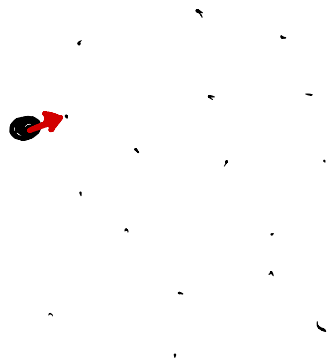
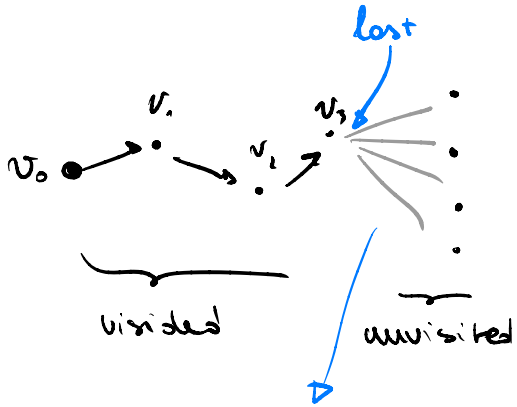
Greedy (Nearest Neighbors)



estendo il percorso sempre con il comune più vicino

All'iterazione generica:

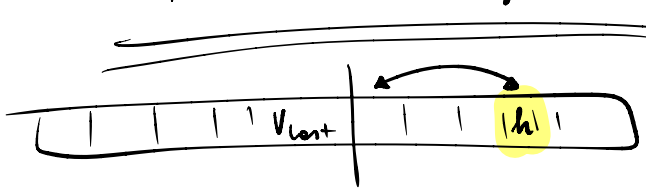


choose the smallest

$$(v,h) = \text{argmin} \{ (i,j) \mid i = last,\ j \in \text{UNVISITED} \}$$

Alla fine collego l'ultimo nodo con il primo

TODO: implementare `tsp_greedy(..)`



$\Rightarrow O(n^2)$

lineare nel numero di archi

Scelgo nodo iniziale iterativamente

⇓

se ho tempo avanzato

salvo il migliore (lista e costo)

(↳ e tempo in cui l'ho trovato)

Se verbose faccio il plot di tutte

Faccio un check di correttezza nell'update

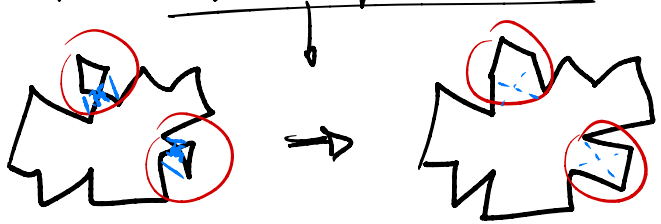
↳ • nella lista non ci devono essere duplicati

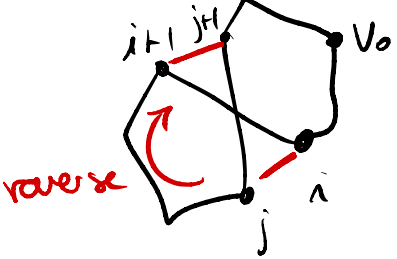
• ricalcolo il costo della soluzione

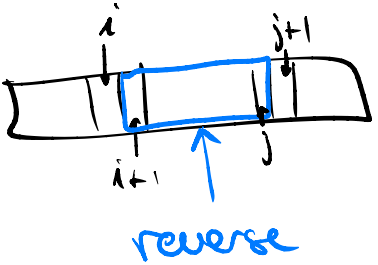
NON FACCIO DOUBLE == DOUBLE !!!

uso Epsylon $\rightarrow 10^{-8}$

Qui vale però togliere gli intrecci nella soluzione







if $c_{i,i+1} + c_{j,j+1} > c_{ij} + c_{i+1,j+1} \Rightarrow$ reverse

↓

NON SERVE TROVARE L'INTRECCIO

$O(n^2)$

- Trovo tutte le coppie e faccio solo lo scambio più favorevole
- Implemento il primo scambio

⇓

Ripeto finché non trovo più miglioramenti

Soluzioni 3% distanti dall'ottimale


MARCO
COSTI