10x10	42,671

(2)

Number of cities	Percentage error(10 random runs)
5	2,20%
6	1,90%
7	2,10%
8	2,60%
9	2,40%
10	2,50%
11	2,30%
12	2,40%
13	2,50%
14	2,60%
15	2,90%