1. Definizione del Problema

Input:

- $S = \{s_1, s_2, ..., s_n\}$: insieme ordinato di n punti (server) sulla retta reale
- $C = \{c_1, c_2, ..., c_m\}$: insieme ordinato di m punti (client) sulla retta reale
- Funzione costo: $cost(c_i, s_j) = |c_i - s_j|$

Output:

- Un matching M che assegna ogni client ad un server distinto minimizzando il costo totale

Obiettivo:
Minimizzare $\Sigma|c_i - s_j|$ per ogni coppia (c_i, s_j) nel matching

2. Pseudocodice

```
METRIC-MATCHING-LINE(S, C)
Ordina S in ordine crescente se non già ordinato
Ordina C in ordine crescente se non già ordinato
n ← |S|
m ← |C|
matching ← array vuoto di dimensione m
used_servers ← array vuoto di dimensione n
total_cost ← 0

for i ← 1 to m do
    min_cost ← ∞
    best_server ← NIL
    for j ← 1 to n do
        if not used_servers[j] then
            current_cost ← |C[i] - S[j]|
            if current_cost < min_cost then
                min_cost ← current_cost
                best_server ← j
    matching[i] ← best_server
    used_servers[best_server] ← true
```

```
    total_cost ← total_cost + min_cost

return (matching, total_cost)
```

3. Dimostrazione della Correttezza

3.1 Proprietà di Sottostruttura Ottima

Il problema presenta una sottostruttura ottima perché:

1. Dato un matching ottimo M per n client e n server
2. Se rimuoviamo una coppia (c, s) da M
3. Il matching rimanente M' deve essere ottimo per il sottoproblema con n-1 client e n-1 server

Dimostrazione per contraddizione:
Supponiamo che M' non sia ottimo per il sottoproblema. Allora esisterebbe un matching migliore M'' per il sottoproblema. Sostituendo M' con M'' nel matching originale M otterremmo un matching con costo totale minore, contraddicendo l'ottimalità di M.

3.2 Proprietà della Scelta Greedy

La strategia greedy è corretta perché:

Lemma 1: Per ogni coppia di punti (c_1, s_1) e (c_2, s_2) dove $c_1 < c_2$ e $s_1 < s_2$:
$|c_1 - s_1| + |c_2 - s_2| \leq |c_1 - s_2| + |c_2 - s_1|$

Dimostrazione:
Per la disuguaglianza triangolare e la proprietà della distanza sulla retta reale, incrociare gli assegnamenti non può mai migliorare il costo totale quando i punti sono ordinati.

4. Analisi della Complessità

4.1 Tempo di Esecuzione

- Ordinamento iniziale: O(n log n + m log m)
- Loop esterno: O(m)
- Loop interno: O(n)
- Complessità totale: O(mn + n log n + m log m)

4.2 Spazio

- Spazio ausiliario: O(n + m)
- Spazio totale: O(n + m)

5. Dimostrazione dell'Ottimalità

Per dimostrare l'ottimalità dell'algoritmo greedy:

1. **Invariante:** Ad ogni passo i, il matching parziale costruito è parte di qualche matching ottimale globale.
2. **Base:** Al passo 0, il matching vuoto è chiaramente parte di ogni matching ottimale.
3. **Passo induttivo:** Se esiste un matching ottimale che non usa l'assegnamento greedy al passo i, possiamo sempre trasformarlo in un matching ottimale che usa tale assegnamento senza aumentare il costo totale (per il Lemma 1).
4. **Conclusione:** L'algoritmo produce un matching ottimale globale.

6. Proprietà Aggiuntive

1. **Unicità:** La soluzione non è necessariamente unica
2. **Stabilità:** Il matching prodotto è stabile rispetto alle distanze
3. **Robustezza:** Piccole perturbazioni negli input producono piccole variazioni nel costo totale