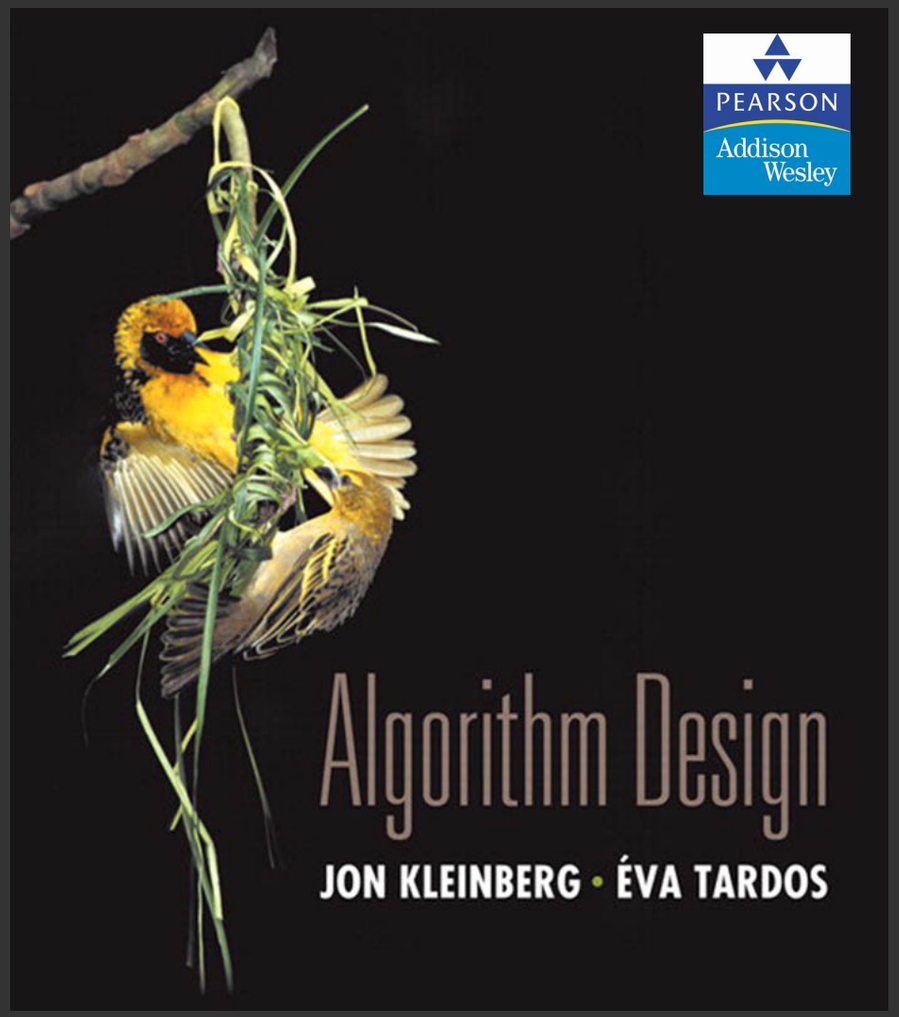


5. Divide et Impera I

- *mergesort*
- *conteggio di inversioni*
- *coppia di punti più vicina*

Traduzione e adattamento di Vincenzo Bonifaci
Original lecture slides by Kevin Wayne

http://www.cs.princeton.edu/~wayne/kleinberg-tardos

Last updated on 05/04/2020 21:56

Paradigma divide et impera

Divide et impera.

- Dividi il problema in vari sottoproblemi (dello stesso tipo).
- Risolvi (impera) ricorsivamente ciascun sottoproblema.
- Combina le soluzioni dei sottoproblemi in una soluzione complessiva.

Uso più comune.

- Dividi un problema di taglia n in due sottoproblemi di taglia $n/2$.
- Risolvi (impera) i due sottoproblemi ricorsivamente.
- Combina le due soluzioni in una complessiva. ← tempo $O(n)$

tempo $O(n)$

Conseguenza.

- Forza bruta: $\Theta(n^2)$.
- Divide et impera: $O(n \log n)$.

attribuita a Giulio Cesare

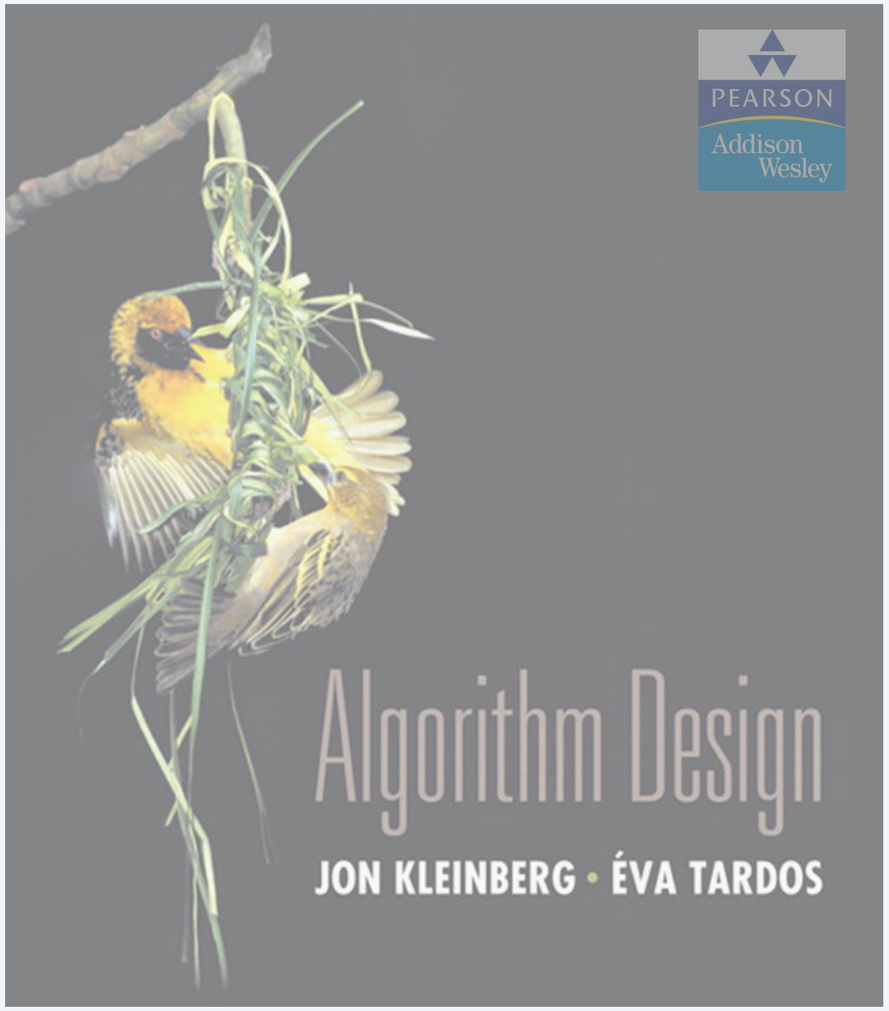
Paradigma divide et impera

Divide et impera.

- Dividi il problema in vari sottoproblemi (dello stesso tipo).
- Risolvi (impera) ricorsivamente ciascun sottoproblema.
- Combina le soluzioni dei sottoproblemi in una soluzione complessiva.

Esempi.

- Mergesort e quicksort (ordinamento).
- Quickhull e mergehull (involucro convesso).
- Algoritmo Shamos–Hoey (coppia più vicina).
- Algoritmo mediana-di-mediane (selezione).
- Algoritmo di Strassen (moltiplicazione di matrici).
- Algoritmo di Karatsuba (moltiplicazione intera).
- Algoritmo Cooley–Tukey (trasformata di Fourier veloce).
- Algoritmo Ramer–Douglas–Peucker (decimazione di una curva).



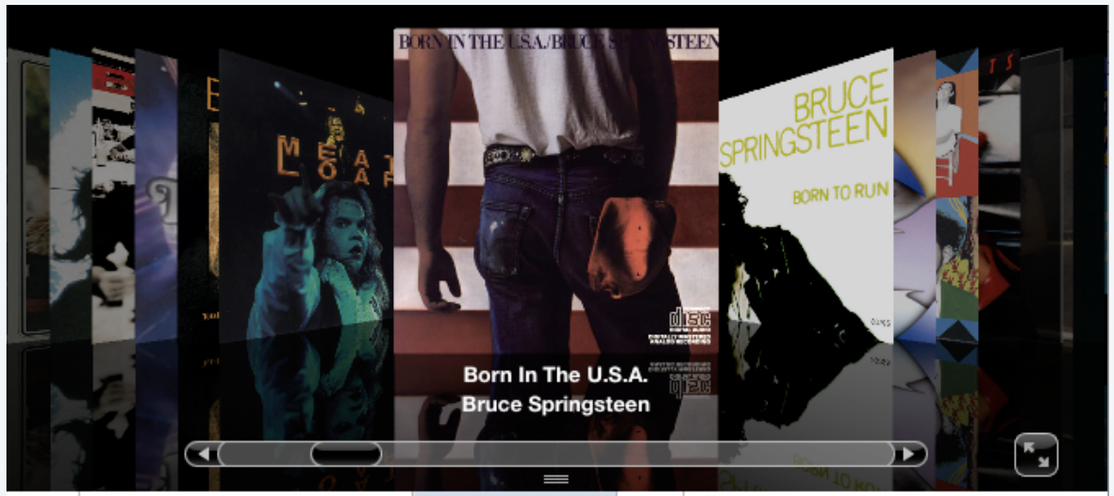
SECTIONS 5.1–5.2

5. DIVIDE ET IMPERA

- *mergesort*
- *counting inversions*
- *randomized quicksort*
- *median and selection*
- *closest pair of points*

Problema dell'ordinamento

Problema. Data una lista L di n elementi da un universo totalmente ordinato, ridisporla in ordine crescente.



	Name	Artist	Time	Album
12	Let It Be	The Beatles	4:03	Let It Be
13	Take My Breath Away	BERLIN	4:13	Top Gun – Soundtrack
14	Circle Of Friends	Better Than Ezra	3:27	Empire Records
15	Dancing With Myself	Billy Idol	4:43	Don't Stop
16	Rebel Yell	Billy Idol	4:49	Rebel Yell
17	Piano Man	Billy Joel	5:36	Greatest Hits Vol. 1
18	Pressure	Billy Joel	3:16	Greatest Hits, Vol. II (1978 – 1985) (Disc 2)
19	The Longest Time	Billy Joel	3:36	Greatest Hits, Vol. II (1978 – 1985) (Disc 2)
20	Atomic	Blondie	3:50	Atomic: The Very Best Of Blondie
21	Sunday Girl	Blondie	3:15	Atomic: The Very Best Of Blondie
22	Call Me	Blondie	3:33	Atomic: The Very Best Of Blondie
23	Dreaming	Blondie	3:06	Atomic: The Very Best Of Blondie
24	Hurricane	Bob Dylan	8:32	Desire
25	The Times They Are A-Changin'	Bob Dylan	3:17	Greatest Hits
26	Livin' On A Prayer	Bon Jovi	4:11	Cross Road
27	Beds Of Roses	Bon Jovi	6:35	Cross Road
28	Runaway	Bon Jovi	3:53	Cross Road
29	Rasputin (Extended Mix)	Boney M	5:50	Greatest Hits
30	Have You Ever Seen The Rain	Bonnie Tyler	4:10	Faster Than The Speed Of Night
31	Total Eclipse Of The Heart	Bonnie Tyler	7:02	Faster Than The Speed Of Night
32	Straight From The Heart	Bonnie Tyler	3:41	Faster Than The Speed Of Night
33	Holding Out For A Hero	Bonny Tyler	5:49	Meat Loaf And Friends
34	Dancing In The Dark	Bruce Springsteen	4:05	Born In The U.S.A.
35	Thunder Road	Bruce Springsteen	4:51	Born To Run
36	Born To Run	Bruce Springsteen	4:30	Born To Run
37	Jungleland	Bruce Springsteen	9:34	Born To Run

Applicazioni degli ordinamenti

Applicazioni ovvie.

- Organizzare una raccolta di MP3.
- Mostrare i risultati di Google PageRank.
- Elencare news RSS in ordine cronologico inverso.

Alcuni problemi sono più semplici se gli elementi sono stati ordinati.

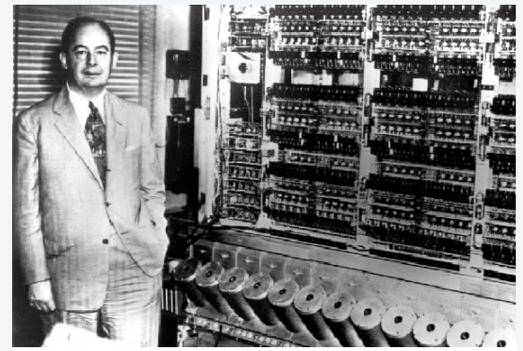
- Identificazione di outlier statistici.
- Ricerca binaria in un database.
- Rimozione di duplicati in una mailing list.

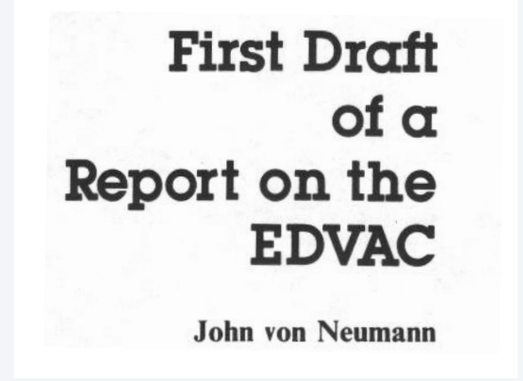
Applicazioni non ovvie.

- Involucro convesso.
- Coppia più vicina di punti.
- Schedulazione di intervalli / partizionamento di intervalli.
- Schedulazione per la minimizzazione della lateness.
- Alberi ricoprenti minimi (algoritmo di Kruskal).
- ...

Mergesort

- Ordina ricorsivamente la prima metà.
- Ordina ricorsivamente la seconda metà.
- Fondi le due metà in una sequenza ordinata.





input

A	L	G	O	R	I	T	H	M	S

ordina la prima metà

A	G	L	O	R	I	T	H	M	S

ordina la seconda metà

A	G	L	O	R	H	I	M	S	T

fondi i risultati

A	G	H	I	L	M	O	R	S	T

Fusione

Scopo. Combina due liste ordinate A e B in una sequenza ordinata C.

- Scandisci A e B da sinistra a destra.
- Confronta a_i e b_j.
- Se $a_i \leq b_j$, apponi a_i a C (è al più pari a ogni elemento residuo di B).
- Se $a_i > b_j$, apponi b_j a C (è minore di ogni elemento residuo di A).

Mergesort: implementazione

Input. Lista L di n elementi da un universo totalmente ordinato.

Output. Gli n elementi in ordine crescente.

MERGE-SORT(L)

IF (la lista L ha un solo elemento)

  RETURN L.

Dividi la lista in due metà A e B.

$A \leftarrow$ MERGE-SORT(A). ⟵ $T(n / 2)$

$B \leftarrow$ MERGE-SORT(B). ⟵ $T(n / 2)$

$L \leftarrow$ MERGE(A, B). ⟵ $\Theta(n)$

RETURN L.

Mergesort: dimostrazione di correttezza

Proposizione. Mergesort ordina correttamente qualunque lista di n elementi.

Dim. [per induzione forte su n]

- Caso base: $n = 1$.
- Ipotesi induttiva: sia vera l'ipotesi per $1, 2, \ldots, n-1$.
- Per ipotesi induttiva, mergesort ordina sia la metà sinistra che la destra.
- L'operazione di fusione combina le due liste ordinate in una lista ordinata complessiva. ▪

Un'utile relazione di ricorrenza

Def. $T(n)$ = massimo numero di confronti eseguiti da mergesort su una lista di lunghezza n.

Ricorrenza.

$$T(n) \leq \begin{cases} 0 & \text{if } n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n & \text{if } n > 1 \end{cases}$$

tra $\lfloor n / 2 \rfloor$ e $n - 1$ confronti

Soluzione. $T(n)$ è $O(n \log_2 n)$.

Dimostrazioni assortite. Descriviamo vari modi di risolvere questa ricorrenza.

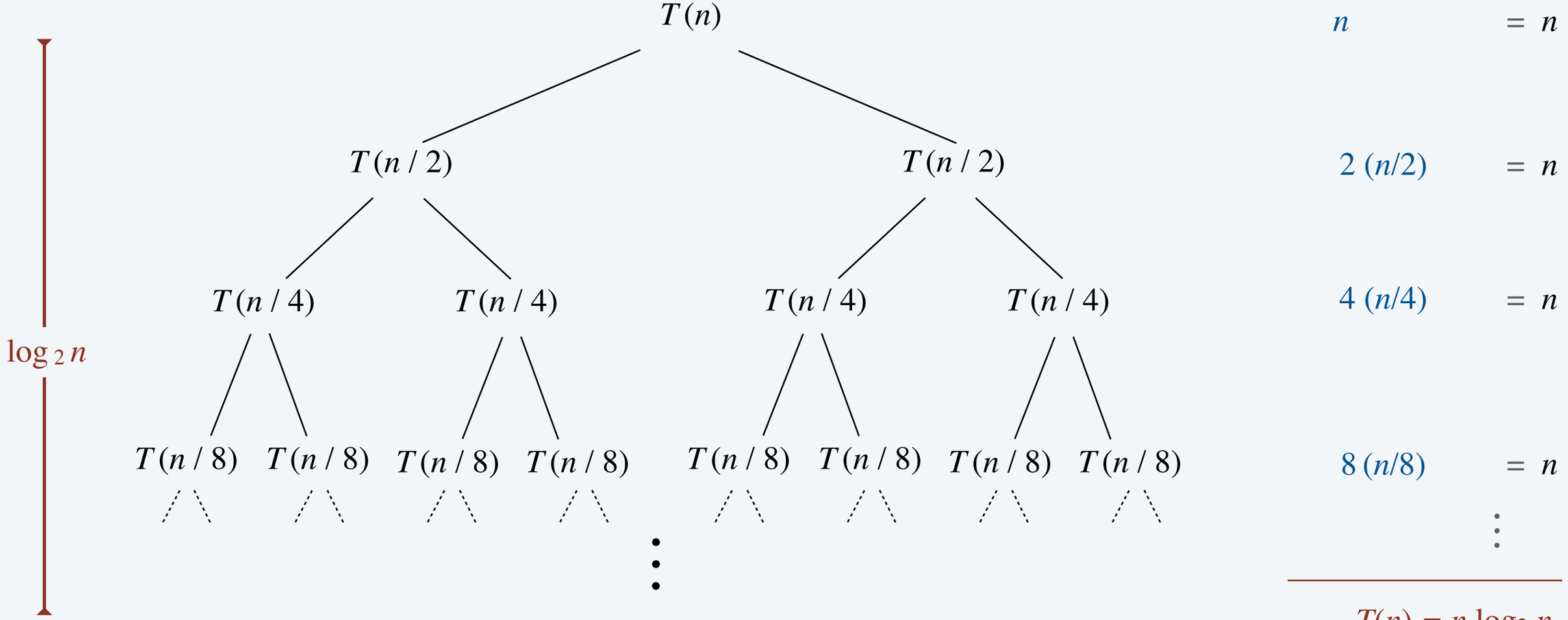
Inizialmente assumiamo che n sia una potenza di 2 e rimpiazziamo ≤ con = nella ricorrenza.

Ricorrenza divide et impera: albero di ricorsione

Proposizione. Se $T(n)$ soddisfa la ricorrenza seguente, allora $T(n) = n \log_2 n$.

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ 2\,T(n/2) + n & \text{if } n > 1 \end{cases}$$

assume che n sia una potenza di 2



Dimostrazione per induzione

Proposizione. Se $T(n)$ soddisfa la seguente ricorrenza, allora $T(n) = n \log_2 n$.

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ 2\,T(n/2) + n & \text{if } n > 1 \end{cases}$$

assume che n sia una potenza di 2

Dim. [per induzione su n]

- Caso base: quando $n = 1$, $T(1) = 0 = n \log_2 n$.
- Ipotesi induttiva: assumiamo $T(n) = n \log_2 n$.
- Obiettivo: mostrare che $T(2n) = 2n \log_2 (2n)$.

ricorrenza

$$T(2n) = 2\,T(n) + 2n$$

ipotesi induttiva

$$= 2\,n \log_2 n + 2n$$

$$= 2\,n\,(\log_2 (2n) - 1) + 2n$$

$$= 2\,n \log_2 (2n). \quad \blacksquare$$

Divide et impera: quiz 1

Qual è la soluzione esatta della seguente ricorrenza?

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + n - 1 & \text{if } n > 1 \end{cases}$$

non assume più che n sia una potenza di 2

A. $T(n) = n \lfloor \log_2 n \rfloor$

B. $T(n) = n \lceil \log_2 n \rceil$

C. $T(n) = n \lfloor \log_2 n \rfloor + 2^{\lfloor \log_2 n \rfloor} - 1$

D. $T(n) = n \lceil \log_2 n \rceil - 2^{\lceil \log_2 n \rceil} + 1$

E. Neanche Knuth lo sa.

Digressione: limitazioni inferiori sulla complessità dell'ordinamento

Problematica. Come dimostrare un bound per tutti i concepibili algoritmi?

Modello di calcolo. Alberi di confronti.

- Si può accedere agli elementi solo tramite confronti a coppie.
- Tutte le altre operazioni (controllo di flusso, spostamento di dati, ecc.) non vengono conteggiate.

Modello di costo. Numero di confronti.

Q. È un modello realistico?

A1. Sì. Sort in Java, Python, C++, ...

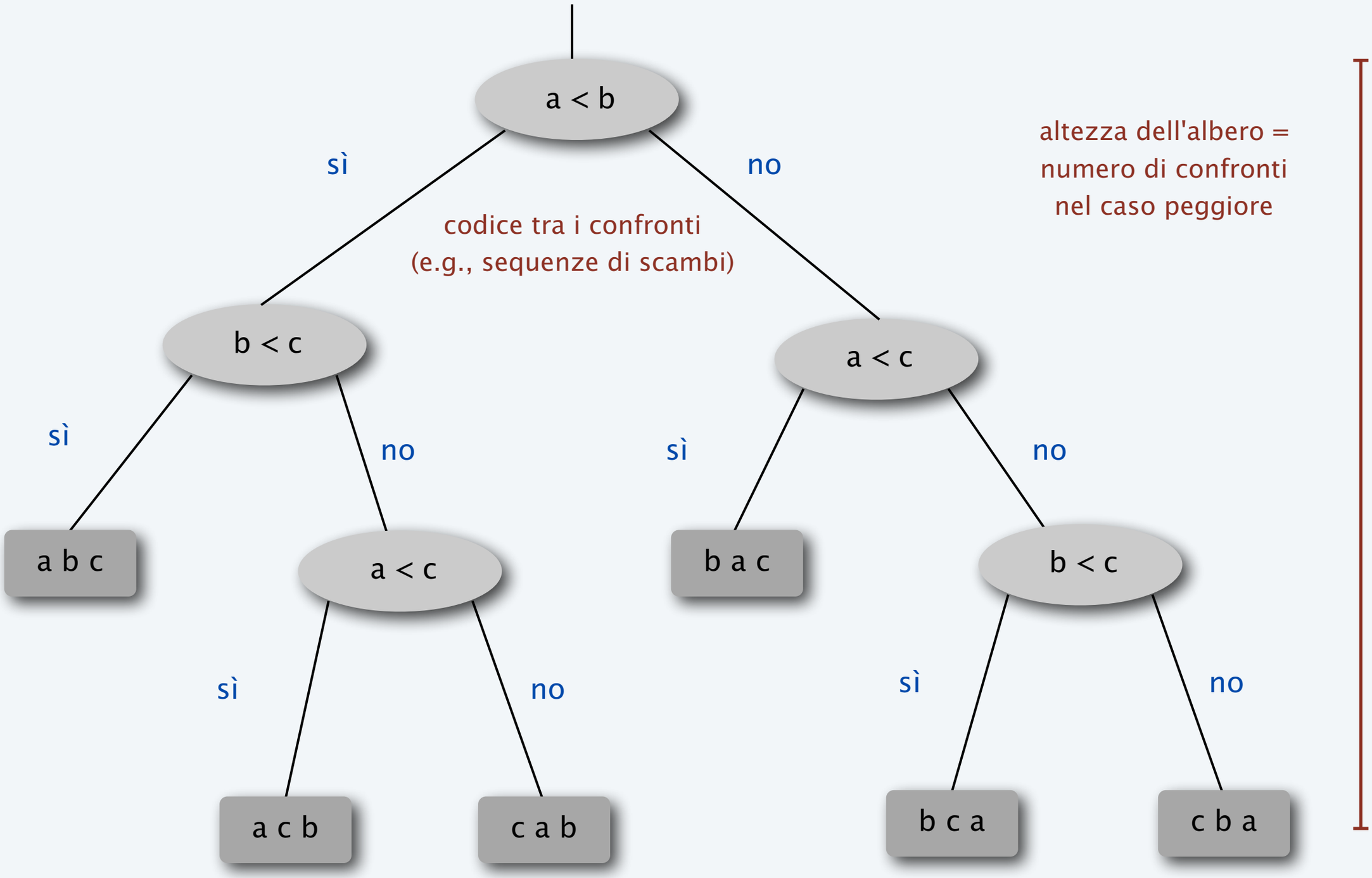
A2. Sì. Mergesort, insertion sort, quicksort, heapsort, ...

A3. No. Bucket sort, radix sorts, ...

sort(*, *key=None*, *reverse=False*)

This method sorts the list in place, using only < comparisons between items. Exceptions are not suppressed - if any comparison operations fail, the entire sort operation will fail (and the list will likely be left in a partially modified state).

Albero dei confronti (per 3 chiavi distinte a, b, e c)

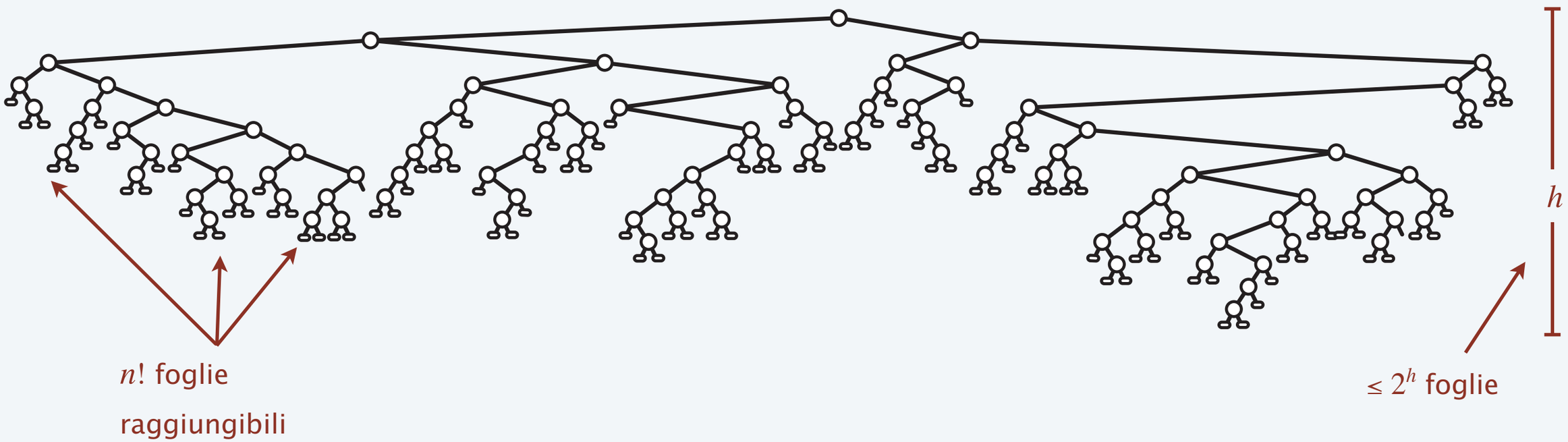


Limitazione inferiore per l'ordinamento

Teorema. Qualunque algoritmo deterministico per l'ordinamento basato su confronti deve effettuare $\Omega(n \log n)$ confronti nel caso peggiore.

Dim. [teoria dell'informazione]

- Assumiamo che l'array consista di n valori distinti, da a_1 ad a_n.
- Numero minimo di confronti (nel caso peggiore) = altezza h dell'albero dei confronti.
- Un albero binario di altezza h ha $\leq 2^h$ foglie.
- $n!$ diversi ordinamenti $\Rightarrow$ $n!$ foglie raggiungibili.



Limitazione inferiore per l'ordinamento

Teorema. Qualunque algoritmo deterministico per l'ordinamento basato su confronti deve effettuare $\Omega(n \log n)$ confronti nel caso peggiore.

Dim. [teoria dell'informazione]

- Assumiamo che l'array consista di n valori distinti, da a_1 ad a_n.
- Numero minimo di confronti = altezza h dell'albero dei confronti.
- Un albero binario di altezza h ha $\leq 2^h$ foglie.
- $n!$ diversi ordinamenti $\Rightarrow$ $n!$ foglie raggiungibili.
- $$2^h \geq \#\text{ foglie raggiungibili} = n!$$
  $$\Rightarrow \quad h \geq \log_2 (n!)$$
  $$\geq n \log_2 n - n \log_2 e \quad \blacksquare$$

Formula di Stirling
$n! \geq (n/e)^n$

Nota. Questa limitazione può essere estesa ad algoritmi randomizzati.

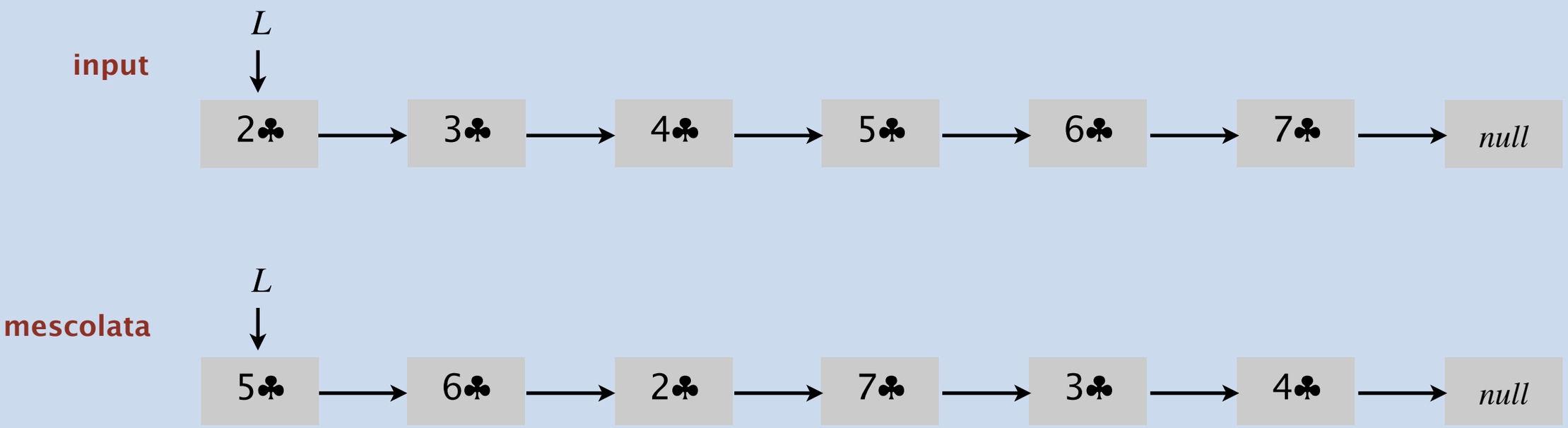
MESCOLARE UNA LISTA COLLEGATA

Problema. Data una lista collegata, ridisporre i suoi nodi uniformemente a caso.

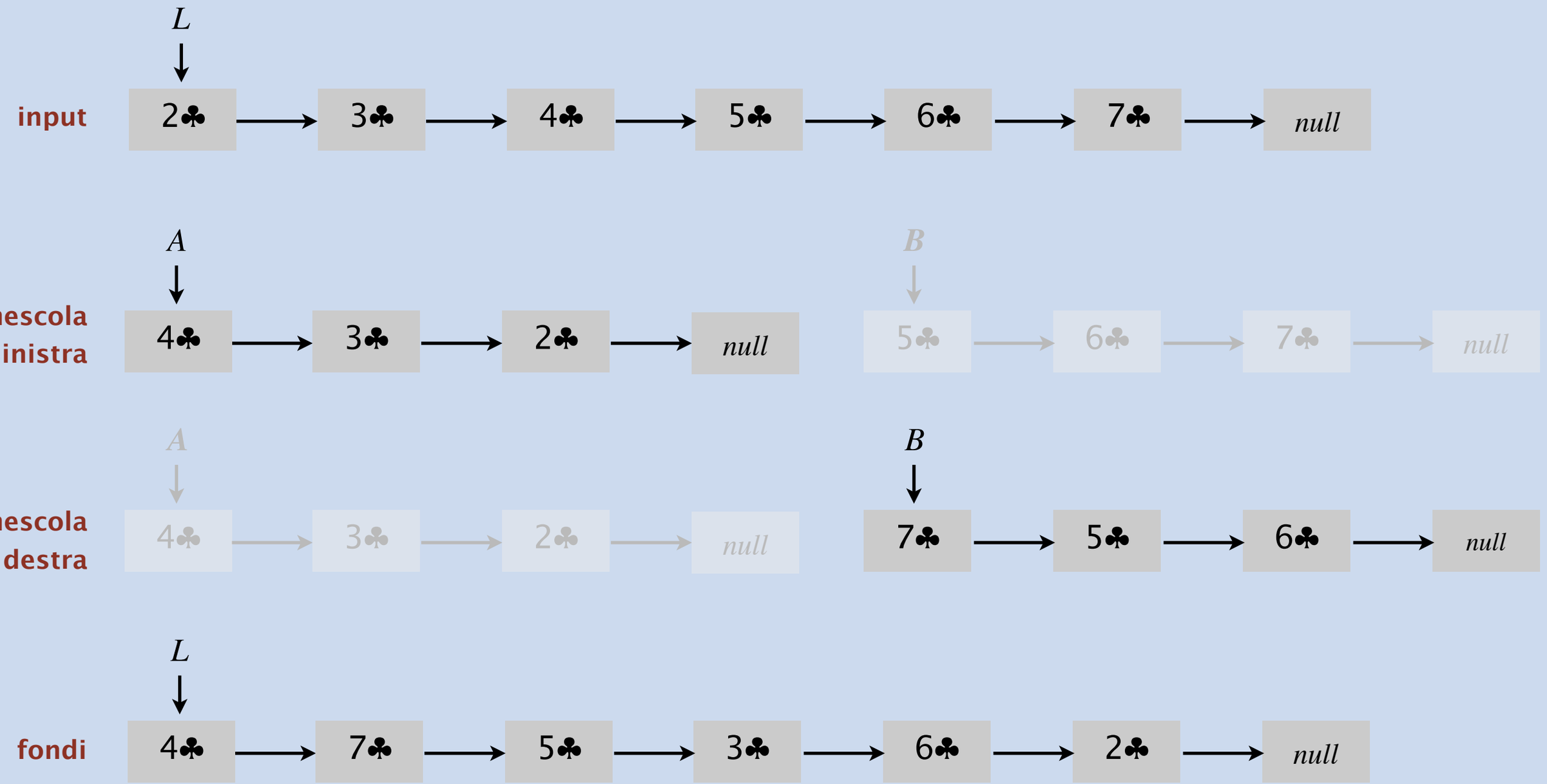
Assunzione. Si ha un generatore perfetto di numeri casuali.

tutte le $n!$ permutazioni sono equiprobabili

Performance. Tempo $O(n \log n)$, spazio aggiuntivo $O(\log n)$.



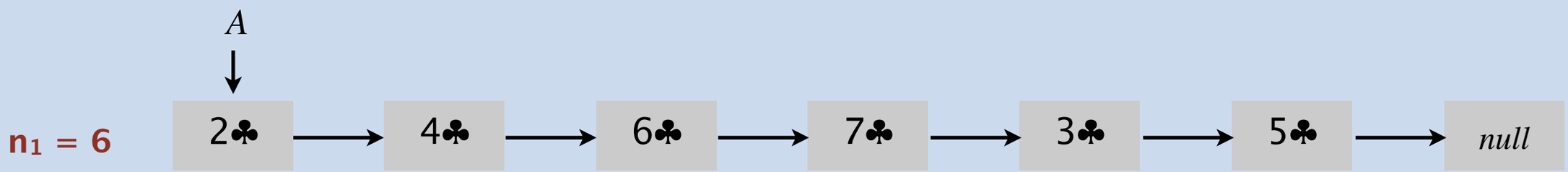
Mescolamento per fusione di una lista collegata

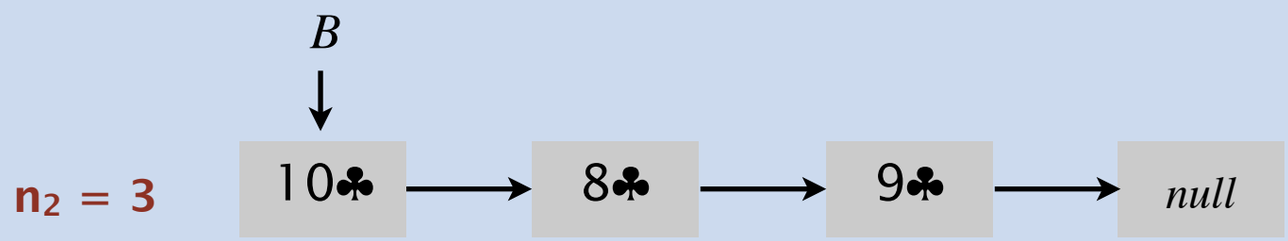


Proposizione. Se la fusione può essere effettuata in tempo $O(n)$, l'algoritmo mescola una lista collegata in tempo $O(n \log n)$ usando spazio extra $O(\log n)$.

Fondere due liste collegate mescolate

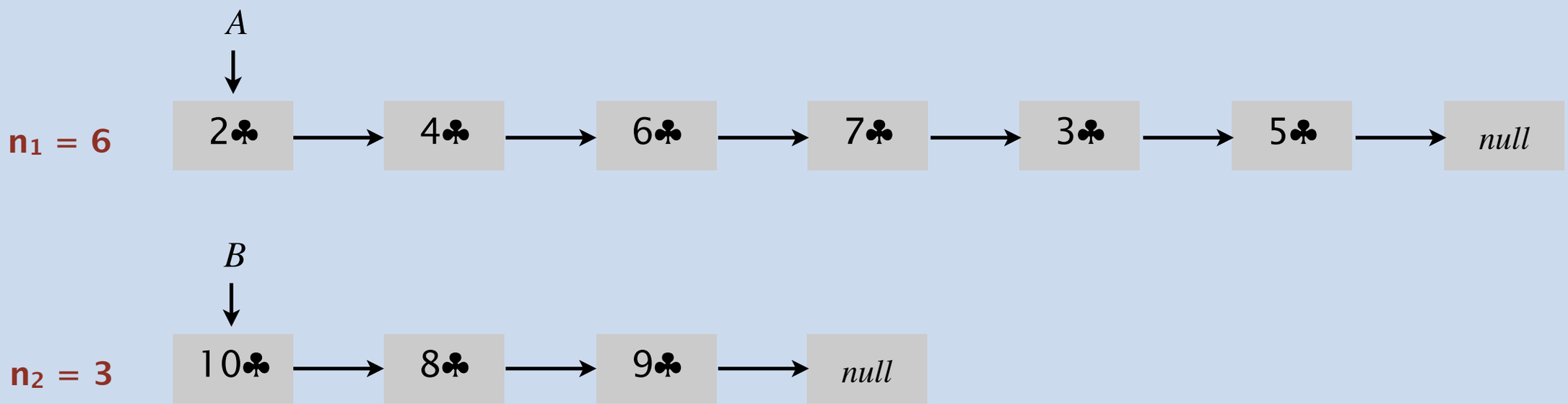
Problema. Date due liste collegate mescolate, creare una lista collegata mescolata complessiva.



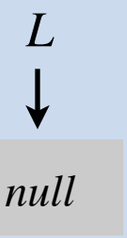


Fondere due liste collegate mescolate

Problema. Date due liste collegate mescolate, creare una lista collegata mescolata complessiva.

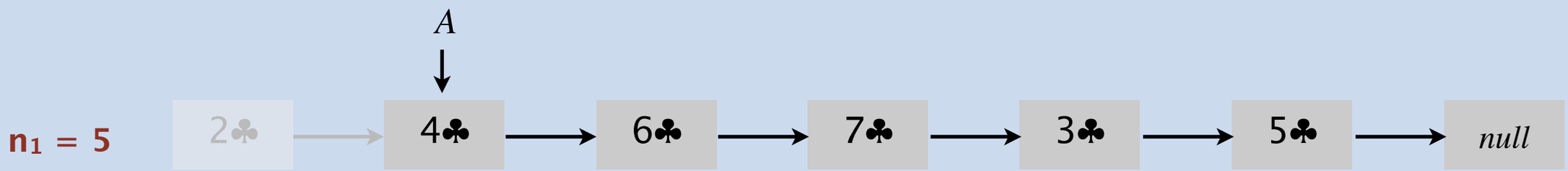


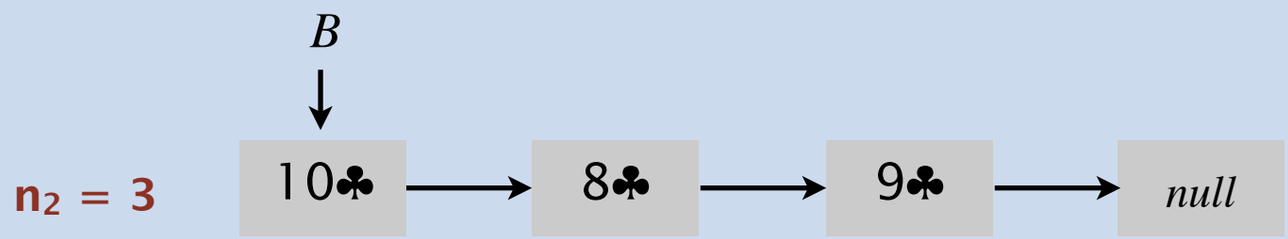
Soluzione. Scegli il primo elemento di A con probabilità $n_1 / (n_1 + n_2)$; scegli il primo elemento di B con probabilità $n_2 / (n_1 + n_2)$; ripeti.



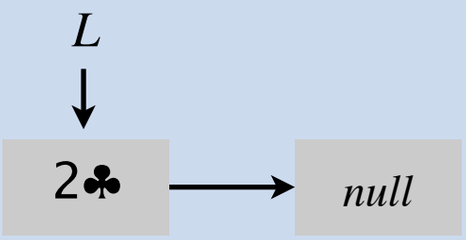
Fondere due liste collegate mescolate

Problema. Date due liste collegate mescolate, creare una lista collegata mescolata complessiva.



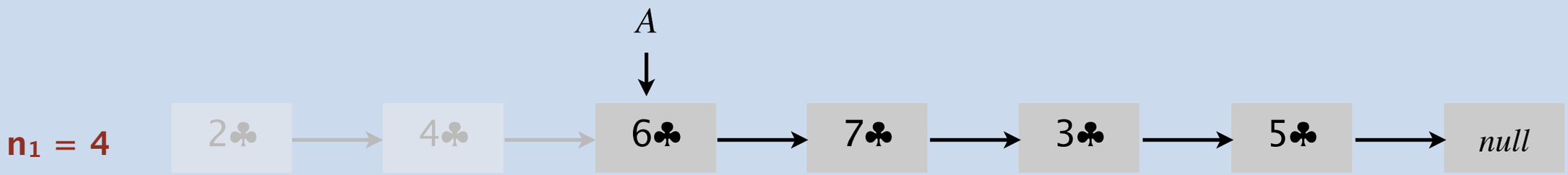


Soluzione. Scegli il primo elemento di A con probabilità $n_1 / (n_1 + n_2)$; scegli il primo elemento di B con probabilità $n_2 / (n_1 + n_2)$; ripeti.



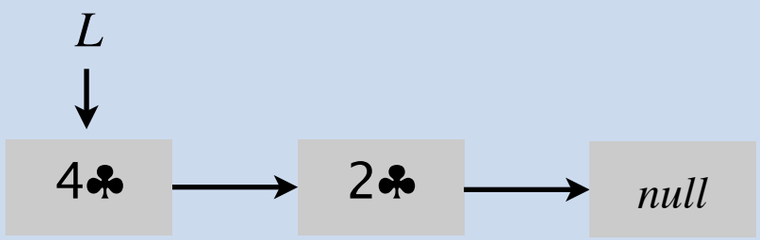
Fondere due liste collegate mescolate

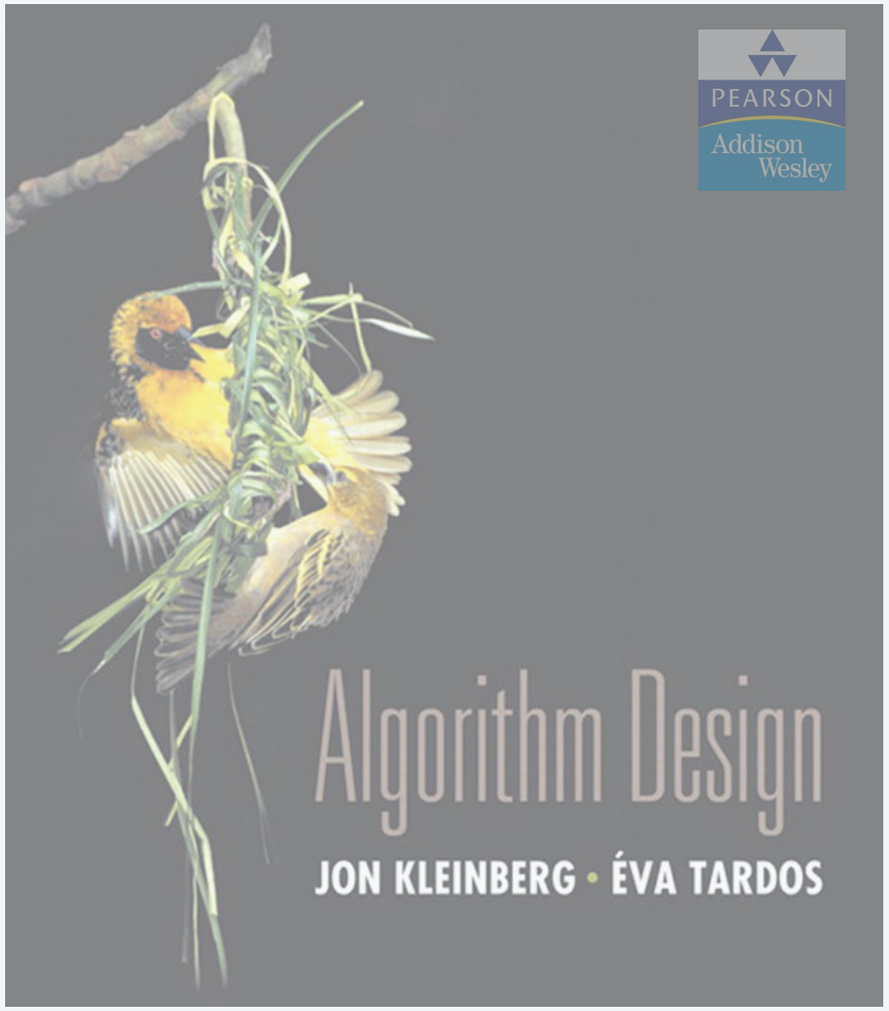
Problema. Date due liste collegate mescolate, creare una lista collegata mescolata complessiva.



Gilbert–Shannon–Reeds model

Soluzione. Scegli il primo elemento di A con probabilità $n_1 / (n_1 + n_2)$; scegli il primo elemento di B con probabilità $n_2 / (n_1 + n_2)$; ripeti.





SECTION 5.3

5. DIVIDE ET IMPERA

- *mergesort*
- *conteggio di inversioni*
- *randomized quicksort*
- *median and selection*
- *closest pair of points*

Conteggio delle inversioni

Sito di musica vuole abbinare le vostre preferenze musicali con altri utenti.

- Voi date un'ordine di preferenza a n canzoni.
- Il sito di musica consulta un database per cercare utenti con gusti simili.

Metrica di similarità: numero di inversioni tra due liste di preferenze.

- La mia classifica: $1, 2, \ldots, n$.
- La tua classifica: $a_1, a_2, \ldots, a_n$.
- Le canzoni i e j sono un'inversione se $i < j$, ma $a_i > a_j$.

	A	B	C	D	E
me	1	2	3	4	5
te	1	3	4	2	5

2 inversioni: 3-2, 4-2

Forza bruta: provare tutte le $\Theta(n^2)$ coppie.

Conteggio di inversioni: applicazioni

- Teoria del voto.
- Sistemi di raccomandazioni.
- Misurazione di "quanto" un array è ordinato.
- Analisi di sensitività della funzione di ranking di Google.
- Aggregazione del rank per effettuare meta-ricerche sul Web.
- Statistica nonparametrica (e.g., distanza tau di Kendall).

Rank Aggregation Methods for the Web

Cynthia Dwork* Ravi Kumar† Moni Naor‡ D. Sivakumar§

ABSTRACT

We consider the problem of combining ranking results from various sources. In the context of the Web, the main applications include building meta-search engines, combining ranking functions, selecting documents based on multiple criteria, and improving search precision through word associations. We develop a set of techniques for the rank aggregation problem and compare their performance to that of well-known methods. A primary goal of our work is to design rank aggregation techniques that can effectively combat "spam," a serious problem in Web searches. Experiments show that our methods are simple, efficient, and effective.

Keywords: rank aggregation, ranking functions, meta-search, multi-word queries, spam

Conteggio di inversioni: divide et impera

- Divide: separa la lista in due metà A e B.
- Impera: conta ricorsivamente le inversioni in ogni metà.
- Combina: conta le inversioni (a, b) con $a \in A$ e $b \in B$.
- Restituisci la somma dei tre conteggi.

input

1	5	4	8	10	2	6	9	3	7

conta inversioni nella metà A

1	5	4	8	10

5-4

conta inversioni nella metà B

2	6	9	3	7

6-3 9-3 9-7

conta inversioni (a, b) con a ∈ A e b ∈ B

1	5	4	8	10	2	6	9	3	7

4-2 4-3 5-2 5-3 8-2 8-3 8-6 8-7 10-2 10-3 10-6 10-7 10-9

output 1 + 3 + 13 = 17

Conteggio di inversioni: come combinare due sottoproblemi?

Q. Come contare le inversioni (a, b) con $a \in A$ e $b \in B$?

A. Facile se A e B sono ordinati!

Algoritmo di riscaldamento.

- Ordina A e B.
- Per ogni elemento $b \in B$,
  - ricerca binaria in A per trovare quanti elementi di A son maggiori di b.

lista A

7	10	18	3	14

lista B

20	23	2	11	16

ordina A

3	7	10	14	18

ordina B

2	11	16	20	23

ricerca binaria per contare le inversioni (a, b) con a ∈ A e b ∈ B

3	7	10	14	18

2	11	16	20	23
5	2	1	0	0

Conteggio di inversioni: come combinare due sottoproblemi?

Conta le inversioni (a, b) con $a \in A$ e $b \in B$, assumendo A e B siano ordinati.

- Scandisci A e B da sinistra a destra.
- Confronta a_i e b_j.
- Se $a_i < b_j$, allora a_i non è invertito con nessun elemento rimasto in B.
- Se $a_i > b_j$, allora b_j è invertito con ogni elemento rimasto in A.
- Apponi l'elemento più piccolo alla lista ordinata C.

Conteggio di inversioni: implementazione dell'algoritmo

Input. Lista L.

Output. Numero di inversioni in L ed L ordinata.

SORT-AND-COUNT(L)

IF (lista L ha un elemento)

  RETURN $(0, L)$.

Dividi la lista in due metà A e B.

$(r_A,\ A) \leftarrow$ SORT-AND-COUNT(A). ⟵ $T(n / 2)$

$(r_B,\ B) \leftarrow$ SORT-AND-COUNT(B). ⟵ $T(n / 2)$

$(r_{AB},\ L) \leftarrow$ MERGE-AND-COUNT(A, B). ⟵ $\Theta(n)$

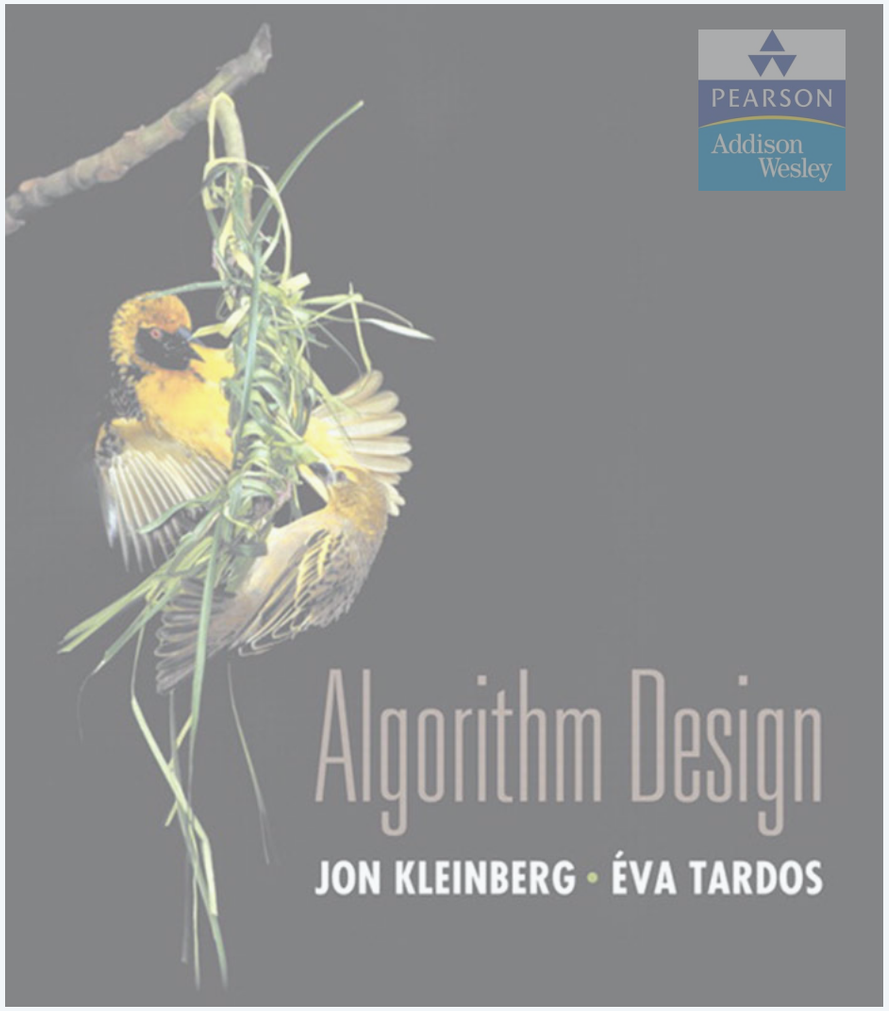
RETURN $(r_A + r_B + r_{AB},\ L)$.

Conteggio di inversioni: analisi dell'algoritmo

Proposizione. L'algoritmo sort-and-count conta il numero di inversioni in una permutazione di ordine n in tempo $O(n \log n)$.

Dim. Il tempo di esecuzione del caso peggiore $T(n)$ soddisfa la ricorrenza:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n) & \text{if } n > 1 \end{cases}$$



SECTION 5.4

5. DIVIDE ET IMPERA

- *mergesort*
- *counting inversions*
- *randomized quicksort*
- *median and selection*
- *coppia di punti più vicina*

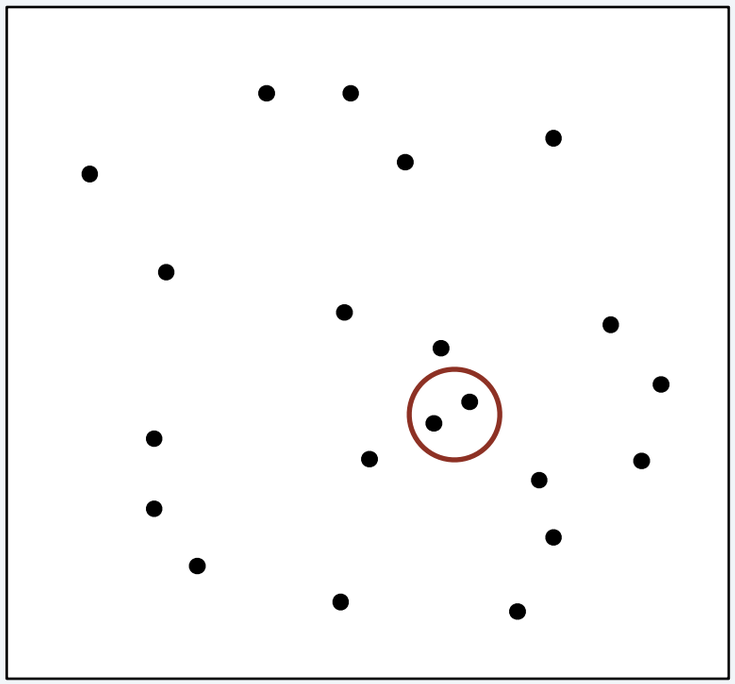
Coppia di punti più vicina

Problema della coppia di punti più vicina. Dati n punti nel piano, trovare la coppia con la più piccola distanza euclidea.

Primitiva geometrica fondamentale.

- Grafica, visione artificiale, sistemi informativi geografici, modellazione molecolare, controllo del traffico aereo.
- Caso particolare di nearest neighbor, MST euclideo, diagrammi Voronoi.

problemi che ammettono algoritmi efficienti ispirati a quello della coppia più vicina



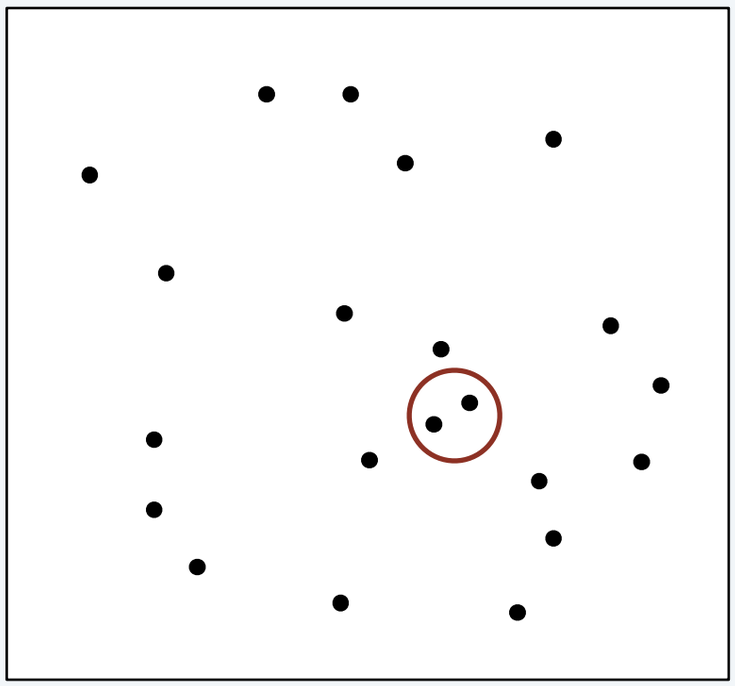
Coppia di punti più vicina

Problema della coppia di punti più vicina. Dati n punti nel piano, trovare la coppia con la più piccola distanza euclidea.

Forza bruta. Controlla tutte le coppie calcolando la distanza $\Theta(n^2)$ volte.

Versione 1D. Semplice algoritmo $O(n \log n)$ se i punti giacciono su una retta.

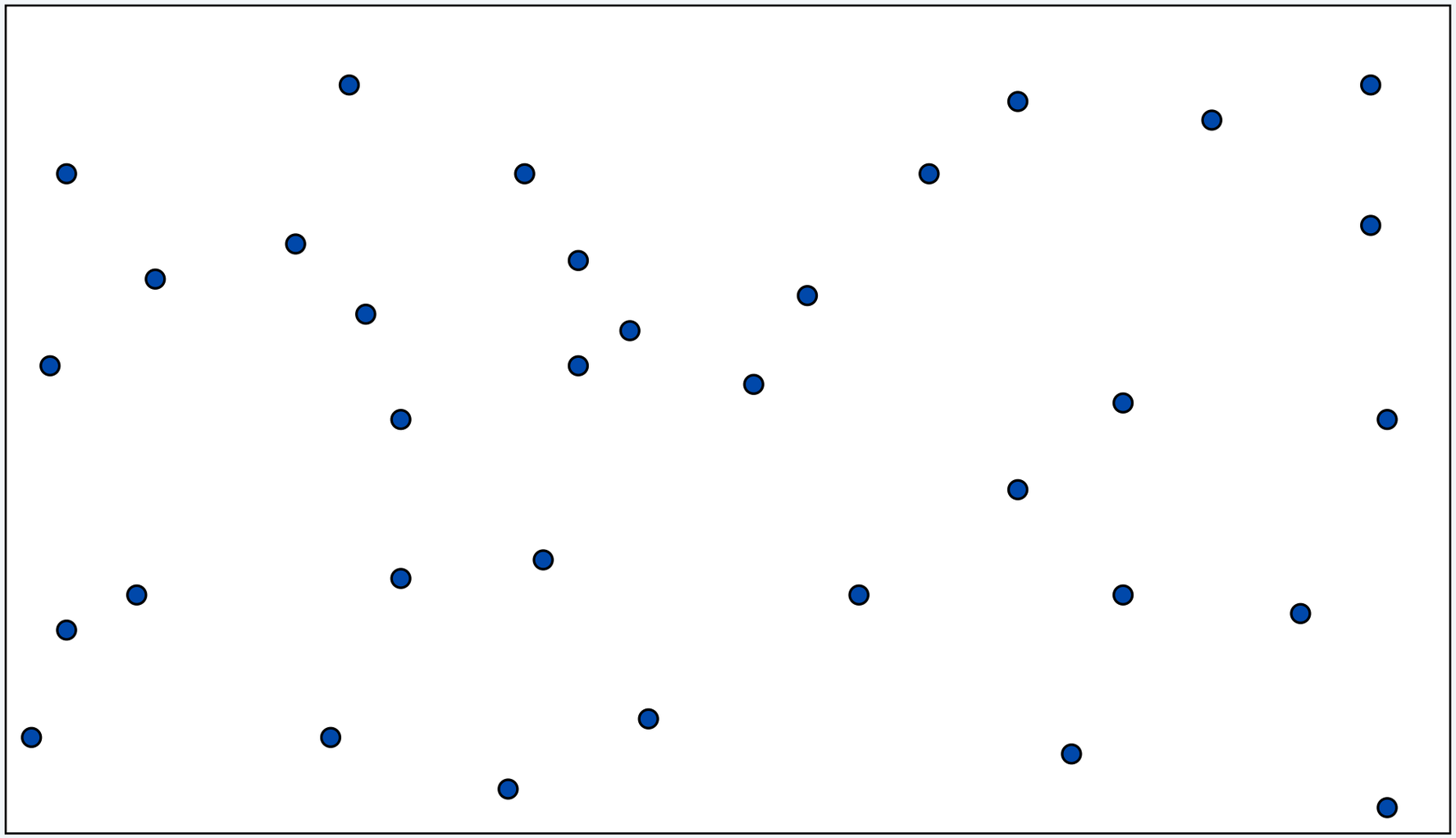
Assunzione di non-degeneratezza. Punti distinti hanno coordinate x distinte.



Coppia di punti più vicina: primo tentativo

Soluzione basata su ordinamento.

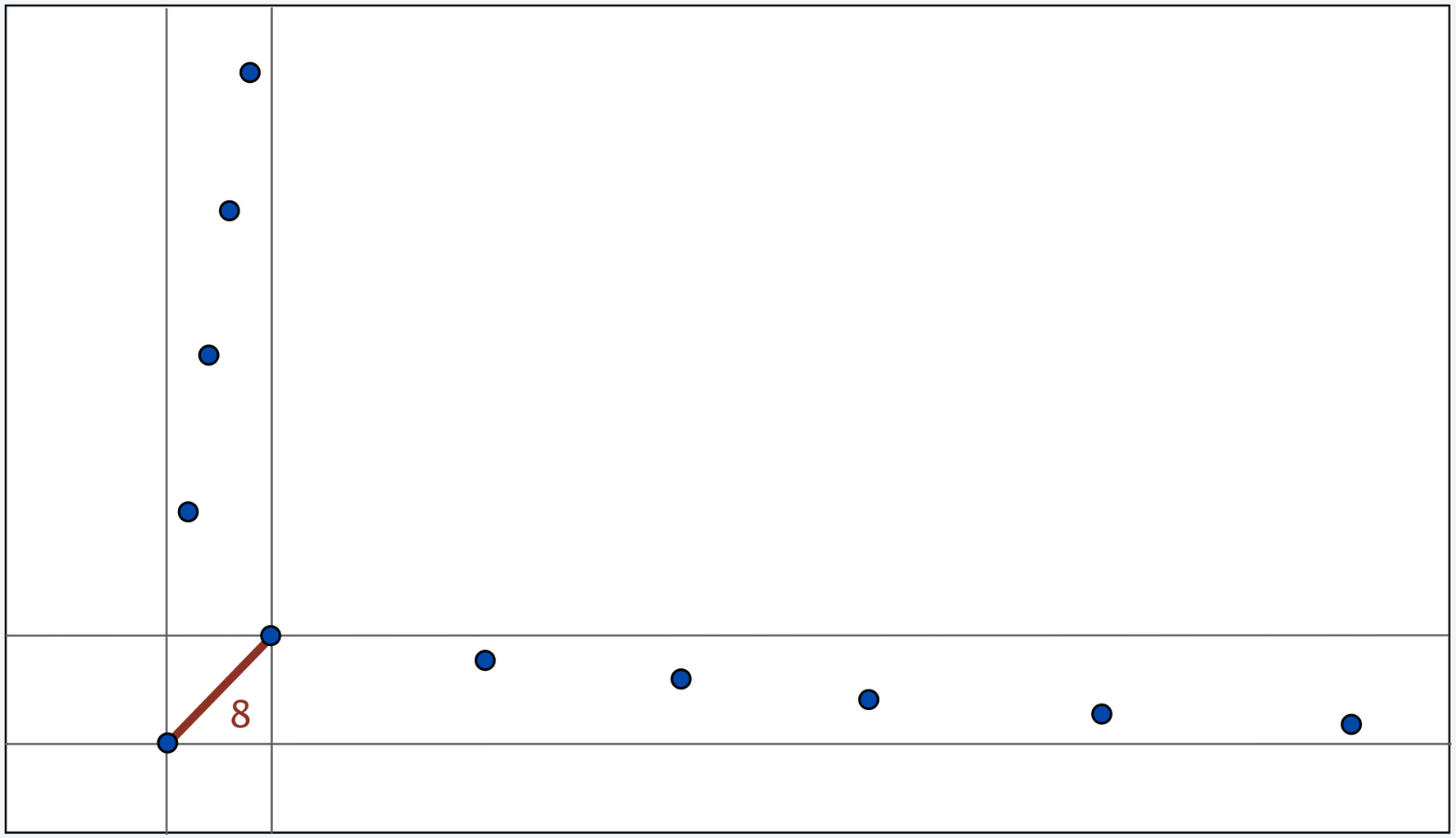
- Ordina per coordinata x e considera i punti vicini.
- Ordina per coordinata y e considera i punti vicini.



Coppia di punti più vicina: primo tentativo

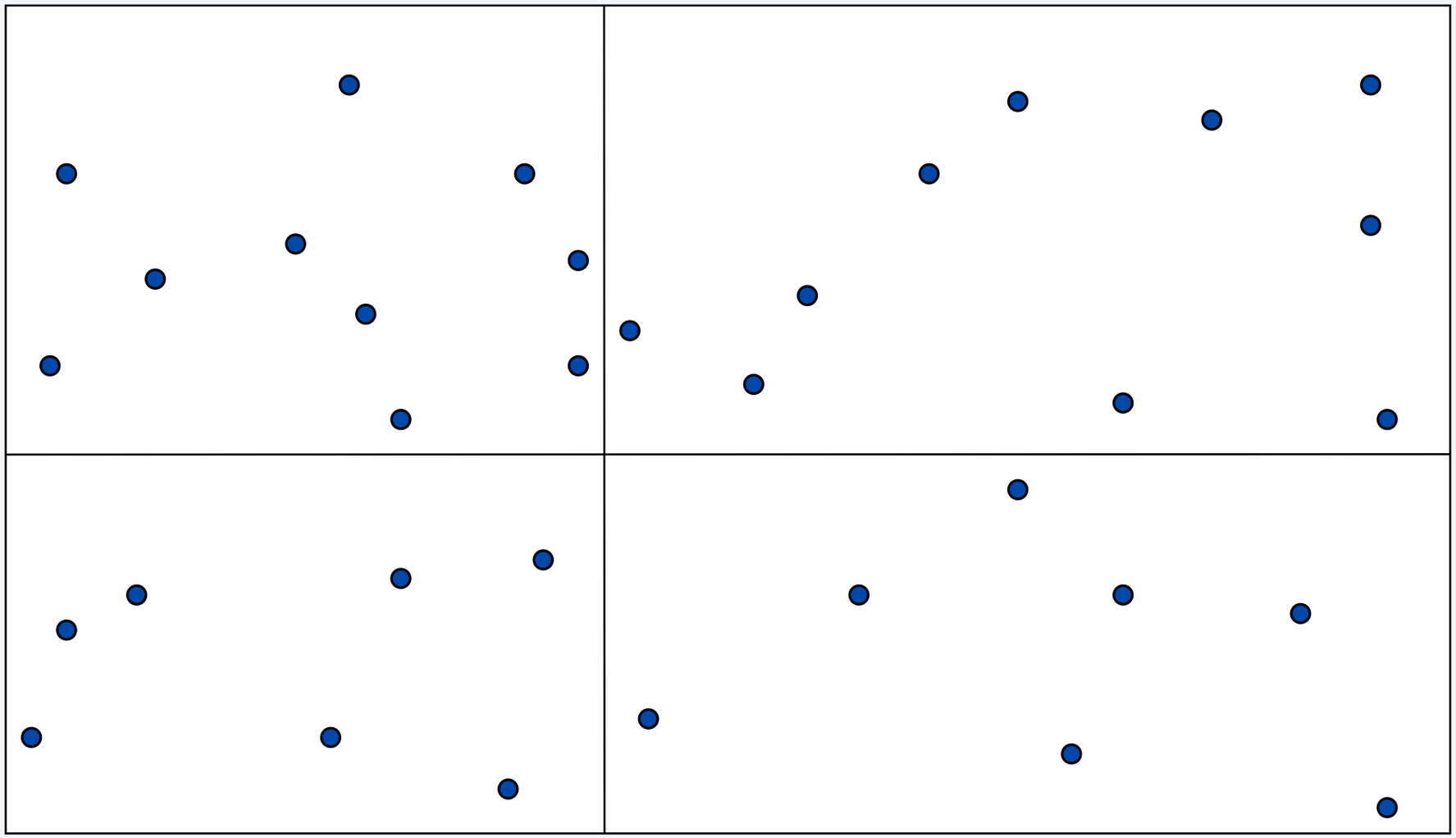
Soluzione basata su ordinamento.

- Ordina per coordinata x e considera i punti vicini.
- Ordina per coordinata y e considera i punti vicini.



Coppia di punti più vicina: secondo tentativo

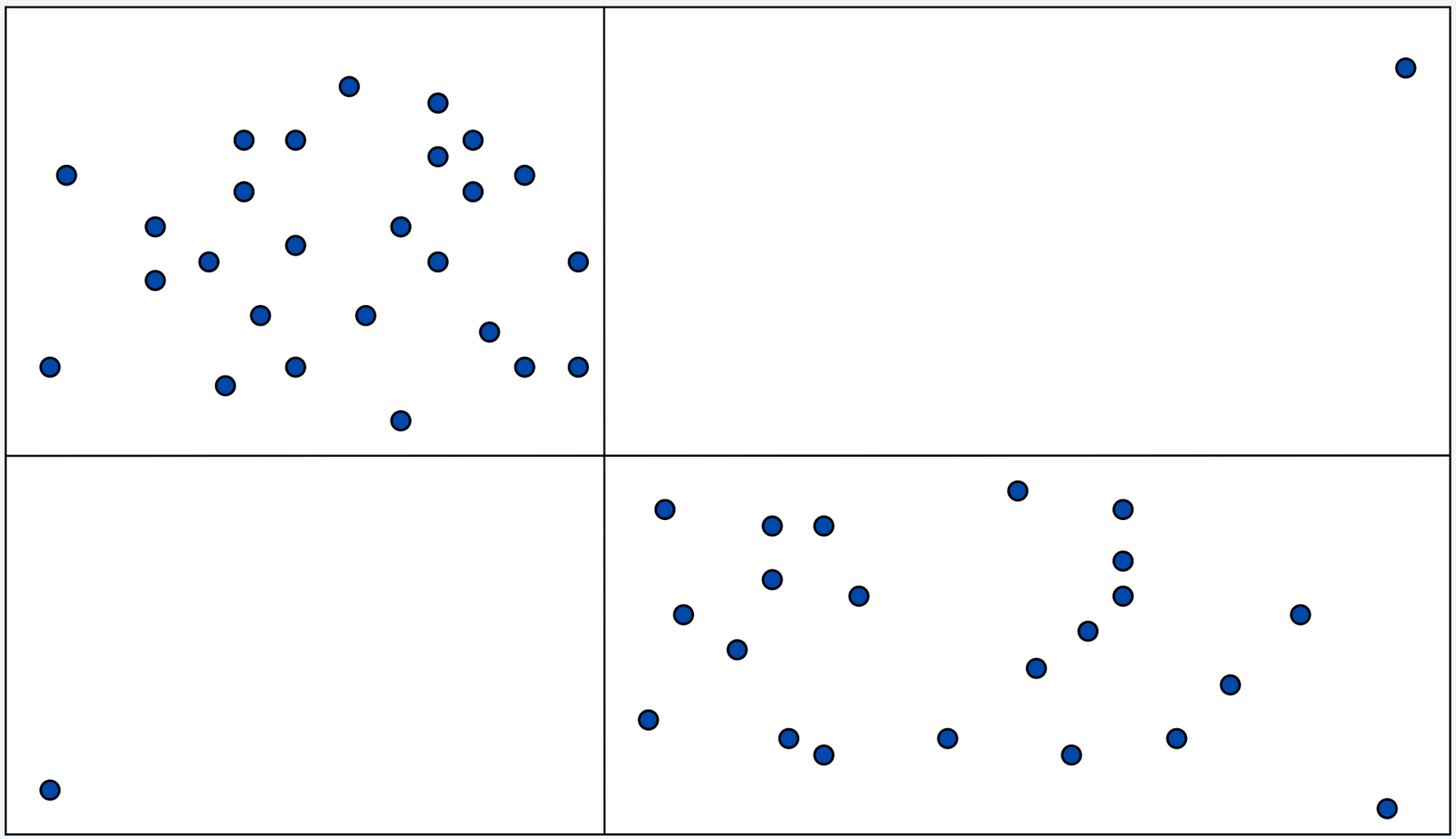
Dividi. Suddividi la regione in 4 quadranti.



Coppia di punti più vicina: secondo tentativo

Dividi. Suddividi la regione in 4 quadranti.

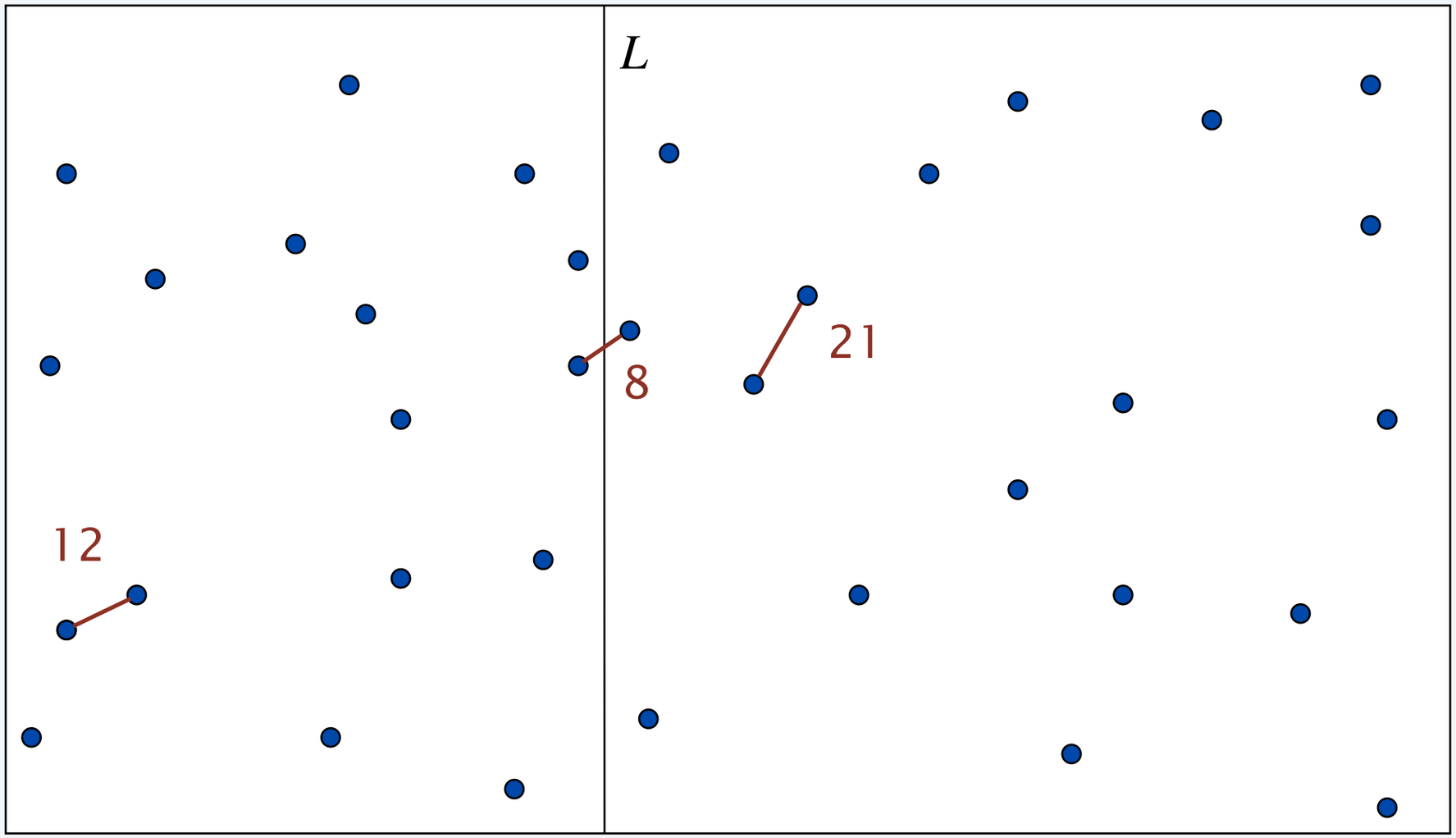
Ostacolo. È impossibile garantire $n/4$ punti in ogni sottoregione.



Coppia di punti più vicina: algoritmo divide et impera

- Divide: traccia una linea verticale L che lasci $n/2$ punti su ogni lato.
- Impera: trova la coppia più vicina ricorsivamente per i due lati.
- Combina: trova la coppia più vicina con un punto in ciascuno dei lati.
- Restituisci la migliore delle 3 soluzioni.

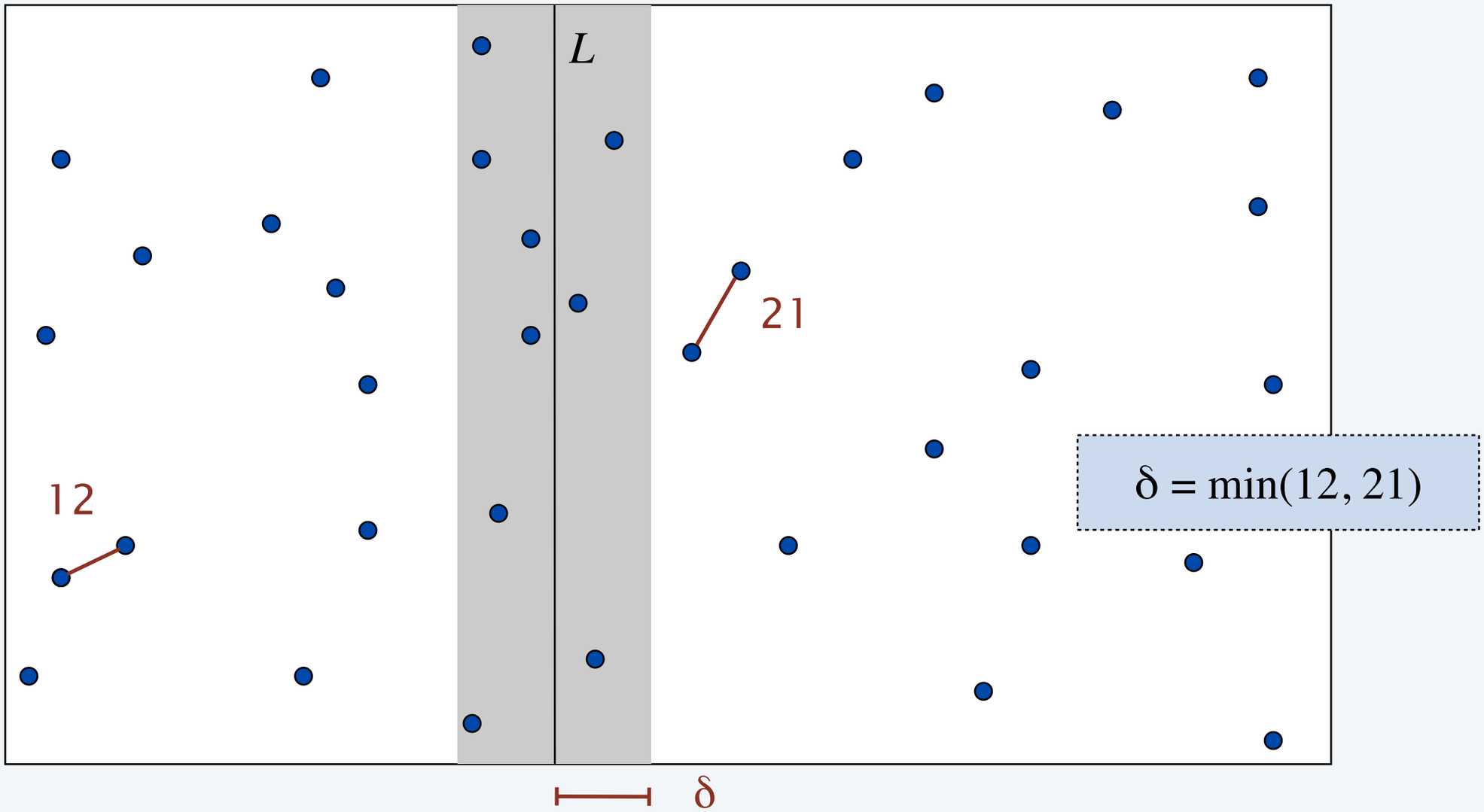
sembra $\Theta(n^2)$



Come trovare la coppia più vicina con un punto per ogni lato?

Trova la coppia più vicina con un punto per lato, assumendo distanza < δ.

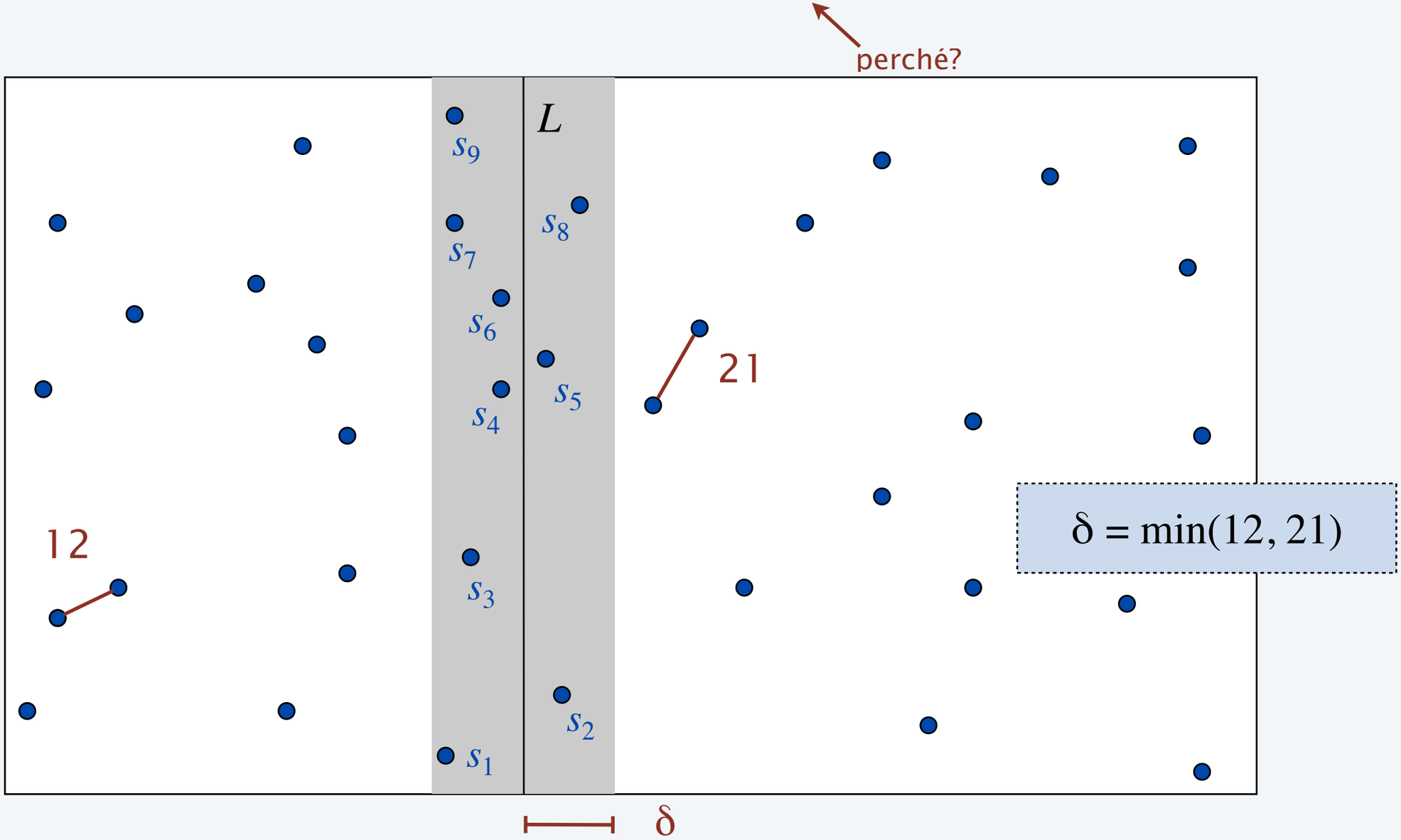
- Osservazione: è sufficiente considerare solo i punti a distanza ≤ δ dalla linea L.



Come trovare la coppia più vicina con un punto per ogni lato?

Trova la coppia più vicina con un punto per lato, assumendo distanza < δ.

- Osservazione: è sufficiente considerare solo i punti a distanza ≤ δ dalla linea L.
- Ordina i punti nella striscia 2δ in base alla loro coordinata y.
- Calcola solo le distanze dei punti entro 7 posizioni nella lista ordinata!



Come trovare la coppia più vicina con un punto per ogni lato?

Def. Sia s_i il punto, nella striscia $2\,\delta$, dalla i-esima più piccola coordinata y.

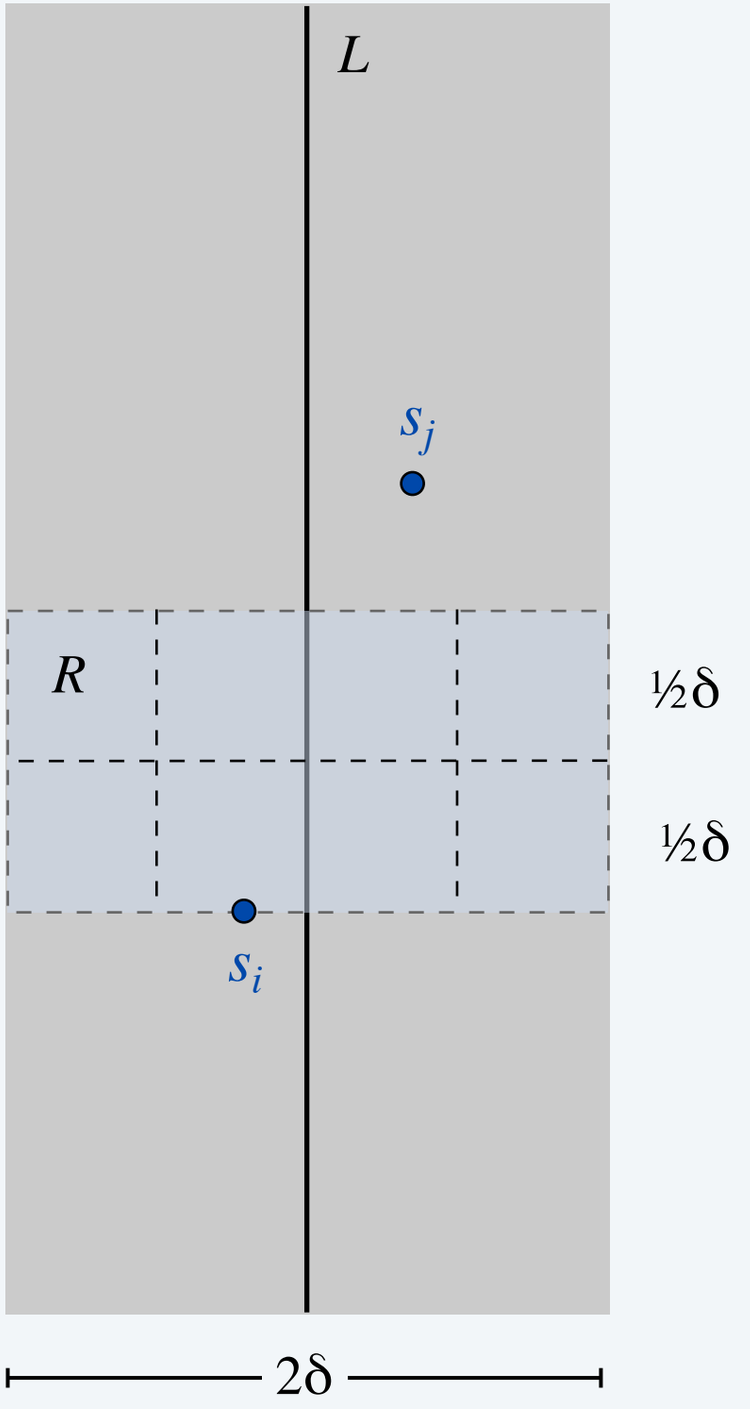
Prop. Se $|j - i| > 7$, allora la distanza tra s_i e s_j è almeno δ.

Dim.

- Considera il rettangolo R 2δ-per-δ nella striscia con coordinata minima y pari a quella di s_i.
- La distanza tra s_i ed un punto s_j sopra R è $\geq \delta$.
- Suddividi R in 8 quadrati.
- Al più 1 punto per quadrato.
- Al più 7 punti oltre ad s_i possono appartenere a R. ▪

il diametro del quadrato è $\delta / \sqrt{2} < \delta$

la costante 7 può essere migliorata tramite un'argomentazione geometrica più sofisticata



Coppia di punti più vicina: algoritmo divide et impera

CLOSEST-PAIR($p_1, p_2, \ldots, p_n$)

Calcola la linea verticale L che lascia metà dei punti su ogni lato della linea. ← $O(n)$

$\delta_1 \leftarrow$ CLOSEST-PAIR(punti nella metà sinistra). ← $T(\lfloor n / 2 \rfloor)$

$\delta_2 \leftarrow$ CLOSEST-PAIR(punti nella metà destra). ← $T(\lceil n / 2 \rceil)$

$\delta \leftarrow \min \{ \delta_1 , \delta_2 \}$. ← $O(n)$

$A \leftarrow$ lista di tutti i punti a distanza $\leq \delta$ dalla linea L. ← $O(n)$

Ordina i punti di A per coordinata y. ← $O(n \log n)$

Scandisci i punti di A per y crescenti e confronta la distanza tra ciascun punto e i 7 successivi nella lista. Se una di queste distanze è minore di δ, aggiorna δ. ← $O(n)$

RETURN δ.

Divide et impera: quiz 6

Qual è la soluzione della seguente ricorrenza?

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n \log n) & \text{if } n > 1 \end{cases}$$

A. $T(n) = \Theta(n)$.

B. $T(n) = \Theta(n \log n)$.

C. $T(n) = \Theta(n \log^2 n)$.

D. $T(n) = \Theta(n^2)$.

Versione migliorata dell'algoritmo per la coppia più vicina

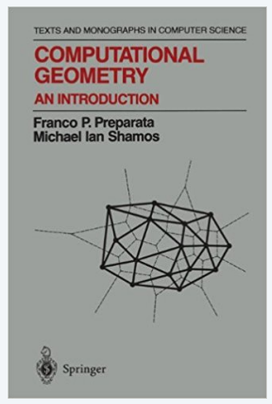
D. Come migliorare il tempo in $O(n \log n)$?

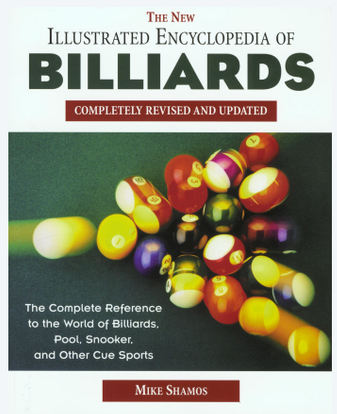
R. Non riordinare da capo i punti nella striscia ogni volta.

- Ogni chiamata ricorsiva restituisce due liste: tutti i punti ordinati per coordinata x, e tutti i punti ordinati per coordinata y.
- Ordina fondendo due liste pre-ordinate.

Teorema. [Shamos 1975] L'algoritmo divide-et-impera per trovare una coppia più vicina nel piano può essere implementato con tempo $O(n \log n)$.

Dim.
$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \Theta(n) & \text{if } n > 1 \end{cases}$$





Divide et impera: quiz 7

Qual è la complessità del problema della coppia più vicina in 2D?

A. $\Theta(n)$.

B. $\Theta(n \log^* n)$.

C. $\Theta(n \log \log n)$.

D. $\Theta(n \log n)$.

E. Neanche Tarjan lo sa.

Complessità computazionale del problema della coppia più vicina

Teorema. [Ben-Or 1983, Yao 1989] Nel modello di alberi di decisione quadratici, ogni algoritmo per la coppia più vicina (anche in 1D) richiede $\Omega(n \log n)$ test quadratici.

$(x_1 - x_2)^2 + (y_1 - y_2)^2$

Lower Bounds for Algebraic Computation Trees

with Integer Inputs*

Andrew Chi-Chih Yao
Department of Computer Science
Princeton University
Princeton, New Jersey 08544

Teorema. [Rabin 1976] Esiste un algoritmo per la coppia più vicina di punti nel piano che ha tempo atteso di esecuzione $O(n)$.

A NOTE ON RABIN'S NEAREST-NEIGHBOR ALGORITHM*

Steve FORTUNE and John HOPCROFT
Department of Computer Science, Cornell University, Ithaca, NY, U.S.A.

Received 20 July 1978, revised version received 21 August 1978

Probabilistic algorithms, nearest neighbor, hashing

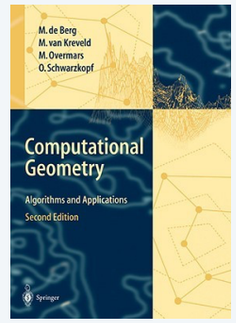
non è soggetto alla limitazione inferiore $\Omega(n \log n)$
perché usa la funzione floor

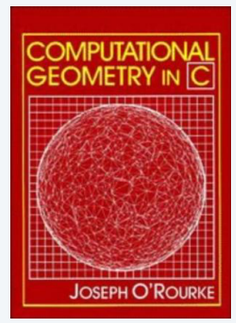
Digressione: geometria computazionale

Algoritmi divide-et-impera ingegnosi per fondamentali problemi geometrici.

problema	brutale	ingegnoso
coppia più vicina	$O(n^2)$	$O(n \log n)$
coppia più lontana	$O(n^2)$	$O(n \log n)$
involucro convesso	$O(n^2)$	$O(n \log n)$
Delaunay/Voronoi	$O(n^4)$	$O(n \log n)$
MST euclideo	$O(n^2)$	$O(n \log n)$

tempo di esecuzione per risolvere un problema 2D con n punti

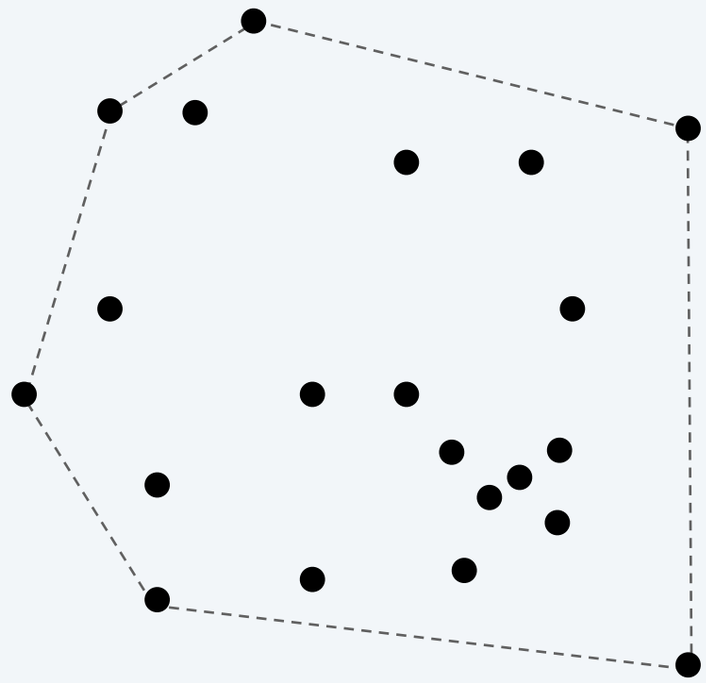




Nota. Dimensione 3D e superiori mettono alla prova i limiti del nostro ingegno.

Involucro convesso

L' involucro convesso di un insieme di n punti è il più piccolo "recinto" convesso che racchiude i punti.

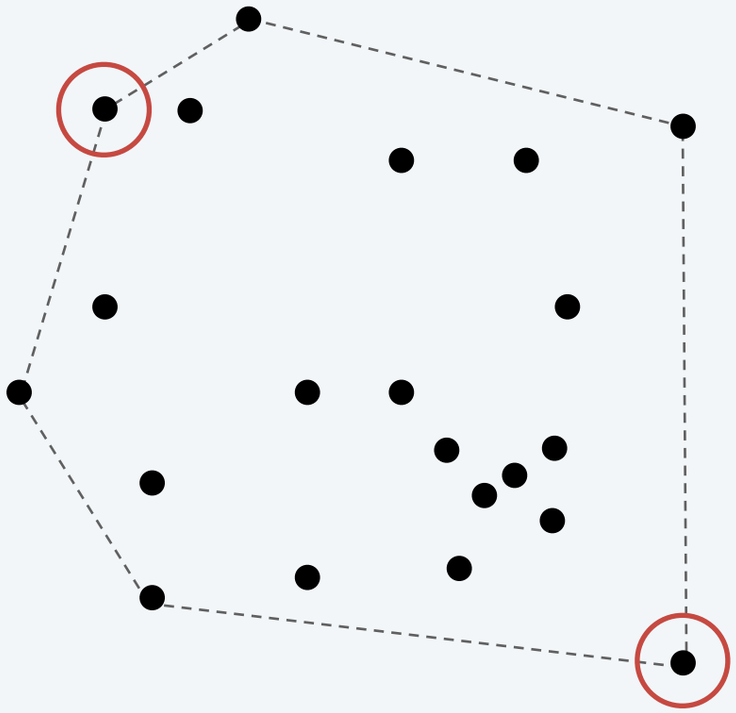


Definizioni equivalenti.

- Poligono convesso ad area minima che racchiude tutti i punti
- Intersezione di tutti gli insiemi convessi che racchiudono tutti i punti.

Coppia più lontana

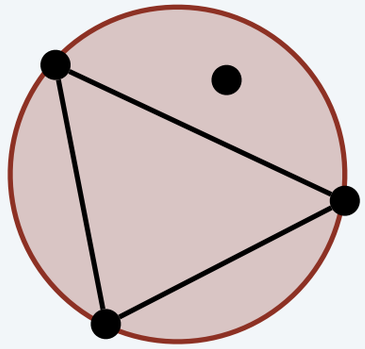
Dati n punti nel piano, trovare una coppia di punti con la massima distanza euclidea tra loro.



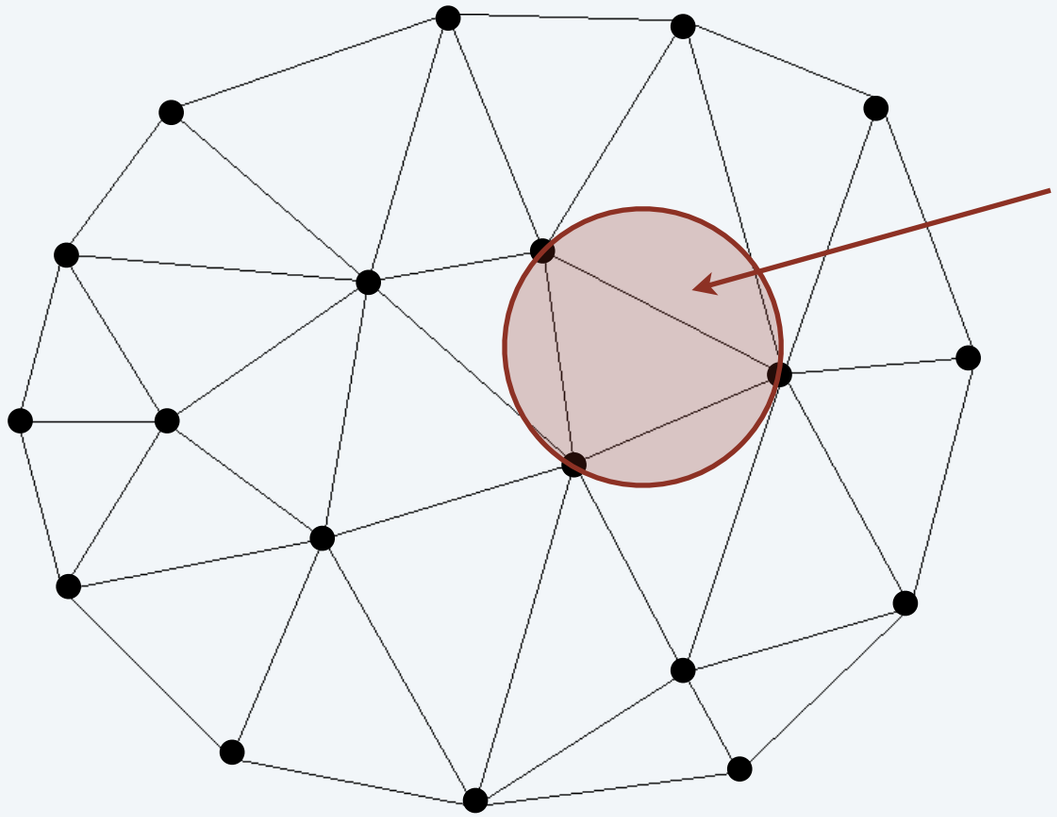
Fatto. I punti della coppia più lontana sono punti estremi dell'involucro convesso.

Triangolazione di Delaunay

La triangolazione di Delaunay è una triangolazione di n punti nel piano tale che nessun punto è dentro il circumcerchio di alcun triangolo.



punto dentro il circumcerchio di 3 punti



Triangolazione di Delaunay di 19 punti

Alcune utili proprietà.

- Nessun arco ne incrocia un altro.
- Tra tutte le triangolazioni, massimizza l'angolo minimo.
- Contiene un arco tra ogni punto ed il punto più vicino ad esso.

MST euclideo

Dati n punti nel piano, trovare un MST che li connette.

[le distanze tra coppie di punti sono distanze euclidee]

Fatto. Il MST euclideo è un sottografo della triangolazione di Delaunay.

Conseguenza. Si può calcolare un MST euclideo in tempo $O(n \log n)$.

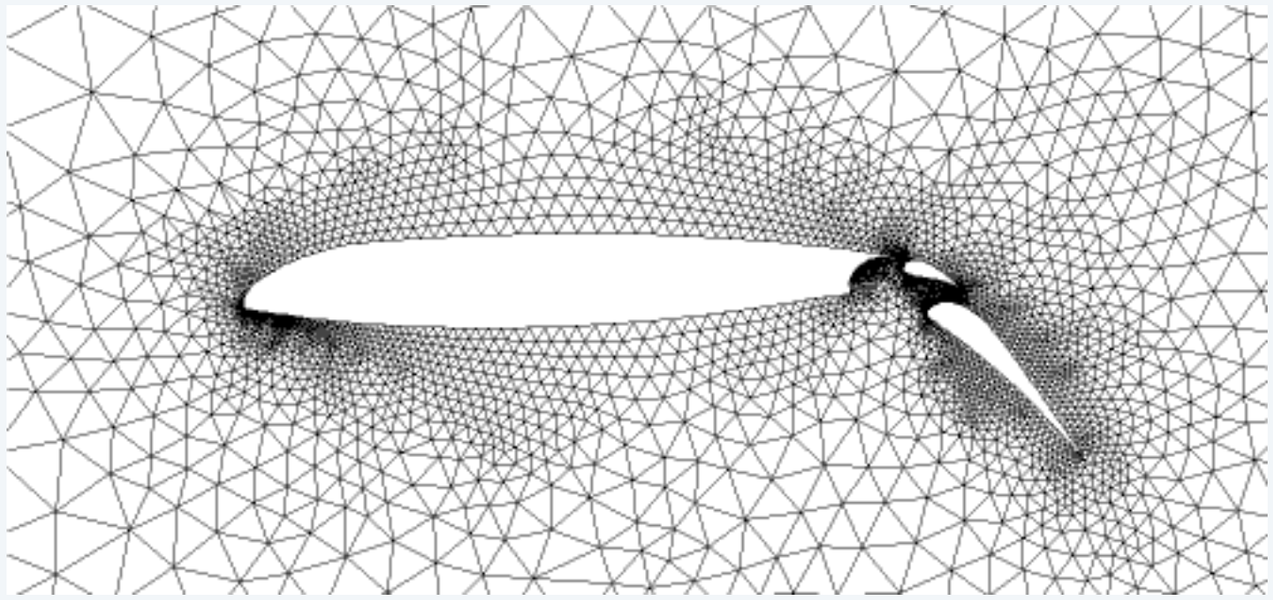
- Calcola una triangolazione di Delaunay.
- Calcola un MST della triangolazione.

← è planare ($\leq 3n$ archi)

Applicazioni della geometria computazionale

Applicazioni.

- Robotica.
- Progetto VLSI.
- Data mining.
- Immagini mediche.
- Visione artificiale.
- Calcolo scientifico.
- Mesh a elementi finiti.
- Simulazioni astronomiche.
- Modelli del mondo fisico.
- Sistemi informativi geografici.
- Computer grafica (film, videogiochi, realtà virtuale).



flusso d'aria intorno ad un'ala di aereoplano

http://www.ics.uci.edu/~eppstein/geom.html