

# Algoritmi euristici

Un algoritmo  $\mathcal{A}$  si dice *euristico* per un problema  $P$  se restituisce una soluzione ammissibile  $z^{\mathcal{A}}$  che non è garantito essere la soluzione ottima.

Sia  $P$  problema di minimizzazione. Un algoritmo euristico si dice  $\delta$ -*approssimato* se

1. Ha complessità polinomiale
2. Per ogni istanza  $I$  di  $P$  con soluzione ottima  $z^*(I)$ , si ha

$$z^{\mathcal{A}}(I) / z^*(I) \leq \delta \quad (1)$$

**Osservazione:** se  $P$  è problema di massimo  $\delta \leq 1$  e (1) vale con il segno di  $\geq$

# Euristiche per il TSP

$G = (V, E)$  grafo completo

$c_{UV}$  costo dell'arco  $UV \in E$

**Euristiche costruttive:** tentano di costruire un "buon" ciclo hamiltoniano a partire da un sottociclo eventualmente vuoto.

**Euristiche migliorative:** a partire da una soluzione ammissibile, si tenta di migliorarla attraverso miglioramenti "locali".

# Euristica *Nearest Neighbor*

Input:  $G=(V,E)$  Output: ciclo hamiltoniano  $T^{(1)}$

```
procedure nearest_neighbor ()
```

```
  Scegli un vertice  $u \in V$  qualsiasi;
```

```
   $W = V \setminus \{u\}$ , aggiungi  $u$  alla lista  $T$ 
```

```
  while  $|W| > 0$  {
```

```
    scegli  $v \in W$  tale che  $c_{uv} = \min \{c_{uj} : j \in W\}$ 
```

```
    Aggiungi  $\{v\}$  alla lista  $T$ 
```

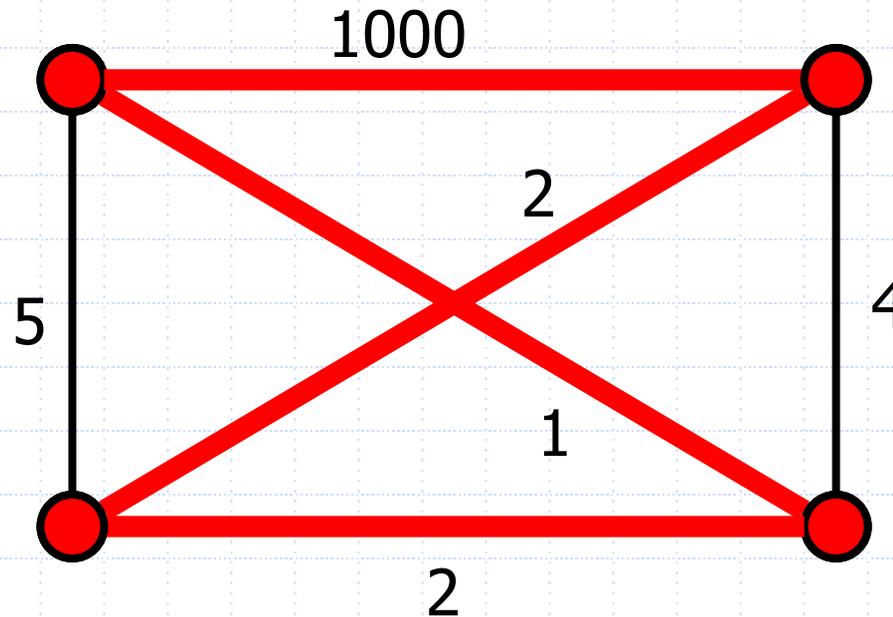
```
     $W = W \setminus \{v\}$ 
```

```
     $u = v$ 
```

```
  }
```

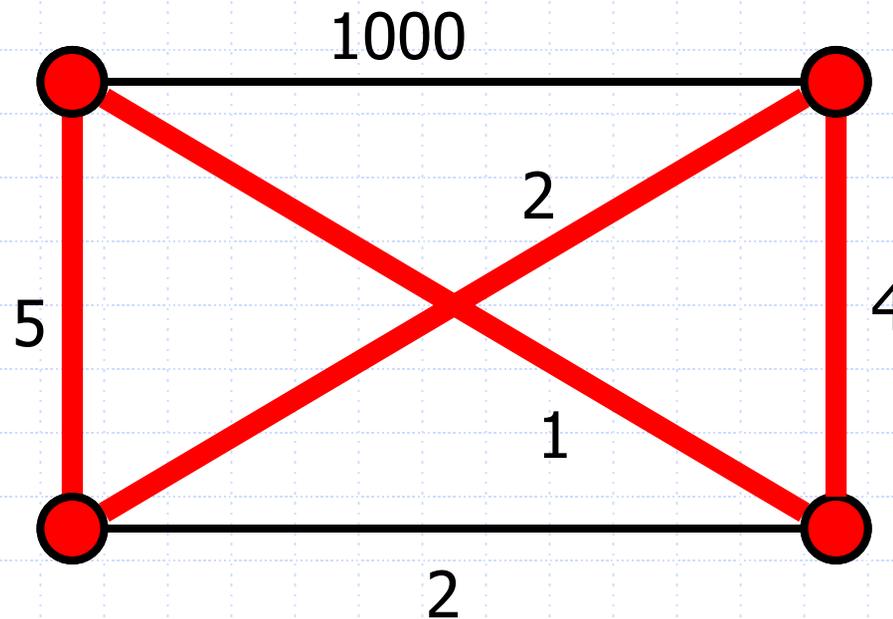
(1) un ciclo hamiltoniano è univocamente rappresentato da una permutazione dei vertici di  $G$

## Esempio *Nearest Neighbor*



Soluzione di valore 1005, è ottima?

## Esempio *Nearest Neighbor*



La soluzione ottima vale 12

**Osservazione:** il rapporto  $z^A(I)/z^*(I)$  può essere reso grande a piacere.

# Euristiche di inserimento

Input:  $G=(V,E)$  Output: ciclo hamiltoniano  $T$

```
procedure insertion_heuristic ()
```

```
  Inizializza  $T$  con un sottociclo
```

```
   $W = V / T;$ 
```

```
  while ( $|W| > 0$ ) {
```

```
    scegli un vertice  $u \in W;$ 
```

```
    scegli la posizione in cui inserire  $u$   
    in  $T;$ 
```

```
    inserisci  $u$  in  $T;$ 
```

```
    elimina  $u$  da  $W;$ 
```

```
  }
```

# Selezione del vertice da inserire

## Definizione

Si definisce distanza di un vertice  $u$  da un ciclo  $T$ , il peso del più piccolo spigolo che collega il vertice ad un altro qualsiasi vertice del ciclo

$$T = (1, 2, \dots, k)$$

$$\text{dist}(u, T) = \min \{ c_{uv} : v \in T \}$$

## Nearest Insertion

Inserisci il vertice  $u$  che minimizza  $\text{dist}(u, T)$ ,  $u \notin T$

## Farthest Insertion

Inserisci il vertice  $u$  che massimizza  $\text{dist}(u, T)$ ,  $u \notin T$

# Selezione della posizione di inserimento

## Osservazione

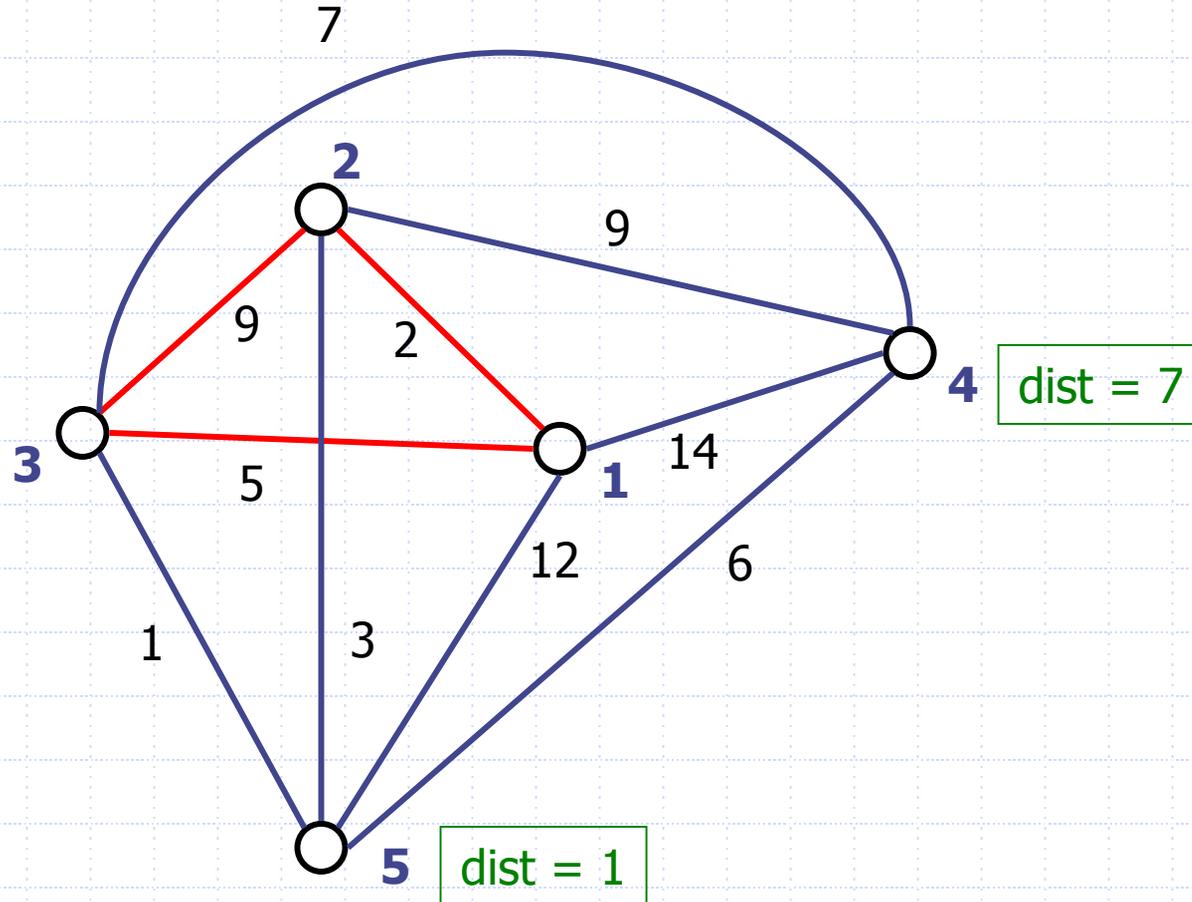
Scegliere la posizione equivale a scegliere lo spigolo da eliminare nel ciclo  $T$ . Un vertice può essere inserito in ogni posizione di  $T$ .

Sia  $T = (v_1, v_2, \dots, v_n)$ ,  $c(T)$  il suo costo e sia  $u$  lo spigolo da aggiungere al ciclo

Se  $c(T(i))$  è il costo del ciclo ottenuto da  $T$  inserendo nella posizione  $i$  il nodo  $u$ , *il costo di inserimento* è pari a  $c(T(i)) - c(T)$ .

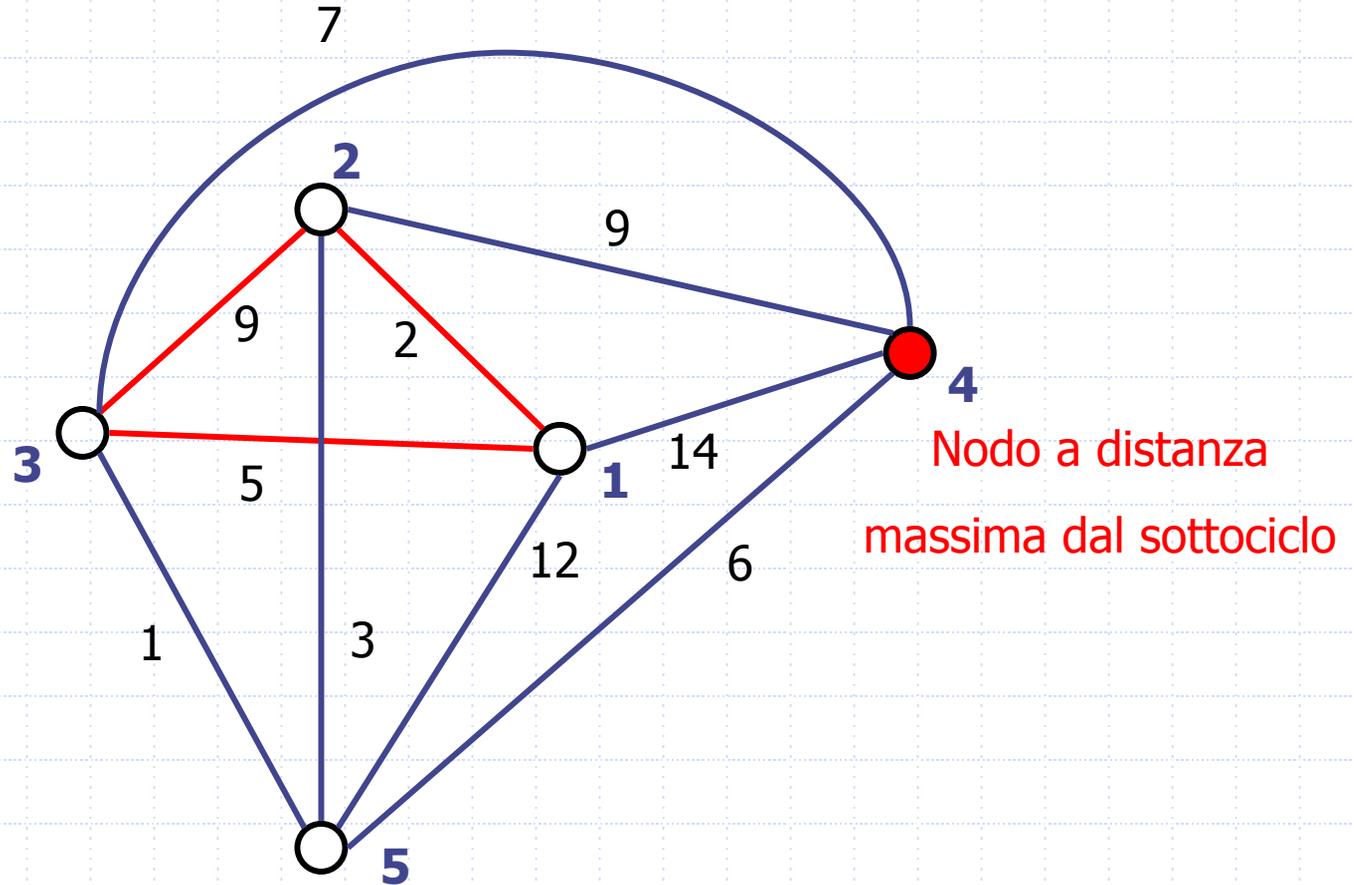
Si seleziona la posizione che minimizza  $c(T(i)) - c(T)$

# Esempio (Farthest Insertion)

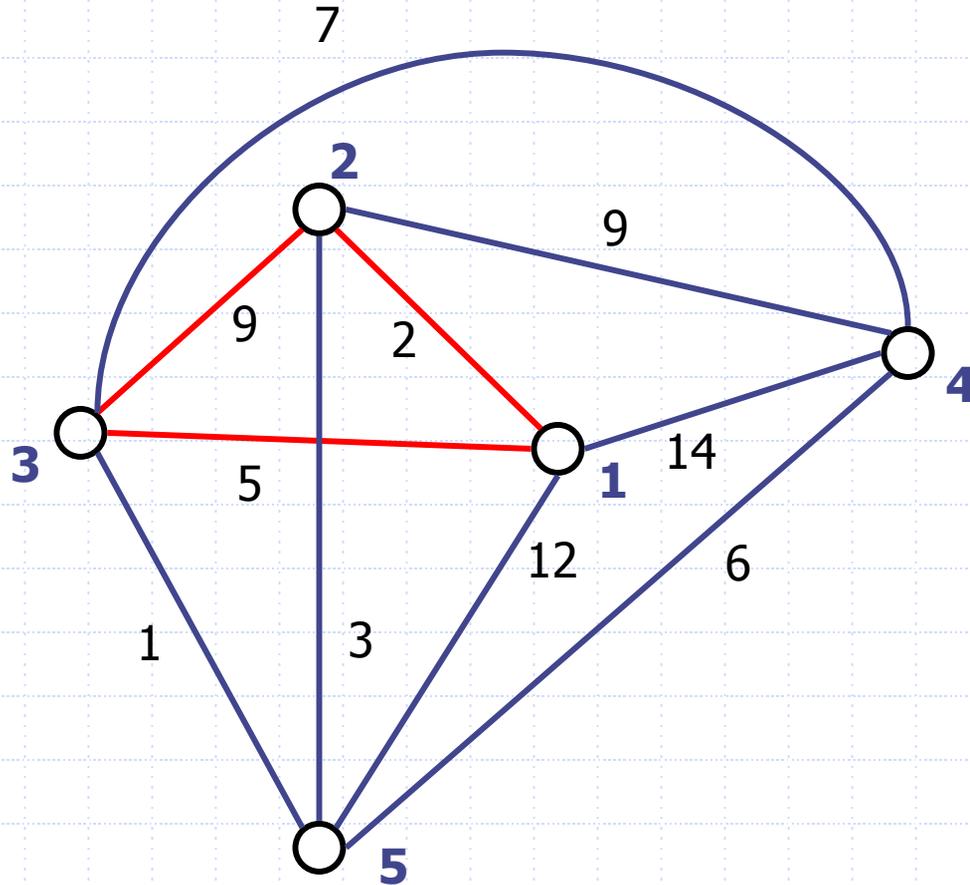


Calcolo della distanza dei nodi dal ciclo  $T = \{1, 2, 3\}$

# Esempio (Farthest Insertion)

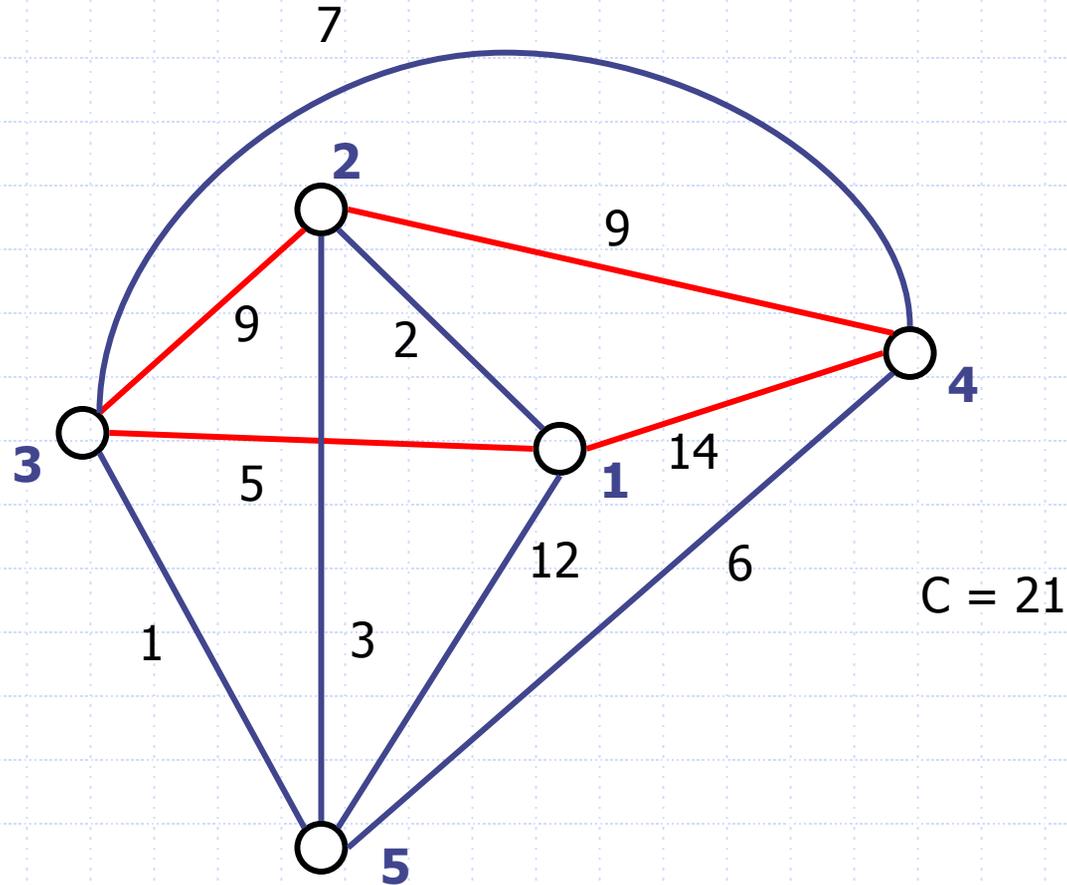


# Esempio (Farthest Insertion)



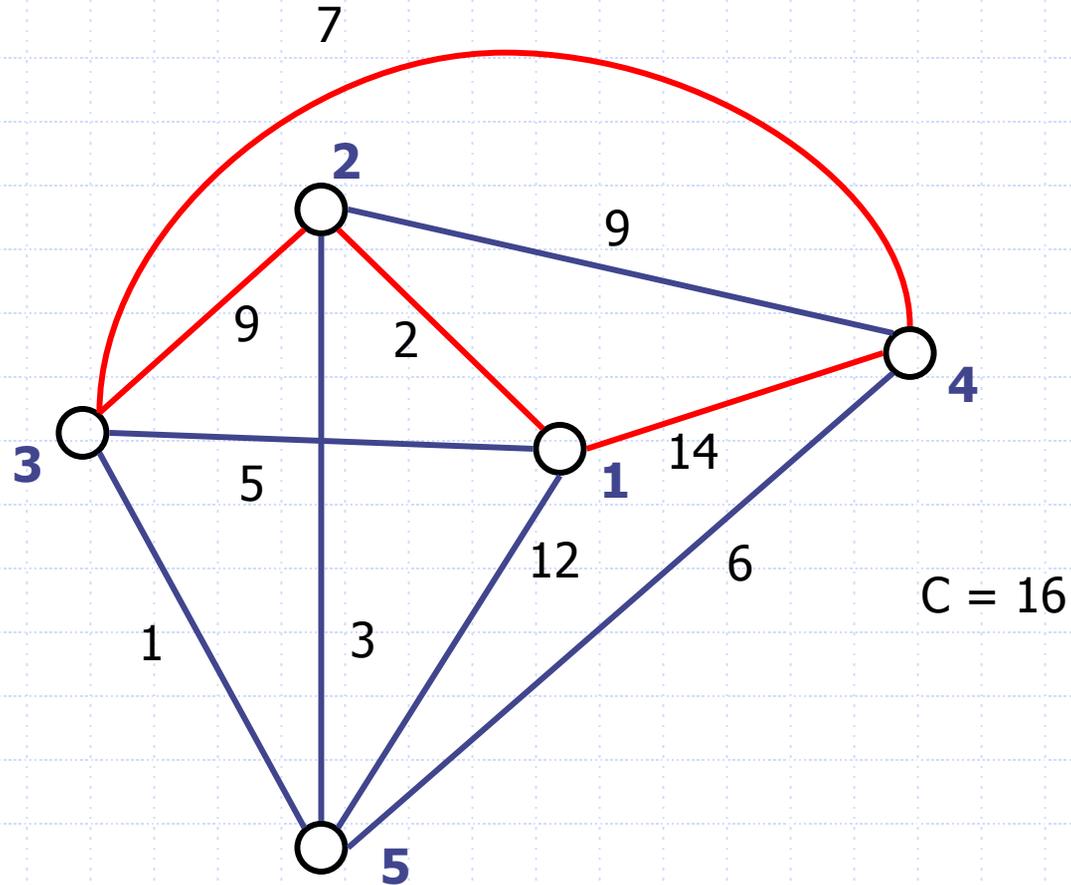
Scelta della posizione di inserimento all'interno del ciclo  $T = \{1, 2, 3\}$

# Esempio



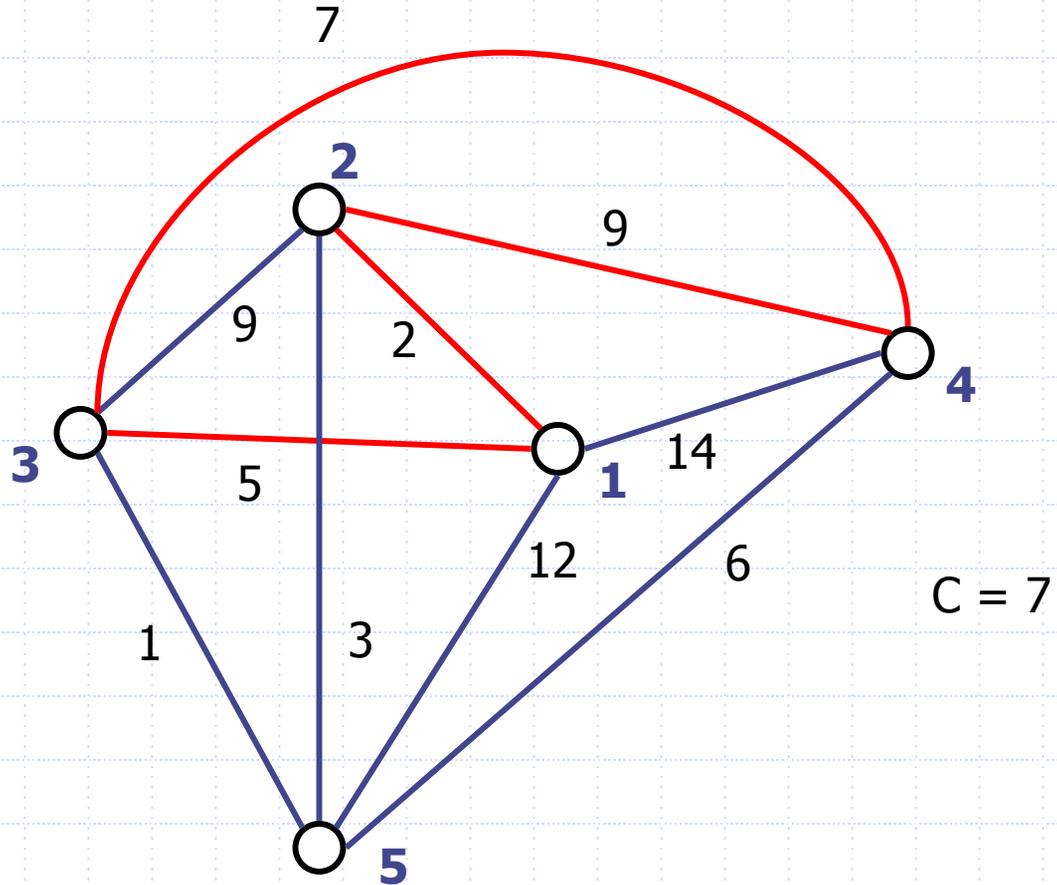
L'inserimento tra il nodo 1 e il nodo 2 costa 21

# Esempio



L'inserimento tra il nodo 1 e il nodo 3 costa 16

# Esempio



L'inserimento tra il nodo 2 e il nodo 3 costa 7, pertanto ottengo il nuovo ciclo  $T' = \{1, 2, 4, 3\}$

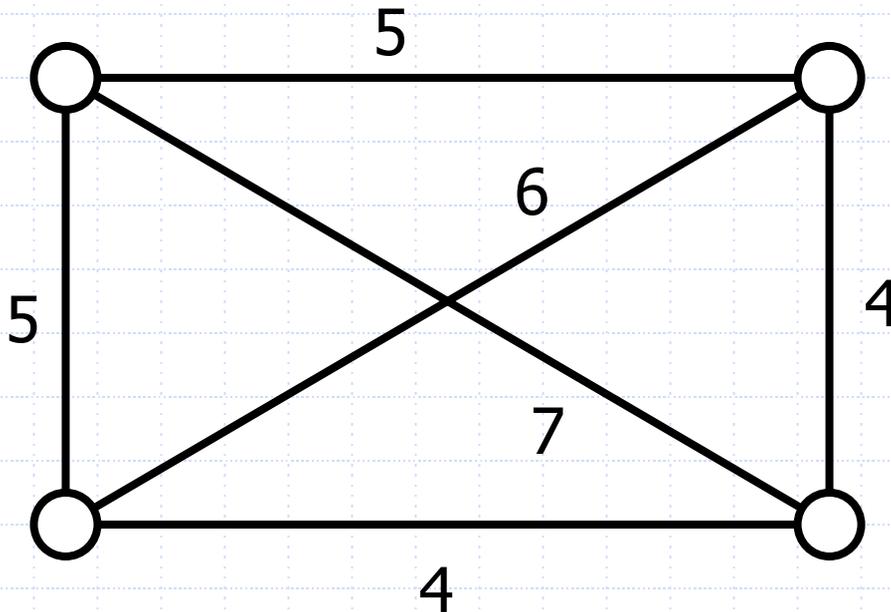
## Altre regole di selezione del vertice

**Random Insertion:** si sceglie un vertice a caso fra quelli non ancora inseriti nel ciclo

**Cheapest Insertion:** si sceglie il vertice che può essere aggiunto al ciclo con il minimo aumento di costo

## Una proprietà strutturale

Si dice che la matrice delle distanze di un grafo  $G$  soddisfa la *disuguaglianza triangolare* se comunque prendo un triangolo  $e_1, e_2, e_3$  in  $G$  si ha  $c_{ei} + c_{ej} \geq c_{ek}$  per  $i \neq j \neq k, i, j, k \in \{1, 2, 3\}$



# Richiamo

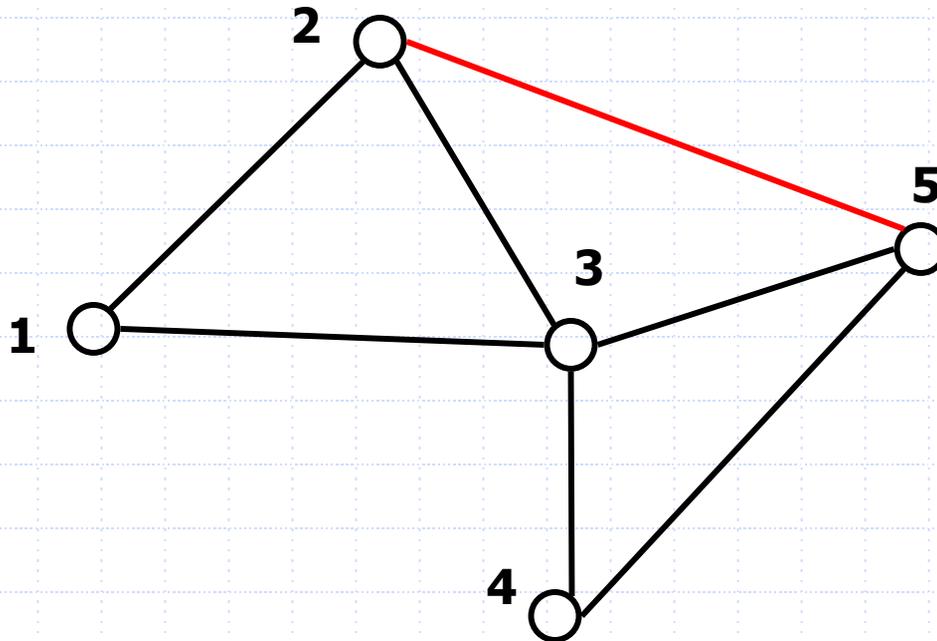
$G = (V, E)$  è un grafo euleriano se e solo se il grado di ogni nodo è pari

Se  $G = (V, E)$  è un grafo euleriano e  $v$  è un vertice di  $G$  allora è possibile costruire un percorso che inizia e finisce in  $v$  e che attraversa ogni spigolo esattamente una volta

## Teorema

Sia  $H = (V, F)$  un grafo completo con la matrice dei costi che soddisfa la disuguaglianza triangolare. Sia  $G = (V, E)$  un sottografo euleriano connesso di  $H$ .  $H$  contiene un ciclo hamiltoniano di lunghezza al più  $\sum_{e \in E} C_e$

# Dimostrazione (idea)



## Euristica *Double Tree*

1. Calcola un minimo albero ricoprente  $K$
2. Raddoppia gli spigoli di  $K$ , formando un percorso euleriano
3. Ricava un ciclo hamiltoniano dal percorso euleriano

Indichiamo con  $z_{DT}^H$  il valore della soluzione che si ottiene applicando l'euristica Double Tree

# Da un percorso euleriano ad un ciclo hamiltoniano

Consideriamo un percorso euleriano  $(v_1, \dots, v_k)$

```
procedure obtain_hamiltonian ()
```

```
   $T = \{v_1\}$ ,  $i=2$ ,  $v = v_1$ 
```

```
  while  $|T| < n$  {
```

```
    if  $v_i \notin T$ 
```

```
       $T = T \cup \{v_i\}$ 
```

```
      collega  $v$  a  $v_i$ 
```

```
       $v = v_i$ 
```

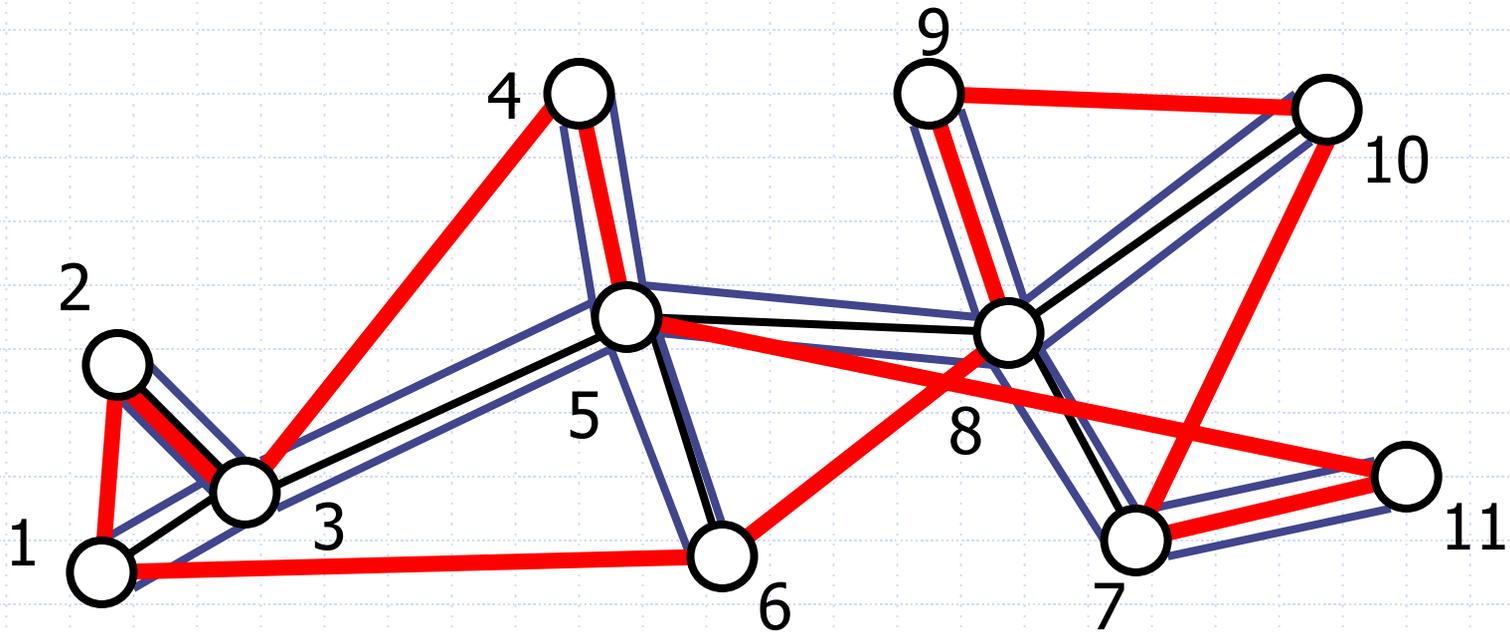
```
       $i ++$ 
```

```
  }
```

```
  collega  $v$  a  $v_1$ 
```

$T$  è un ciclo hamiltoniano

# Esempio



Tour (5,4,5,3,2,3,1,3,5,6,5,8,9,8,10,8,7,11,7,8,5)

# Proprietà

Double tree è un algoritmo 2-approssimato per il TSP

## Dimostrazione

1.  $z^* \geq z_{\text{TREE}}$  (1-albero è un rilassamento per TSP)
2. Per costruzione, la lunghezza del "doppio albero" è  $2 * z_{\text{TREE}}$
3.  $z_{\text{DT}}^H \leq 2 * z_{\text{TREE}}$

Pertanto:

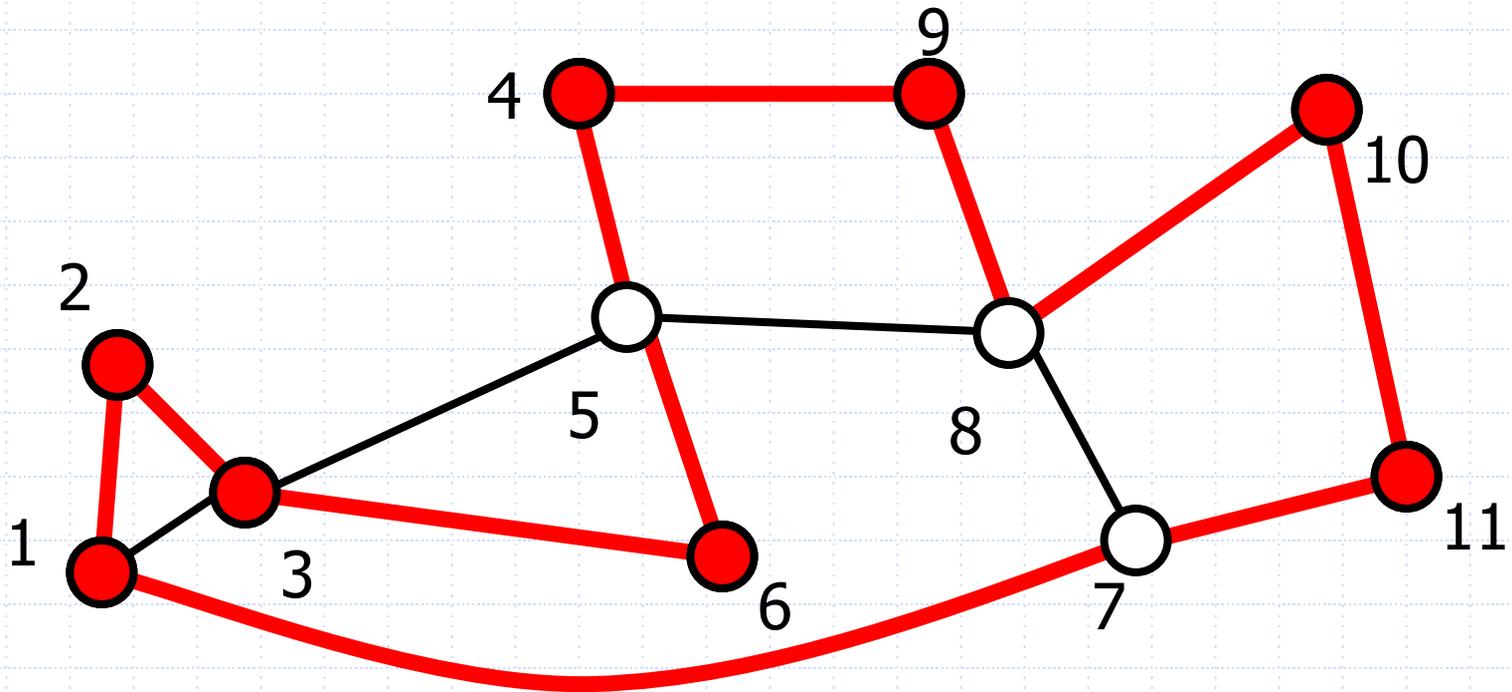
$$z_{\text{DT}}^H / z^* \leq 2 * z_{\text{TREE}} / z_{\text{TREE}} = 2$$

# Euristica di Christofides

1. Calcola un minimo albero ricoprente  $K$
2. Sia  $V' \subseteq V$  l'insieme dei vertici che hanno grado dispari in  $K$
3. Trova il matching perfetto  $M$  di peso minimo sui nodi  $V'$
4.  $M \cup K$  è un percorso euleriano
5. Ricava un ciclo hamiltoniano dal percorso euleriano

Indichiamo con  $z_{CH}^H$  il valore della soluzione che si ottiene applicando l'euristica di Christofides

# Esempio



Tour (1,2,3,6,5,4,9,8,10,11,7,8,5,3,1)

Nodi di grado dispari

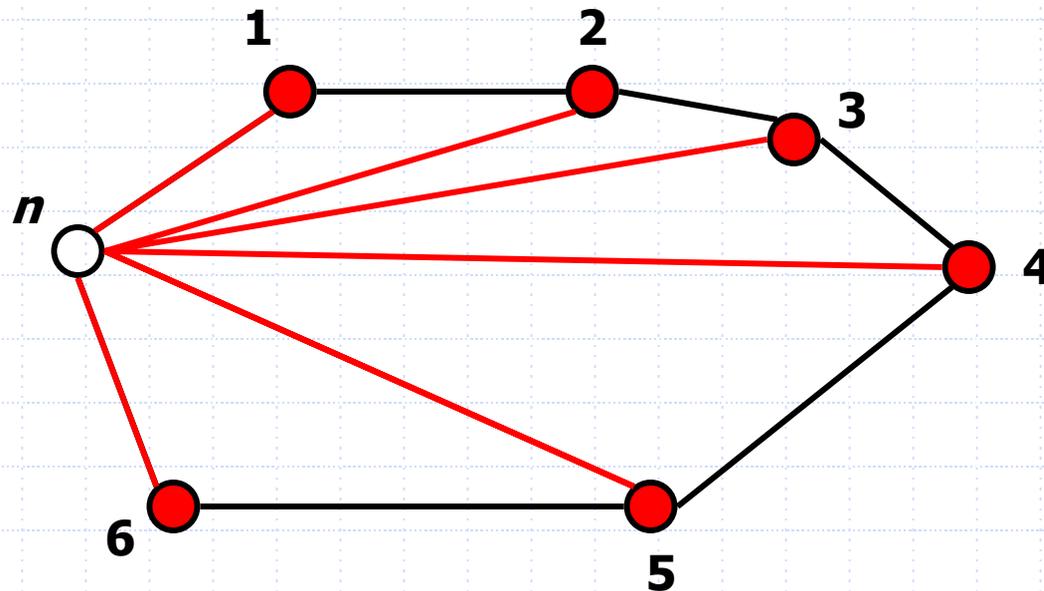
# Proprietà

Christofides è un algoritmo  $3/2$ -approssimato per il TSP

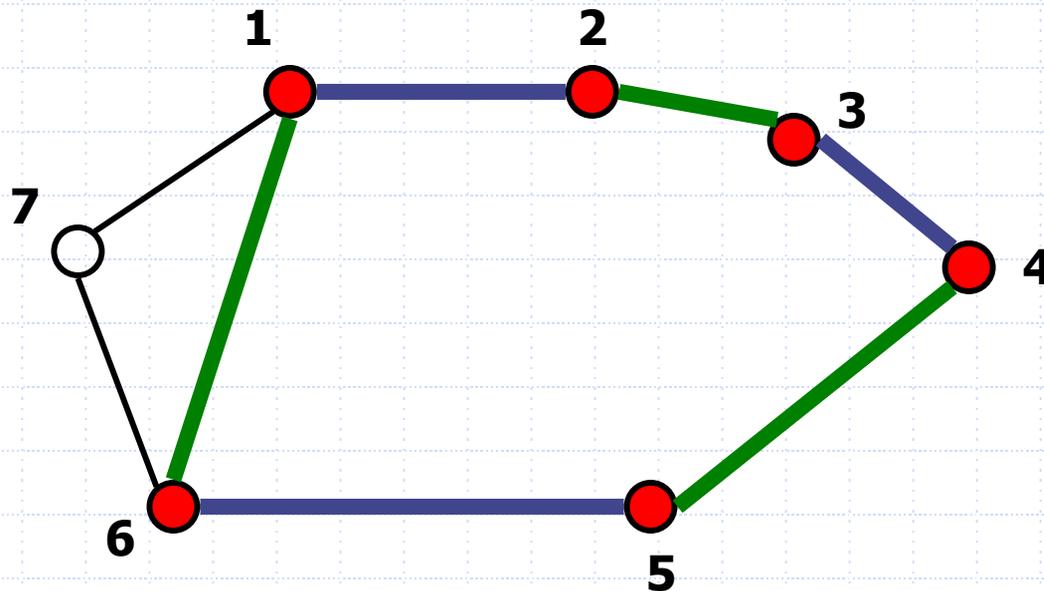
## Dimostrazione

1.  $z^* \geq z_{\text{TREE}}$  (1-albero è un rilassamento per TSP)

Consideriamo il seguente ciclo hamiltoniano (ottimo) di peso  $z^*$  e supponiamo che i nodi rossi appartengano a  $V''$ .



# Dimostrazione



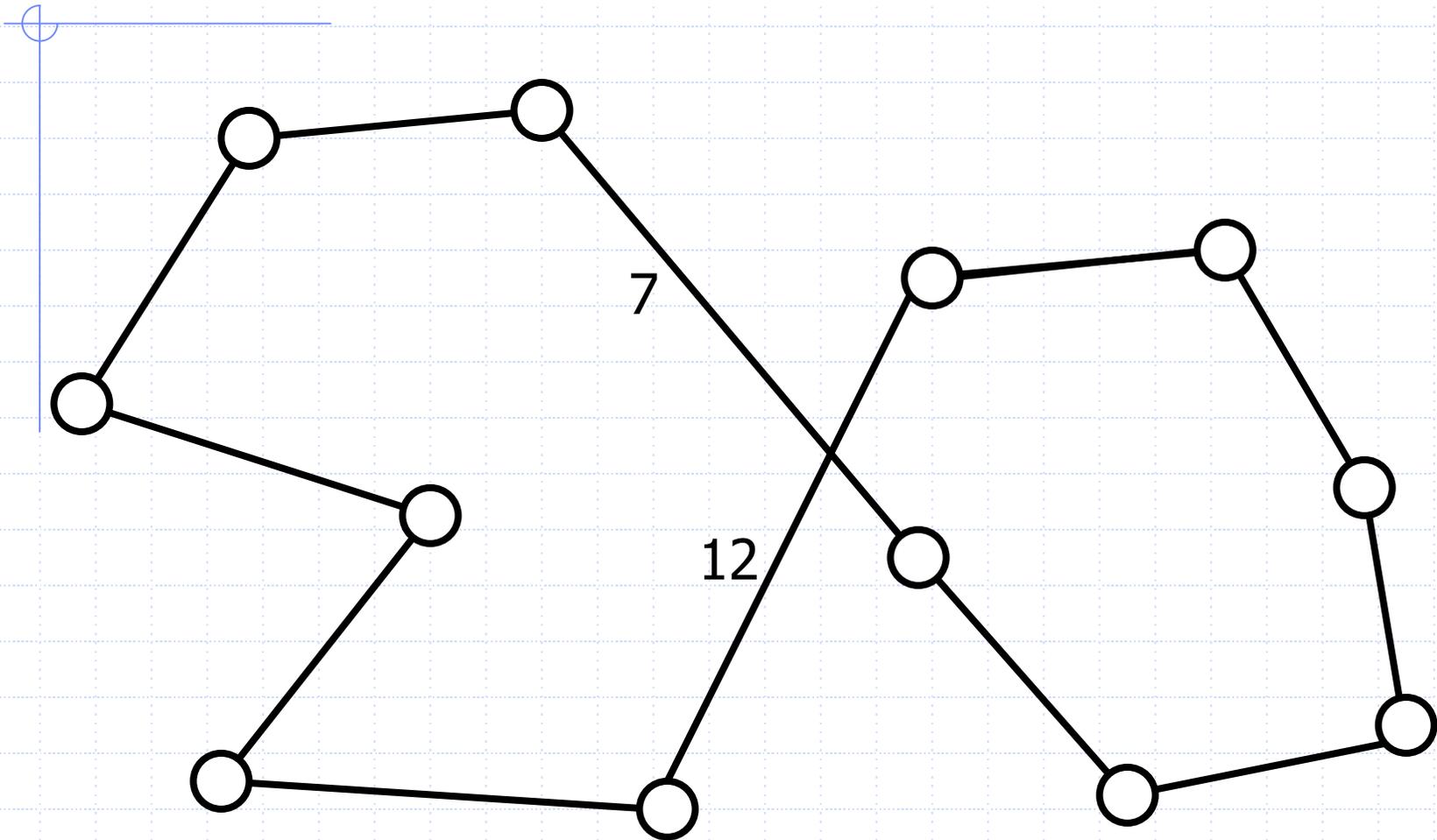
Dalla disuguaglianza triangolare si ha che

$z^* \geq z_{M\text{verde}} + z_{M\text{blu}}$ . Ma  $z_M \leq z_{M\text{verde}}$  e  $z_M \leq z_{M\text{blu}}$ , ovvero  $z^* \geq 2z_M$

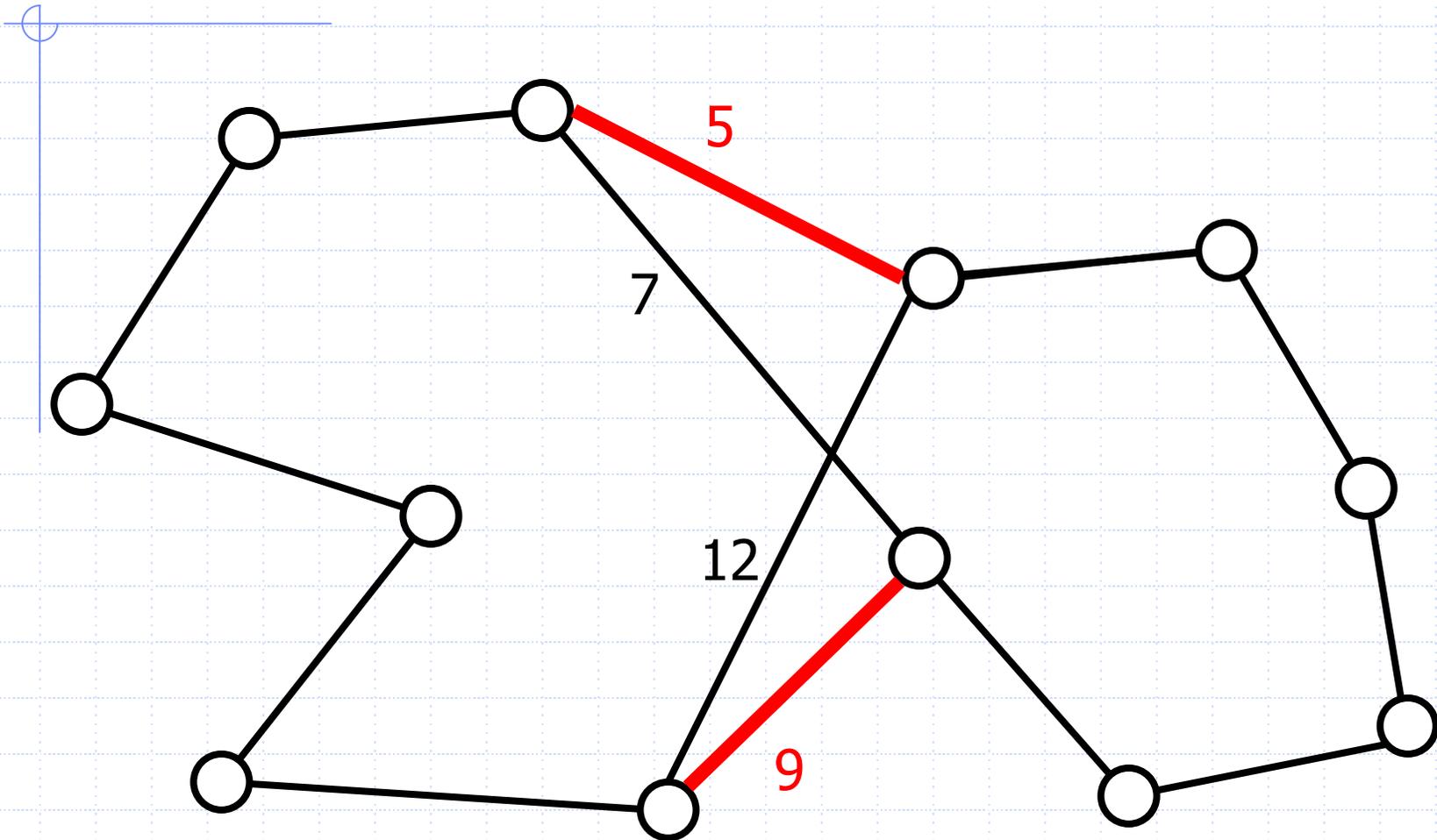
Allora:

$$z_{CH}^H \leq z_{TREE} + z_M \leq z^* + z^*/2 = 3/2 z^*$$

# 2-opt exchange (euristica di miglioramento)

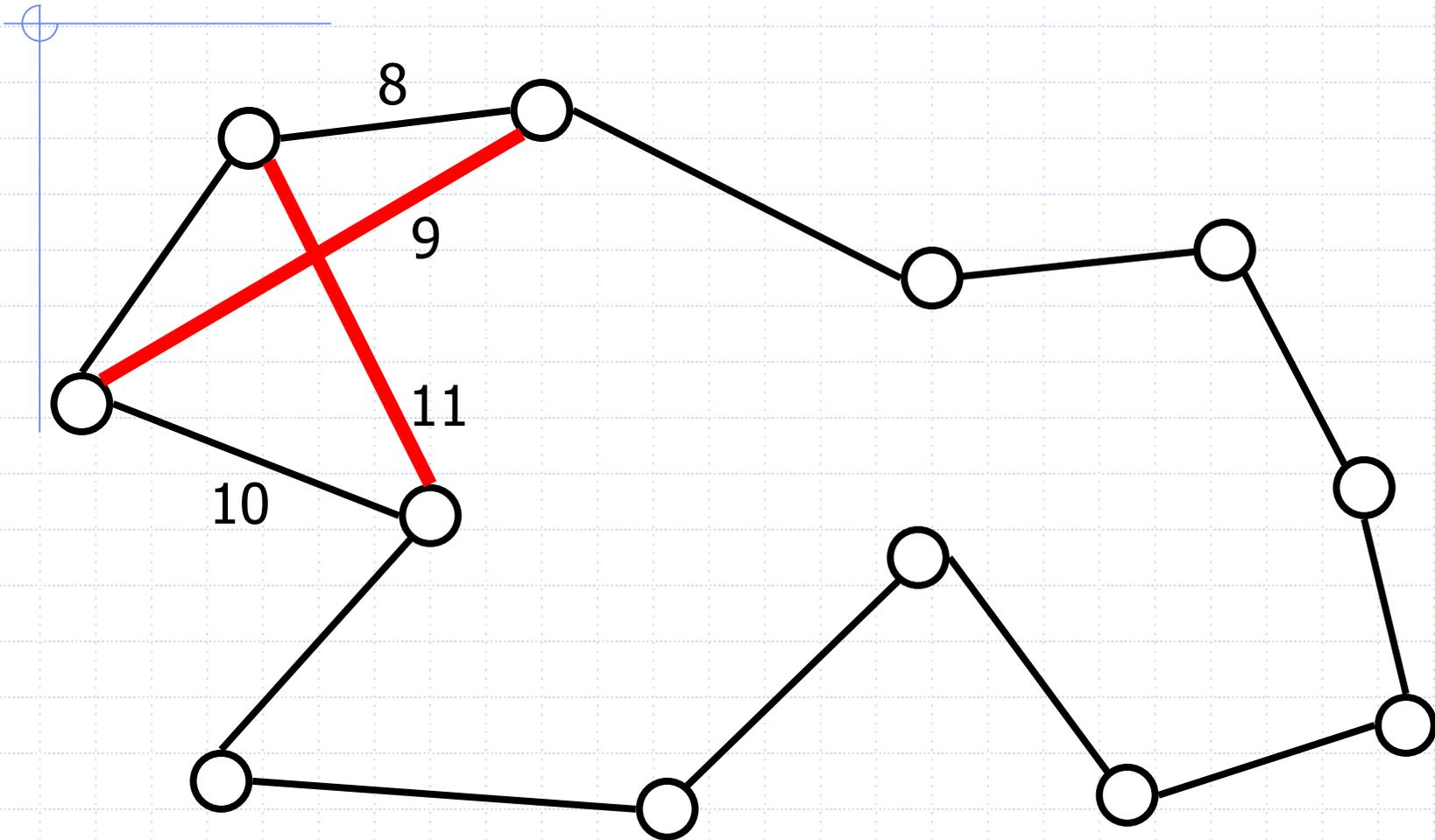


## 2-opt exchange



$$T' = T - (7 + 12) + (5 + 9) \text{ (miglioramento)}$$

## 2-opt exchange



$T' = T - (8 + 10) + (11 + 9)$ : nessun miglioramento, STOP

# 3-opt exchange

