Sorting signed permutations by inversions in $O(n\log n)$ time.

20 maggio 2009

Daniele Bonetta, Paolo Picci

Summary

Introduzione

Cavoli e rape

Brassica Oleracea (cavolo) e Brassica Campestris (rapa): geni mitocondriali per il 99% identici, ma in ordine molto diverso

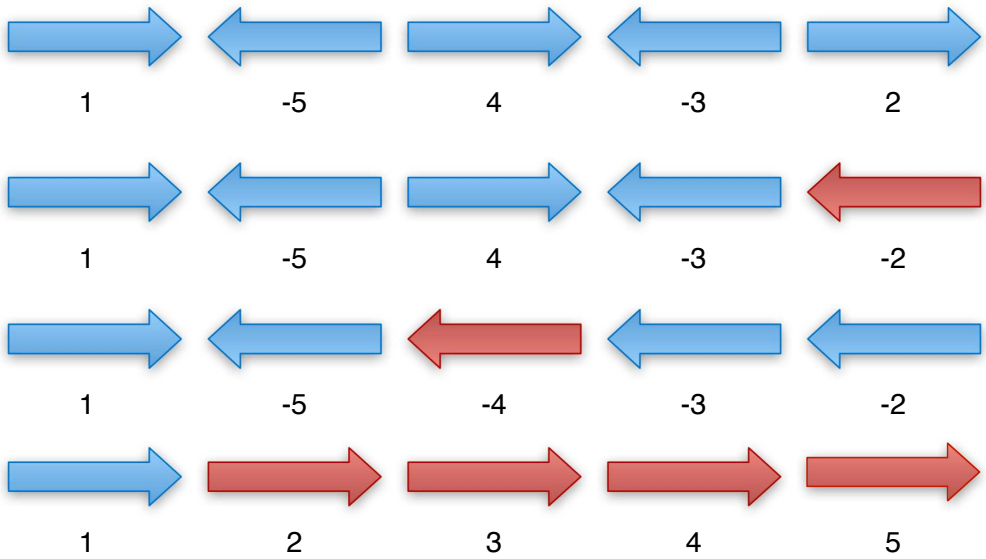
- Si studia come effettuare il riarrangiamento genomico attraverso l'uso di una sola operazione: inversione.

- Si studia come effettuare il riarrangiamento genomico attraverso l'uso di una sola operazione: inversione.
- Dati due genomi si etichettano i geni che compongono il primo con numeri crescenti da 1 a n. Il secondo genoma viene etichettato con una *permutazione* $1, 2, ..., n$

- Si studia come effettuare il riarrangiamento genomico attraverso l'uso di una sola operazione: inversione.
- Dati due genomi si etichettano i geni che compongono il primo con numeri crescenti da 1 a n. Il secondo genoma viene etichettato con una *permutazione* $1, 2, ..., n$
- Dunque un' *inversione* è equivalente al "rovesciamento" di una sequenza di numeri

Riarrangiamento Genomico
Dal Cavolo alla Rapa...

Un semplice esempio di riarrangiamento genomico che conduce alla permutazione identità tramite inversioni:



Terminologia

Permutazione

- Si definisce *permutazione* π la successione $(\pi_1, \pi_2, ..., \pi_n)$ dove ogni elemento π_i è un intero con segno. Tutti gli elementi della permutazione sono distinti, ed Il valore assoluto degli stessi è un elemento dell'insieme $\{1, 2, ..., n\}$.

Permutazione

- Si definisce *permutazione* π la successione $(\pi_1, \pi_2, ..., \pi_n)$ dove ogni elemento π_i è un intero con segno. Tutti gli elementi della permutazione sono distinti, ed Il valore assoluto degli stessi è un elemento dell'insieme $\{1, 2, ..., n\}$.
- Per convenzione ogni permutazione di n elementi è compresa tra gli elementi 0 e $n + 1$, e si considerano permutazioni lineari.

Permutazione

- Si definisce *permutazione* π la successione $(\pi_1, \pi_2, ..., \pi_n)$ dove ogni elemento π_i è un intero con segno. Tutti gli elementi della permutazione sono distinti, ed Il valore assoluto degli stessi è un elemento dell'insieme $\{1, 2, ..., n\}$.
- Per convenzione ogni permutazione di n elementi è compresa tra gli elementi 0 e $n + 1$, e si considerano permutazioni lineari.
- (Per ogni permutazione lineare di lunghezza n esistono $n + 1$ permutazioni circolari di lunghezza $n + 1$, ciascuna delle quali è equivalente per sequenze di inversioni di ordinamento)

Inversione

Un'*inversione* $\rho(i,j)$ si applica alla permutazione

$$\pi = (\pi_1, ..., \pi_i, ..., \pi_j, ..., \pi_n)$$

invertendo tutti gli elementi tra i e j, cambiando il loro segno:

$$\pi = (\pi_1, ..., \pi_{i-1}, -\pi_j, ..., -\pi_i, \pi_{j+1}, ..., \pi_n)$$

Elementi adiacenti

- Adiacenze:
  - Due elementi adiacenti π_i e π_{i+1} con $0 \leq i \leq n+1$ formano una *adiacenza*

Elementi adiacenti

- Adiacenze:
  - Due elementi adiacenti π_i e π_{i+1} con $0 \leq i \leq n+1$ formano una *adiacenza*
  - Una adiacenza si dice *non-Breakpoint* sse $\pi_{i-1} - \pi_i = 1$

Elementi adiacenti

- Adiacenze:
  - Due elementi adiacenti π_i e π_{i+1} con $0 \leq i \leq n+1$ formano una *adiacenza*
  - Una adiacenza si dice *non-Breakpoint* sse $\pi_{i-1} - \pi_i = 1$
  - Altrimenti è detta *Breakpoint*

Elementi adiacenti

- Adiacenze:
  - Due elementi adiacenti π_i e π_{i+1} con $0 \leq i \leq n+1$ formano una *adiacenza*
  - Una adiacenza si dice *non-Breakpoint* sse $\pi_{i-1} - \pi_i = 1$
  - Altrimenti è detta *Breakpoint*
- Una coppia (π_i, π_j) in una permutazione si dice *orientata* se è composta da due elementi π_i e π_j di segno opposto tali che: $\pi_i + \pi_j = \pm 1$

Elementi adiacenti

- Adiacenze:
  - Due elementi adiacenti π_i e π_{i+1} con $0 \leq i \leq n+1$ formano una *adiacenza*
  - Una adiacenza si dice *non-Breakpoint* sse $\pi_{i-1} - \pi_i = 1$
  - Altrimenti è detta *Breakpoint*
- Una coppia (π_i, π_j) in una permutazione si dice *orientata* se è composta da due elementi π_i e π_j di segno opposto tali che: $\pi_i + \pi_j = \pm 1$
- L'inversioe prodotta da una coppia orientata (chiamata *inversione orientata*) è dunque:
  - $\rho(i, j-1)$ per $\pi_i + \pi_j = 1$
  - $\rho(i+1, j)$ per $\pi_i + \pi_j = -1$

Elementi adiacenti

- Adiacenze:
  - Due elementi adiacenti π_i e π_{i+1} con $0 \leq i \leq n+1$ formano una *adiacenza*
  - Una adiacenza si dice *non-Breakpoint* sse $\pi_{i-1} - \pi_i = 1$
  - Altrimenti è detta *Breakpoint*
- Una coppia (π_i, π_j) in una permutazione si dice *orientata* se è composta da due elementi π_i e π_j di segno opposto tali che: $\pi_i + \pi_j = \pm 1$
- L'inversioe prodotta da una coppia orientata (chiamata *inversione orientata*) è dunque:
  - $\rho(i, j-1)$ per $\pi_i + \pi_j = 1$
  - $\rho(i+1, j)$ per $\pi_i + \pi_j = -1$
- Un'inversione orientata genera sempre un non-Breakpoint.

Elementi adiacenti
Esempio

- Data la permutazione $(4\ 5\ -3\ 1\ -6\ 2\ -7)$

Elementi adiacenti
Esempio

- Data la permutazione $(4\ 5\ -3\ 1\ -6\ 2\ -7)$
- Alcune adiacenze sono $(4,5)$, $(-3,1)$ e $(2,7)$

Elementi adiacenti
Esempio

- Data la permutazione $(4\ 5\ -3\ 1\ -6\ 2\ -7)$
- Alcune adiacenze sono $(4,5)$, $(-3,1)$ e $(2,7)$
- Una breakpoint è $(-3,1)$ e una non breakpoint è $(4,5)$

Elementi adiacenti
Esempio

- Data la permutazione $(4\ 5\ -3\ 1\ -6\ 2\ -7)$
- Alcune adiacenze sono $(4,5)$, $(-3,1)$ e $(2,7)$
- Una breakpoint è $(-3,1)$ e una non breakpoint è $(4,5)$
- Una coppia orientata è $(2,-3)$

Elementi adiacenti
Esempio

- Data la permutazione $(4\ 5\ -3\ 1\ -6\ 2\ -7)$
- Alcune adiacenze sono $(4,5)$, $(-3,1)$ e $(2,7)$
- Una breakpoint è $(-3,1)$ e una non breakpoint è $(4,5)$
- Una coppia orientata è $(2,-3)$
- Che corrisponde all'inversione $\rho(i+1,j)=(4,6)$

FCI
Definizione

Si definisce *Frame Control Interval* di una permutazione di lunghezza n una sua sottostringa $(a, s_1, s_2, ..., s_k, b)$ o $(-b, s_1, s_2, ..., s_k, -a)$, tale che:

- Per ogni i compresa in $1 \leq i \leq k$, si ha $|a| < |s_i| < |b|$

FCI
Definizione

Si definisce *Frame Control Interval* di una permutazione di lunghezza n una sua sottostringa $(a, s_1, s_2, ..., s_k, b)$ o $(-b, s_1, s_2, ..., s_k, -a)$, tale che:

- Per ogni i compresa in $1 \leq i \leq k$, si ha $|a| < |s_i| < |b|$
- Per ogni l tale che $|a| < l < |b|$ esiste un j compreso in $1 \leq j \leq k$ con $|s_j| = l$

FCI
Definizione

Si definisce *Frame Control Interval* di una permutazione di lunghezza n una sua sottostringa $(a, s_1, s_2, ..., s_k, b)$ o $(-b, s_1, s_2, ..., s_k, -a)$, tale che:

- Per ogni i compresa in $1 \leq i \leq k$, si ha $|a| < |s_i| < |b|$
- Per ogni l tale che $|a| < l < |b|$ esiste un j compreso in $1 \leq j \leq k$ con $|s_j| = l$
- E l'FCI non sia un'unione di intervalli più corti con le proprietà di cui sopra.

FCI
Esempio

Ad esempio la permutazione $(2\ 1\ 3\ 5\ -4\ 6\ 8\ 7)$ ha un FCI con elementi di bordo 0 e 9, e ha un FCI annidato con elementi di bordo 3 e 6.

Component

- Un *componente* è composto da tutti gli elementi interni ad un FCI che non sono all'interno di alcun sottointervallo, più gli elementi di bordo di un FCI (dunque un componente ha almeno 4 elementi).
- Un componente *bad* è un componente dove tutti gli elementi hanno lo stesso segno.

Component

- Un *componente* è composto da tutti gli elementi interni ad un FCI che non sono all'interno di alcun sottointervallo, più gli elementi di bordo di un FCI (dunque un componente ha almeno 4 elementi).
- Un componente *bad* è un componente dove tutti gli elementi hanno lo stesso segno.
- Ad esempio la permutazione (2 3 6 7 4 5 8 9 10 1) ha un componente *bad* compreso tra gli elementi $(3, 8)$.

Inversioni safe & unsafe

- Un' *inversione* è detta *unsafe* se genera bad components

Inversioni safe & unsafe

- Un' *inversione* è detta *unsafe* se genera bad components
- Altrimenti è detta *safe* (ovviamente...)

Inversioni safe & unsafe

- Un' *inversione* è detta *unsafe* se genera bad components
- Altrimenti è detta *safe* (ovviamente...)
- Una permutazione è *positiva* sse non è la permutazione Identità, e ha tutti gli elementi positivi.

Inversioni safe & unsafe

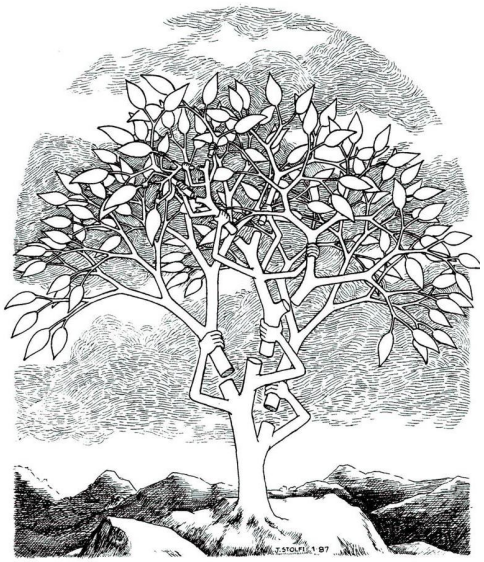
- Un' *inversione* è detta *unsafe* se genera bad components
- Altrimenti è detta *safe* (ovviamente...)
- Una permutazione è *positiva* sse non è la permutazione Identità, e ha tutti gli elementi positivi.
- Una permutazione positiva garantisce che esista al suo interno almeno un componente bad
- Una permutazione contenente componenti bad può essere trasformata in un'altra che non contiene alcun componente bad in tempo lineare

Assunzione fondamentale

Assunzione fondamentale

Si suppone che nella permutazione da ordinare (in input) non esistano componenti bad.

Strutture dati



Splay tree
Proprieta fondamentali

Uno **Splay Tree** è una struttura dati ad albero che migliora le prestazioni di un normale BST.

- Non è bilanciato inizialmente

Splay tree
Proprieta fondamentali

Uno **Splay Tree** è una struttura dati ad albero che migliora le prestazioni di un normale BST.

- Non è bilanciato inizialmente
- Le singole operazioni non sono efficienti (sono $O(n)$)

Splay tree
Proprieta fondamentali

Uno **Splay Tree** è una struttura dati ad albero che migliora le prestazioni di un normale BST.

- Non è bilanciato inizialmente
- Le singole operazioni non sono efficienti (sono $O(n)$)
- Una serie di m operazioni per un albero con n nodi richiede $O(m \log n)$ (con $m \geq n$).

Splay tree
Operazione di Splay

Quando un nodo x è acceduto, si effettua un'operazione di Splay sulla radice per muovere x. Per effettuare l'operazione possono essere messi in atto tre differenti rotazioni in funzione dei seguenti fattori:

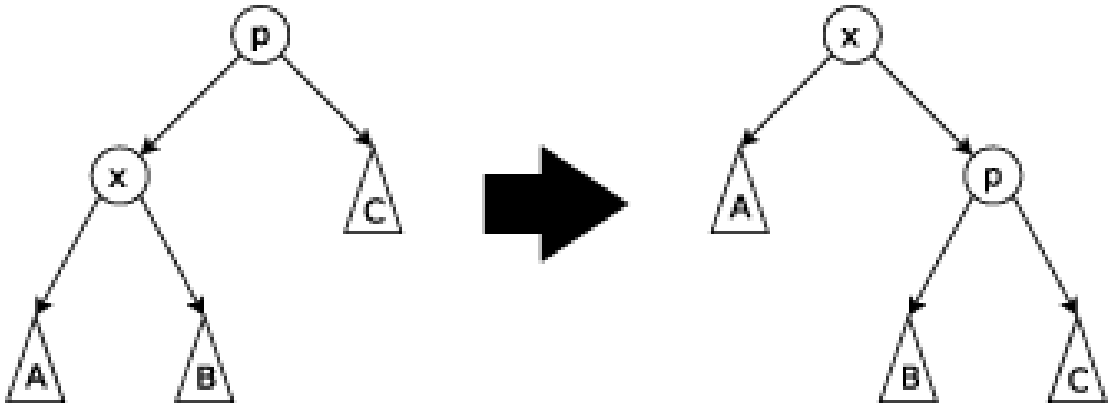
- Se x sia a sinistra o destra del nodo radice p
- Se p sia la radice, e se non:
- Se p sia o meno il figlio destro o sinistro del suo rispettivo padre g

Queste operazioni vanno sotto il nome di:

- Singolo **Zig**
- **ZigZag**
- **ZigZig**

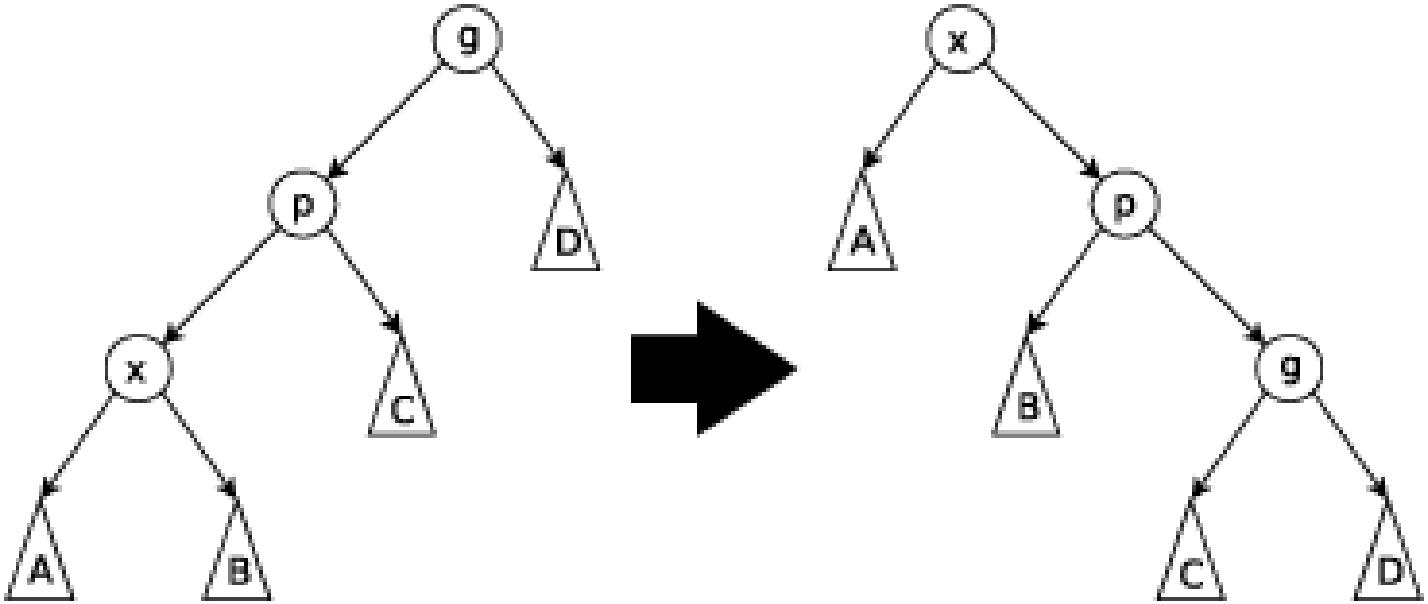
Splay tree
rotazione Zig

Se p è la radice di x si applica una rotazione *singola*.



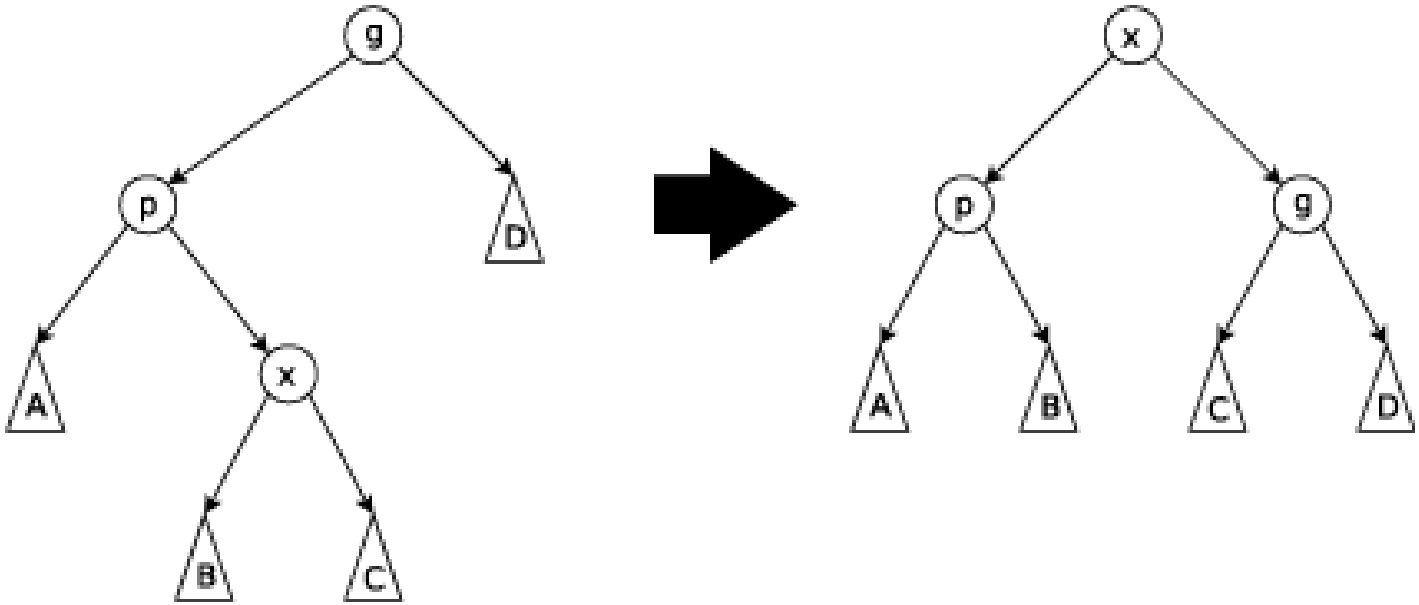
Splay tree
rotazione ZigZig

Se p non è la radice di x e sia x che p sono figli destri o entrambi figli sinistri



Splay tree
rotazione ZigZag

Se p non è la radice di x e x è figlio destro mentre p è figlio sinistro, o vice versa.



Albero Splay adottato

Per gestire le inversioni su una permutazione si usa uno Splay Tree che contenga la permutazione, con l'aggiunta di un **flag booleano** per ogni nodo. Con questo approccio si ottiene:

- I nodi rappresentano gli elementi della permutazione

Albero Splay adottato

Per gestire le inversioni su una permutazione si usa uno Splay Tree che contenga la permutazione, con l'aggiunta di un **flag booleano** per ogni nodo. Con questo approccio si ottiene:

- I nodi rappresentano gli elementi della permutazione
- La permutazione è memorizzata in spazio lineare

Albero Splay adottato

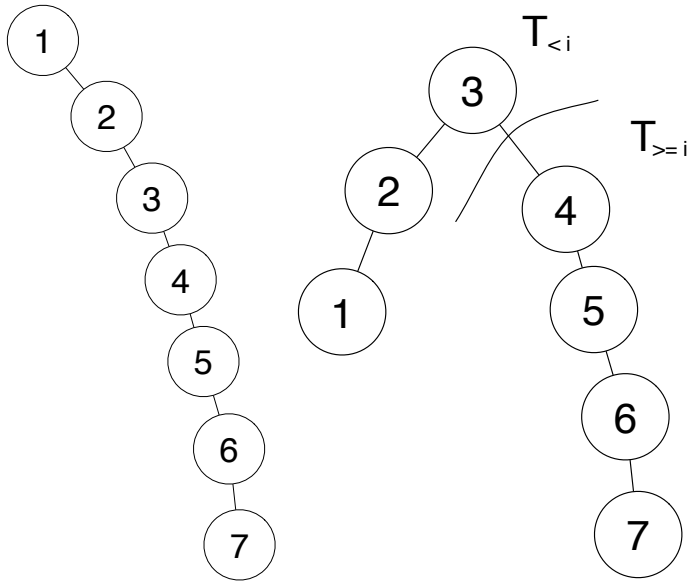
Per gestire le inversioni su una permutazione si usa uno Splay Tree che contenga la permutazione, con l'aggiunta di un **flag booleano** per ogni nodo. Con questo approccio si ottiene:

- I nodi rappresentano gli elementi della permutazione
- La permutazione è memorizzata in spazio lineare
- Un'inversione può essere fatta con tempo logaritmico grazie al fatto che le operazioni di Splay sono $O(n \log n)$

Splay Tree: Inversione
Primo splay

Per effettuare un'inversione $\rho(i,j)$ tra gli elementi di indice i e j (compresi) si eseguono queste operazioni:

- Si effettua uno splay sul nodo di indice $i-1$

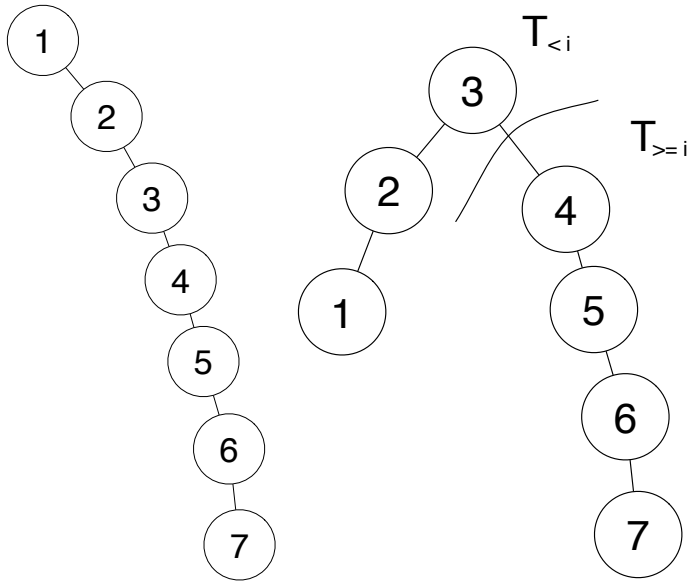


Inversione di esempio $\rho(4,6)$

Splay Tree: Inversione
Primo splay

Per effettuare un'inversione $\rho(i,j)$ tra gli elementi di indice i e j (compresi) si eseguono queste operazioni:

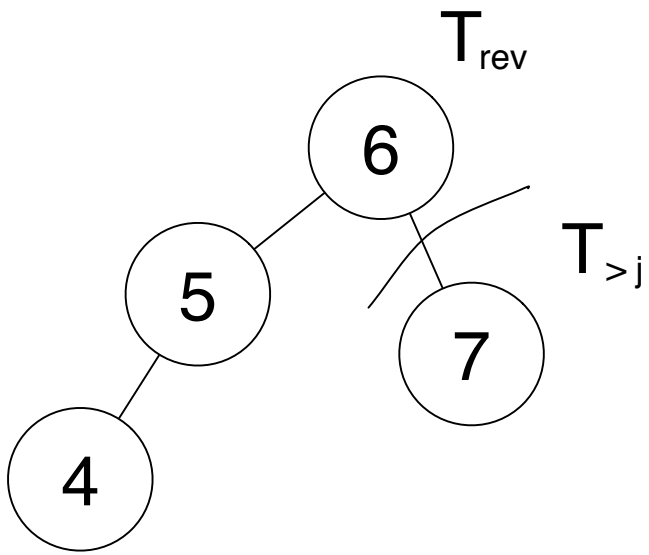
- Si effettua uno splay sul nodo di indice $i-1$
- Il sottoalbero alla destra del nodo è diviso alla radice restituendo i due sottoalberi $T_{<i}$ e $T_{\geq i}$



Inversione di esempio $\rho(4,6)$

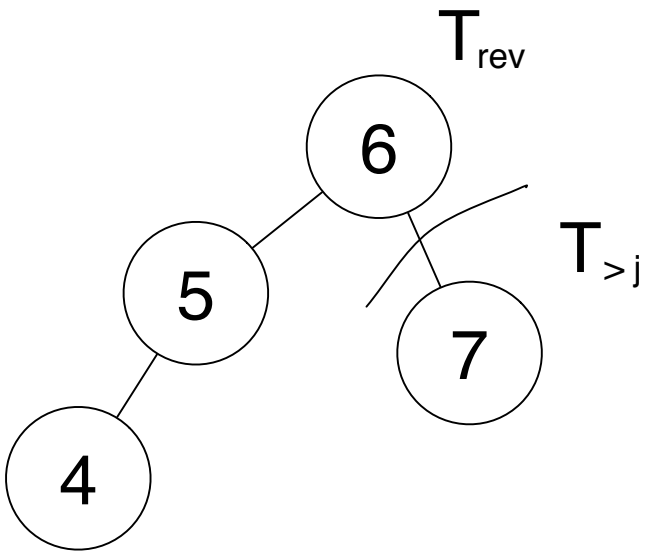
Splay Tree: Inversione
Secondo splay

- Si effettua un secondo splay sull'indice j del sottoalbero $T_{<i}$



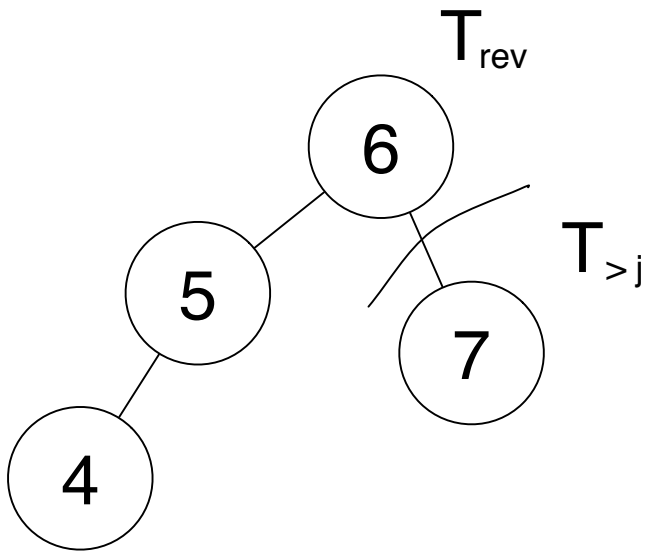
Splay Tree: Inversione
Secondo splay

- Si effettua un secondo splay sull'indice j del sottoalbero $T_{<i}$
- Il sottoalbero alla destra del nodo viene ancora diviso alla radice restituendo i sottoalberi T_{rev} e $T_{>j}$ (T_{rev} conterrà tutti gli elementi da invertire per ρ)



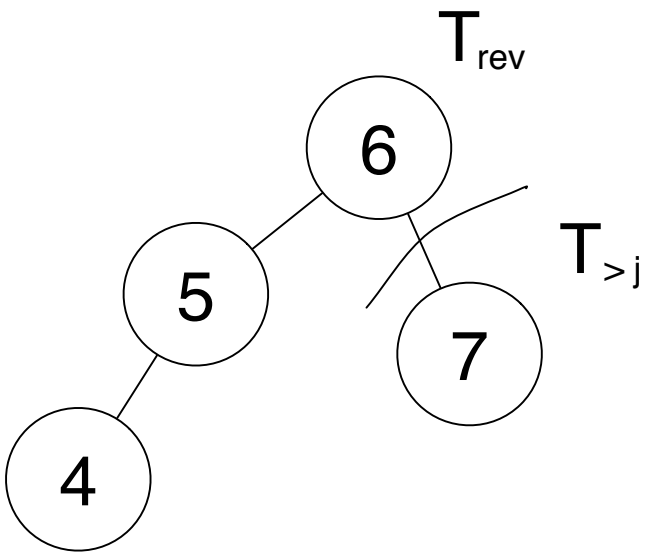
Splay Tree: Inversione
Secondo splay

- Si effettua un secondo splay sull'indice j del sottoalbero $T_{<i}$
- Il sottoalbero alla destra del nodo viene ancora diviso alla radice restituendo i sottoalberi T_{rev} e $T_{>j}$ (T_{rev} conterrà tutti gli elementi da invertire per ρ)
- Si ottengono così i sottoalberi T_{rev}, $T_{<i}$ e $T_{>j}$



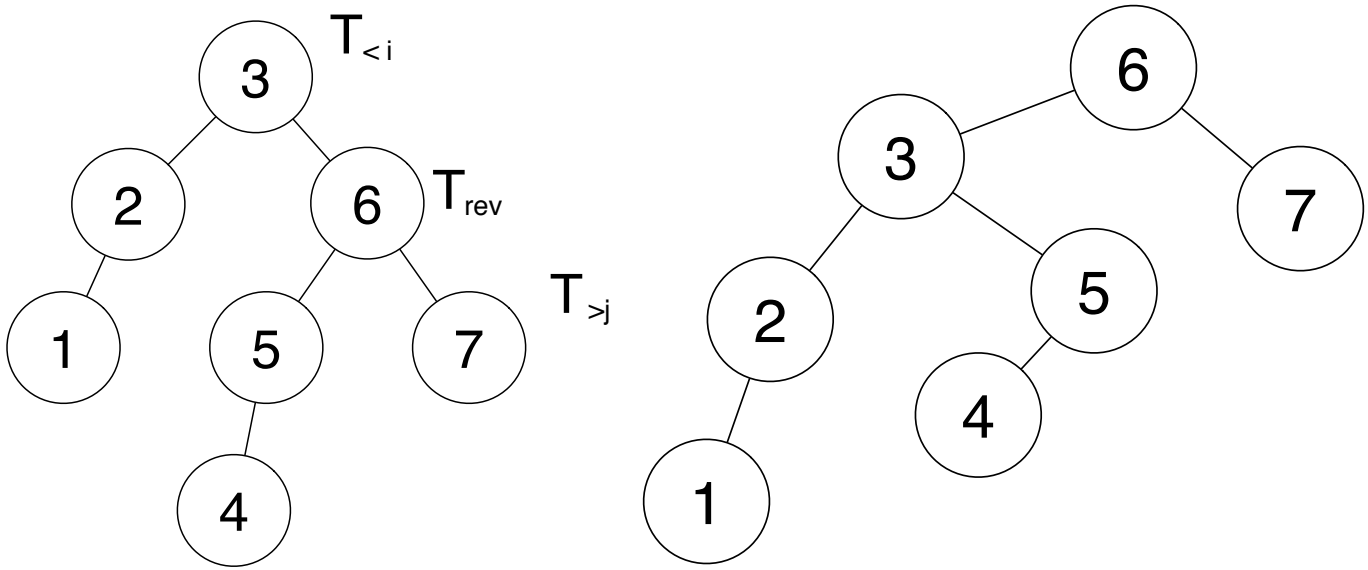
Splay Tree: Inversione
Secondo splay

- Si effettua un secondo splay sull'indice j del sottoalbero $T_{<i}$
- Il sottoalbero alla destra del nodo viene ancora diviso alla radice restituendo i sottoalberi T_{rev} e $T_{>j}$ (T_{rev} conterrà tutti gli elementi da invertire per ρ)
- Si ottengono così i sottoalberi T_{rev}, $T_{<i}$ e $T_{>j}$
- Si marca il flag della radice di T_{rev}



Splay Tree: Inversione
Terzo splay

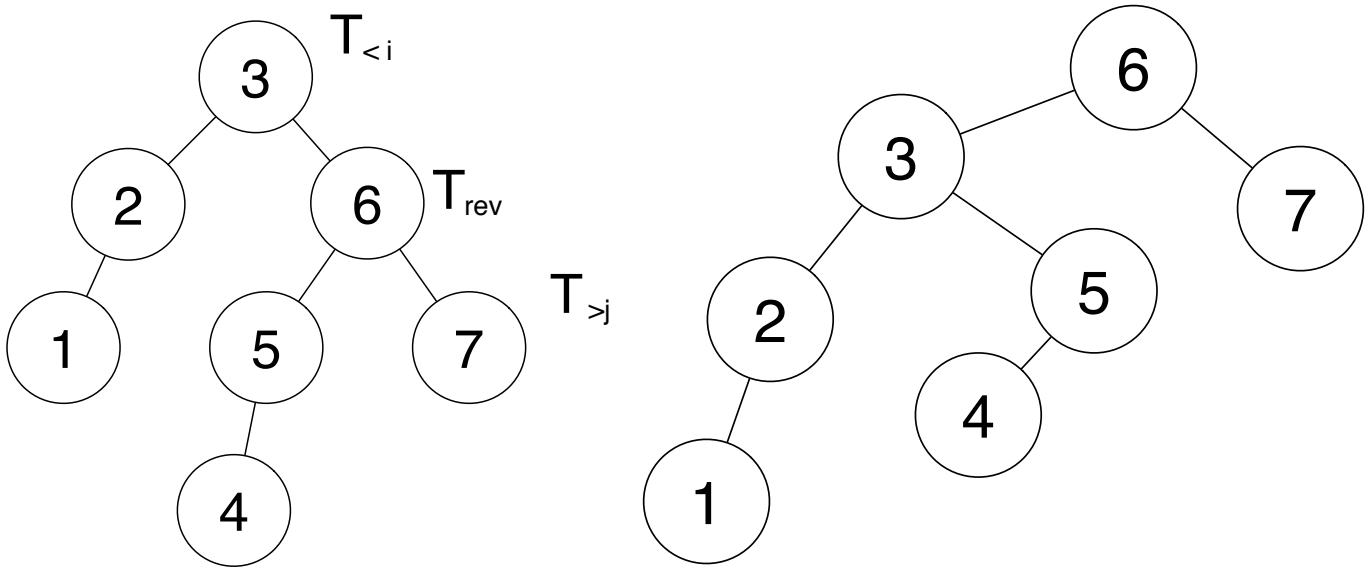
- Si uniscono T_{rev} e $T_{<i}$, così da ottenere $T_{\leq j}$



Inversione di esempio $\rho(4, 6)$

Splay Tree: Inversione
Terzo splay

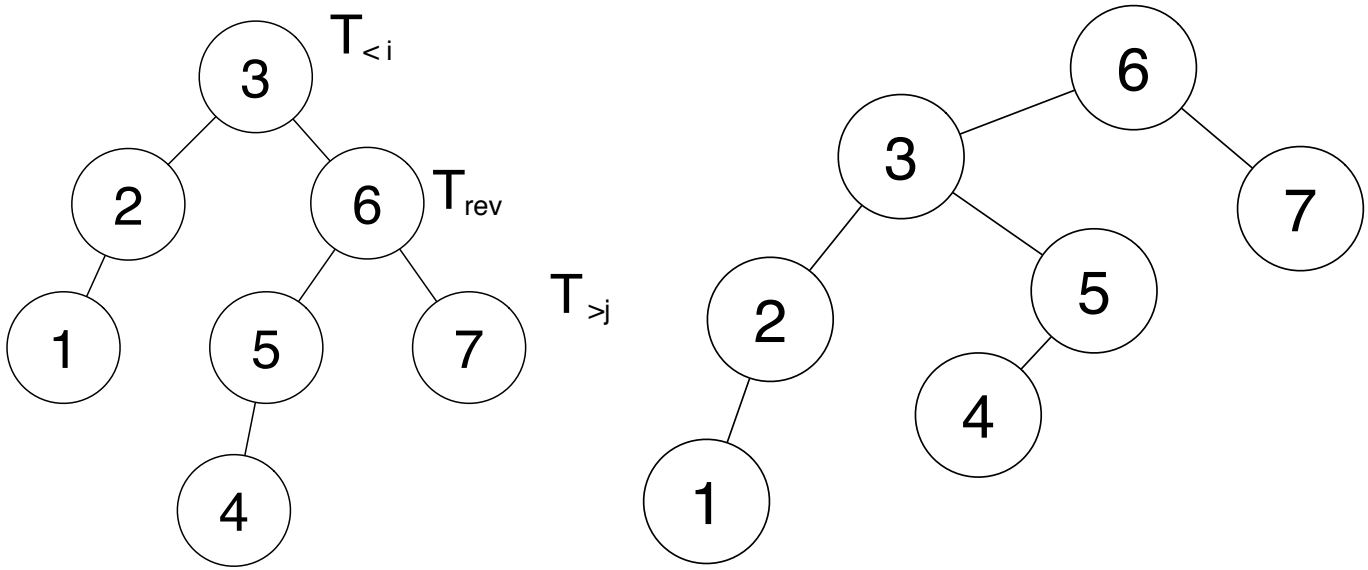
- Si uniscono T_{rev} e $T_{<i}$, così da ottenere $T_{\leq j}$
- Si effettua un terzo splay in j per il sottoalbero $T_{\leq j}$



Inversione di esempio $\rho(4, 6)$

Splay Tree: Inversione
Terzo splay

- Si uniscono T_{rev} e $T_{<i}$, così da ottenere $T_{\leq j}$
- Si effettua un terzo splay in j per il sottoalbero $T_{\leq j}$
- Si uniscono poi $T_{\leq j}$ e $T_{>j}$ facendo in modo che $T_{>j}$ diventi il figlio destro della radice di $T_{\leq j}$



Inversione di esempio $\rho(4, 6)$

Splay Tree
Approccio *Lazy*

- Come visto, nel sottoalbero T_{rev} sono presenti tutti e soli gli elementi coinvolti nell'inversione

Splay Tree
Approccio *Lazy*

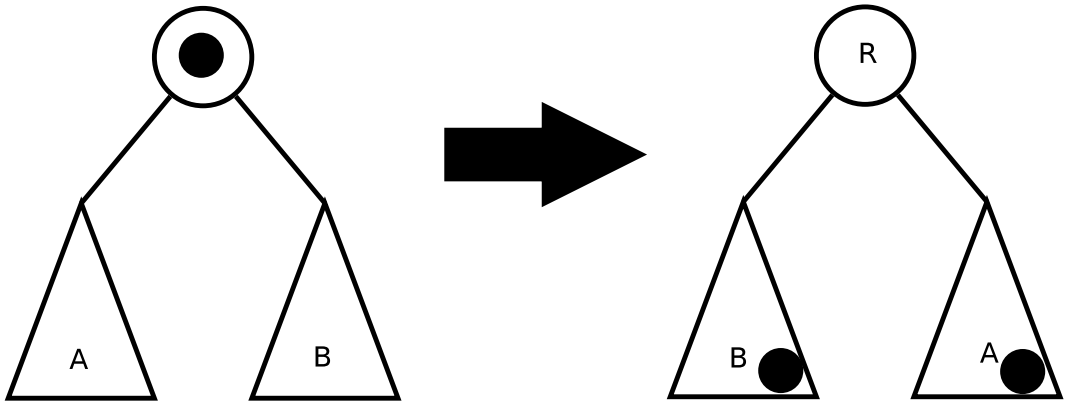
- Come visto, nel sottoalbero T_{rev} sono presenti tutti e soli gli elementi coinvolti nell'inversione
- Effettuare l'inversione avrebbe un costo computazionale troppo elevato

Splay Tree
Approccio *Lazy*

- Come visto, nel sottoalbero T_{rev} sono presenti tutti e soli gli elementi coinvolti nell'inversione
- Effettuare l'inversione avrebbe un costo computazionale troppo elevato
- Si usa dunque un approccio *Lazy* così da ridurre la complessita in tempo: i figli di un sottalbero radicato in un nodo marcato con il flag sono da considerarsi invertiti, e dunque i loro indici saranno da ricalcolare

Splay Tree
Propagazione dei flag: push

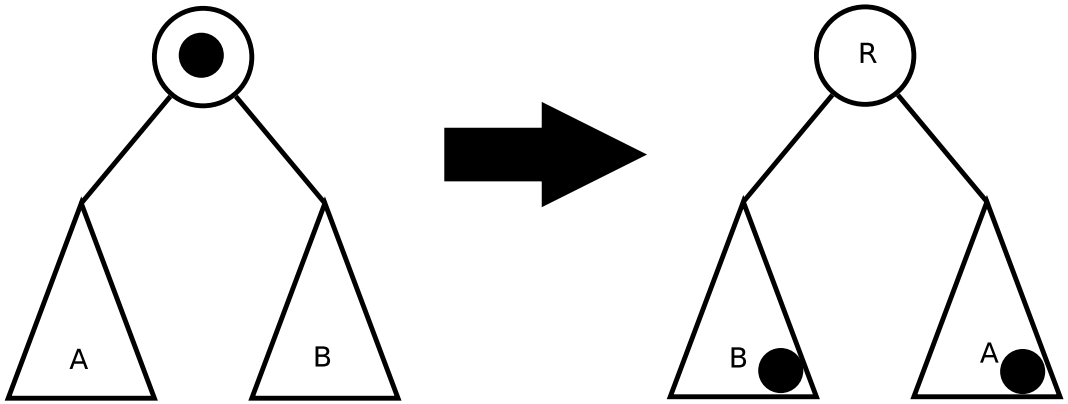
Esiste la possibilit di effettuare un'operazione per *spostare* i flag di uno splay tree propagandoli fino alle foglie: push.



- Si annulla il flag del nodo con il flag da azzerare

Splay Tree
Propagazione dei flag: push

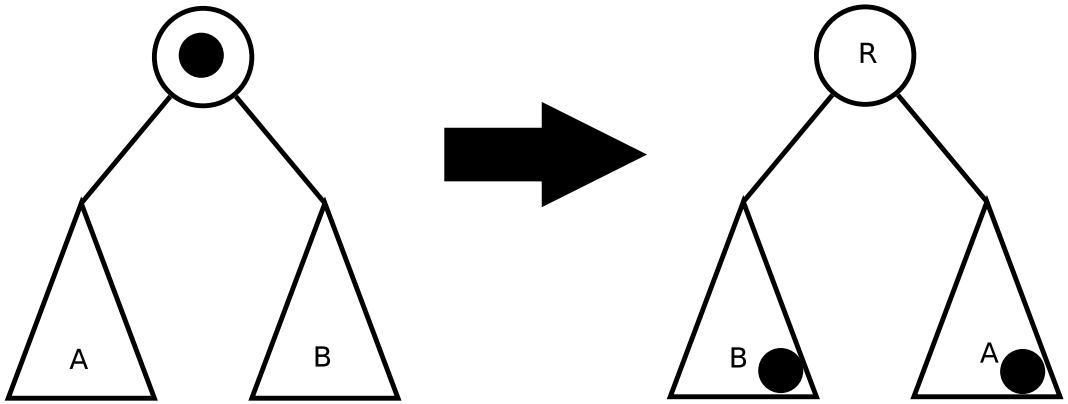
Esiste la possibilit di effettuare un'operazione per *spostare* i flag di uno splay tree propagandoli fino alle foglie: push.



- Si annulla il flag del nodo con il flag da azzerare
- Si scambiano ed invertono i suoi sottoalberi destro e sinistro

Splay Tree
Propagazione dei flag: push

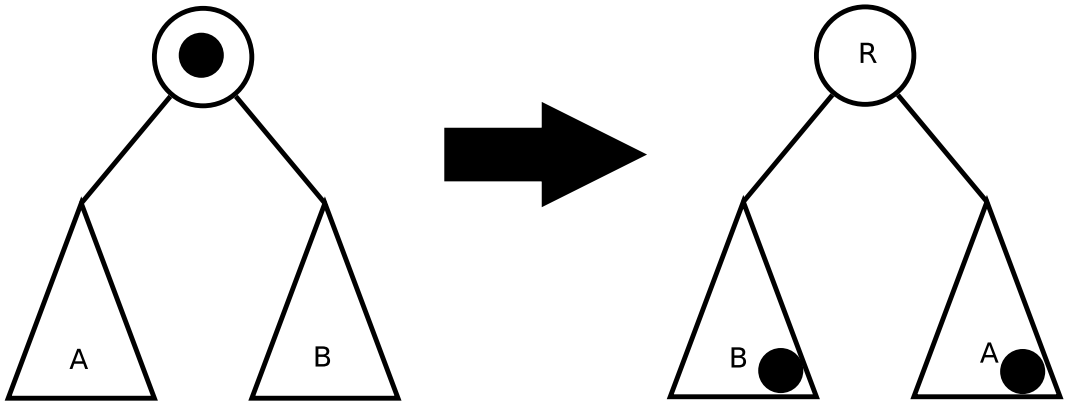
Esiste la possibilit di effettuare un'operazione per *spostare* i flag di uno splay tree propagandoli fino alle foglie: push.



- Si annulla il flag del nodo con il flag da azzerare
- Si scambiano ed invertono i suoi sottoalberi destro e sinistro
- Si marcano i flag dei due figli e si cambia il segno degli elementi nel nodo

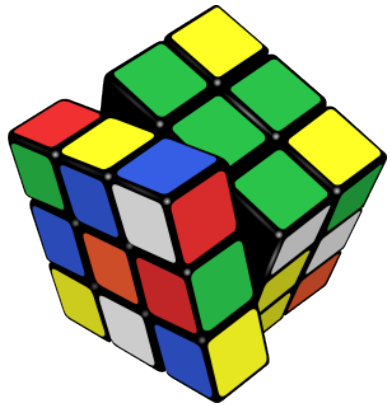
Splay Tree
Propagazione dei flag: push

Esiste la possibilit di effettuare un'operazione per *spostare* i flag di uno splay tree propagandoli fino alle foglie: push.



- Si annulla il flag del nodo con il flag da azzerare
- Si scambiano ed invertono i suoi sottoalberi destro e sinistro
- Si marcano i flag dei due figli e si cambia il segno degli elementi nel nodo
- Se il nodo e una foglia, si inverte semplicemente il segno

Algoritmi



Approccio
Come raggiungere il mito di ($n\log n$)?

Invece di affrontare il problema dal punto di vista della struttura dati (inventandone una che riesca ad indicare quale tra tutte le possibili inversioni sia la migliore, magari in tempo logaritmico) l'approccio adottato è quello di considerare la questione principale senza girarci intorno: *identificare* la miglior inversione orientata e tenere traccia solo di quella.

contributo chiave dell'articolo

Our key contribution is that we don't need to mantain informations about *all* oriented inversion for every permutation at each sorting step -a few suffice in most cases [...] We maintain information about only the MAX inversions in the data structure

Coppia e inversione MAX

Coppia e inversione MAX

Sia (π_i, π_j) una coppia orientata in una permutazione, e sia π_j l'elemento negativo tra i due. L'inversione orientata corrispondente a (π_i, π_j) è una inversione MAX sse π_j ha il massimo valore tra tutti gli elementi negativi nella permutazione. La coppia (π_i, π_j) è detta coppia MAX della permutazione

Ad esempio, per la permutazione $(4\ 5\ -3\ 1\ -6\ 2\ -7)$ l'inversione MAX è data da $\rho(4, 6)$, che corrisponde alla coppia orientata $(2, -3)$.

Algoritmo MAX

Nella struttura dati si tiene traccia solo delle inversioni MAX, e si effettuano delle inversioni solo in corrispondenza delle stesse. Il risultato è l'algoritmo MAX:

while(la permutazione contiene almeno un elemento negativo)
do
  Trova l'indice dell'elemento negativo massimo π_j
  Trova l'indice di $\pi_i = |\pi_j| - 1$
  Effettua l'inversione corrispondente alla coppia (π_i, π_j)
end

Algoritmo MAX
Esempio

while (esiste un elemento negativo nella permutazione) {
    Trova l'indice π_j massimo negativo
    Trova l'indice $\pi_i = |\pi_j| - 1$
    Inverti la coppia orientata (π_i, π_j)
}

$$0\ (\ 2\ 3\ {-1}\ 4\ )\ 5$$

Algoritmo MAX
Esempio

while (esiste un elemento negativo nella permutazione) {
    Trova l'indice π_j massimo negativo
    Trova l'indice $\pi_i = |\pi_j| - 1$
    Inverti la coppia orientata (π_i, π_j)
}

$$0\ (\ 2\ 3\ -1\ 4\ )\ 5$$

- π_j = -1 j = 3

Algoritmo MAX
Esempio

while (esiste un elemento negativo nella permutazione) {
    Trova l'indice π_j massimo negativo
    Trova l'indice $\pi_i = |\pi_j| - 1$
    Inverti la coppia orientata (π_i, π_j)
}

$$0\ (\ 2\ 3\ -1\ 4\ )\ 5$$

- $\pi_j = -1$ j $= 3$
- $\pi_i = 0$ i $= 0$

Algoritmo MAX
Esempio

while (esiste un elemento negativo nella permutazione) {
    Trova l'indice π_j massimo negativo
    Trova l'indice $\pi_i = |\pi_j| - 1$
    Inverti la coppia orientata (π_i, π_j)
}

0 (2 3 -1 4) 5

- π_j = -1 j = 3
- π_i = 0 i = 0
- Coppia orientata (0, -1) $\rho(1, 3)$

Algoritmo MAX
Esempio

while (esiste un elemento negativo nella permutazione) {
    Trova l'indice π_j massimo negativo
    Trova l'indice $\pi_i = |\pi_j| - 1$
    Inverti la coppia orientata (π_i, π_j)
}

$$0\ (\ 1\ -3\ -2\ 4\ )\ 5$$

Algoritmo MAX
Esempio

while (esiste un elemento negativo nella permutazione) {
    Trova l'indice π_j massimo negativo
    Trova l'indice $\pi_i = |\pi_j| - 1$
    Inverti la coppia orientata (π_i, π_j)
}

$$0\ (\ 1\ -3\ -2\ 4\ )\ 5$$

- $\pi_j =$ -2 j = 3

Algoritmo MAX
Esempio

while (esiste un elemento negativo nella permutazione) {
    Trova l'indice π_j massimo negativo
    Trova l'indice $\pi_i = |\pi_j| - 1$
    Inverti la coppia orientata (π_i, π_j)
}

$$0\ (\ 1\ -3\ -2\ 4\ )\ 5$$

- $\pi_j = -2$ j $= 3$
- $\pi_i = 1$ i $= 1$

Algoritmo MAX
Esempio

while (esiste un elemento negativo nella permutazione) {
    Trova l'indice π_j massimo negativo
    Trova l'indice $\pi_i = |\pi_j| - 1$
    Inverti la coppia orientata (π_i, π_j)
}

$$0\ (\ 1\ -3\ -2\ 4\ )\ 5$$

- π_j = -2 j = 3
- π_i = 1 i = 1
- Coppia orientata (1, -2) $\rho(2, 3)$

Algoritmo MAX
Esempio

while (esiste un elemento negativo nella permutazione) {
    Trova l'indice π_j massimo negativo
    Trova l'indice $\pi_i = |\pi_j| - 1$
    Inverti la coppia orientata (π_i, π_j)
}

$$0\ (\ 1\ 2\ 3\ 4\ )\ 5$$

Lemma 1

Considerando che

- Ogni permutazione che contiene un elemento negativo contiene un'inversione MAX
- Ogni sequenza di inversioni orientate *safe* è ottimale (la migliore possibile)

Si può definire il seguente Lemma:

Lemma 1

In assenza di inversioni MAX unsafe per ogni step di ordinamento, l'algoritmo MAX produce una sequenza di ordinamento ottimale.

Algoritmo MIN

Le stesse considerazioni possono essere fatte *mutatis mutandis* se si considera una coppia orientata con il minimo elemento negativo. In questo caso l'algoritmo (identico) si chiama algoritmo MIN:

while(la permutazione contiene almeno un elemento negativo)
do
  Trova l'indice dell'elemento negativo minimo π_k
  Trova l'indice di $\pi_l = |\pi_k| + 1$
  Effettua l'inversione corrispondente alla coppia (π_k, π_l)
end

Algoritmo MAX: limiti

- L'algoritmo MAX fallisce quando si trova "bloccato" in una permutazione completamente positiva che non sia la permutazione di identità.

Algoritmo MAX: limiti

- L'algoritmo MAX fallisce quando si trova "bloccato" in una permutazione completamente positiva che non sia la permutazione di identità.
- Questo avviene solo quando un'inversione MAX è unsafe

Algoritmo MAX: limiti

- L'algoritmo MAX fallisce quando si trova "bloccato" in una permutazione completamente positiva che non sia la permutazione di identità.
- Questo avviene solo quando un'inversione MAX è unsafe
- La stessa considerazione vale se si usa lo stesso algoritmo che considera però l'elemento minimo negativo (algoritmo MIN)

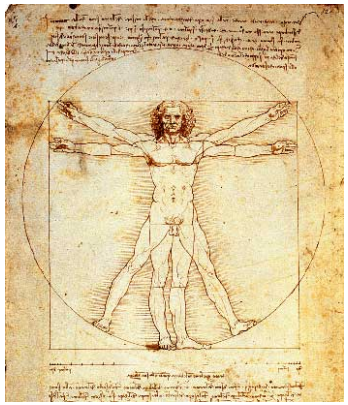
Algoritmo MAX: limiti

- L'algoritmo MAX fallisce quando si trova "bloccato" in una permutazione completamente positiva che non sia la permutazione di identità.
- Questo avviene solo quando un'inversione MAX è unsafe
- La stessa considerazione vale se si usa lo stesso algoritmo che considera però l'elemento minimo negativo (algoritmo MIN)
- Si cerca di superare queste limitazioni combinando i due approcci ed introducendo una scelta casuale: algoritmo RAND.

Algoritmo RAND

while (esiste un elemento negativo nella permutazione) **do**
  Schegli a caso tra MAX e MIN
  if (MAX) **then**
    Trova l'indice dell'elemento negativo massimo π_j
    Trova l'indice di $\pi_i = |\pi_j| - 1$
    Effettua l'inversione rispetto alla coppia (π_i, π_j)
  else if (MIN) **then**
    Trova l'indice dell'elemento negativo minimo π_k
    Trova l'indice di $\pi_i = |\pi_k| + 1$
    Effettua l'inversione rispetto alla coppia (π_i, π_k)
end while

Funzionamento



Problema

Problema

Come trovare la coppia MAX?

Splay Tree: mantenimento delle coppie Max e Min
Soluzione logaritmica

Si tiene traccia del massimo elemento negativo di un'inversione mediante lo splay tree:

- Sia MAX_{neg} il massimo elemento negativo di un sottoalbero

Splay Tree: mantenimento delle coppie Max e Min
Soluzione logaritmica

Si tiene traccia del massimo elemento negativo di un'inversione mediante lo splay tree:

- Sia MAX_{neg} il massimo elemento negativo di un sottoalbero
- Sia MIN_{pos} il minimo elemento positivo dello stesso un sottoalbero

Splay Tree: mantenimento delle coppie Max e Min
Soluzione logaritmica

Si tiene traccia del massimo elemento negativo di un'inversione mediante lo splay tree:

- Sia MAX_{neg} il massimo elemento negativo di un sottoalbero
- Sia MIN_{pos} il minimo elemento positivo dello stesso un sottoalbero
- Si mantengono i valori MAX_{neg} e MIN_{pos} per ogni nodo dell'albero

Splay Tree: mantenimento delle coppie Max e Min
Soluzione logaritmica

Si tiene traccia del massimo elemento negativo di un'inversione mediante lo splay tree:

- Sia MAX_{neg} il massimo elemento negativo di un sottoalbero
- Sia MIN_{pos} il minimo elemento positivo dello stesso un sottoalbero
- Si mantengono i valori MAX_{neg} e MIN_{pos} per ogni nodo dell'albero
- Il valore MAX_{neg} della radice dell'albero e` il massimo elemento negativo della permutazione, *ovvero* l'elemento negativo della coppia MAX

Splay Tree: mantenimento delle coppie Max e Min
Soluzione logaritmica

Si tiene traccia del massimo elemento negativo di un'inversione mediante lo splay tree:

- Sia MAX_{neg} il massimo elemento negativo di un sottoalbero
- Sia MIN_{pos} il minimo elemento positivo dello stesso un sottoalbero
- Si mantengono i valori MAX_{neg} e MIN_{pos} per ogni nodo dell'albero
- Il valore MAX_{neg} della radice dell'albero e` il massimo elemento negativo della permutazione, *ovvero* l'elemento negativo della coppia MAX
- Ovviamente quando viene settato il flag di un nodo si scambiano tra di loro i valori di MAX_{neg} e MIN_{pos}.

Splay Tree: mantenimento delle coppie Max e Min
Soluzione logaritmica

Un'operazione di splay effettua una serie di rotazioni incentrate sull'indice di un nodo.

- Ogni rotazione cambia al massimo la posizione di tre nodi dell'albero, mantenendo possibile la ricerca binaria

Splay Tree: mantenimento delle coppie Max e Min
Soluzione logaritmica

Un'operazione di splay effettua una serie di rotazioni incentrate sull'indice di un nodo.

- Ogni rotazione cambia al massimo la posizione di tre nodi dell'albero, mantenendo possibile la ricerca binaria
- MAX_{neg} può essere ricalcolato solo per il sottoalbero affetto dall' operazione

Splay Tree: mantenimento delle coppie Max e Min
Soluzione logaritmica

Un'operazione di splay effettua una serie di rotazioni incentrate sull'indice di un nodo.

- Ogni rotazione cambia al massimo la posizione di tre nodi dell'albero, mantenendo possibile la ricerca binaria
- MAX_{neg} può essere ricalcolato solo per il sottoalbero affetto dall' operazione
- Il valore di MAX_{neg} può essere mantenuto per ciascuno dei sottoalberi semplicemente controllando i figli della radice ad ogni operazione
- (Ricordate: l'applicazione di un'inversione $\rho(\pi_i, \pi_j)$ comporta la scomposizione dell'albero in tre sottoalberi, il settaggio di un flag (per l'albero da invertire), e la ricomposizione degli stessi)

Siccome il calcolo di MAX_{neg} richiede tempo $O(1)$ per nodo, l'inserimento dei valori MAX_{neg} e MIN_{pos} non altera la complessità della struttura dati, e vale dunque:

Lemma 2

Per ogni permutazione (con segno) di dimensione n, esiste una struttura dati che effettua un'inversione in tempo $O(\log n)$, e mantiene informazioni riguardo al massimo elemento negativo della permutazione.

Trovare la coppia MAX

Si possono quindi trovare gli elementi della coppia MAX

- L'elemento negativo della coppia MAX (π_j) si torva nella struttura dati:

Trovare la coppia MAX

Si possono quindi trovare gli elementi della coppia MAX

- L'elemento negativo della coppia MAX (π_j) si torva nella struttura dati:
  - Se l'elemento di un nodo non è uguale al suo valore MAX_{neg}, allora il massimo elemento negativo si trova nei figli

Trovare la coppia MAX

Si possono quindi trovare gli elementi della coppia MAX

- L'elemento negativo della coppia MAX (π_j) si torva nella struttura dati:
  - Se l'elemento di un nodo non è uguale al suo valore MAX_{neg}, allora il massimo elemento negativo si trova nei figli
  - Quindi basta scendere dalla radice

Trovare la coppia MAX

Si possono quindi trovare gli elementi della coppia MAX

- L'elemento negativo della coppia MAX (π_j) si torva nella struttura dati:
  - Se l'elemento di un nodo non è uguale al suo valore MAX_{neg}, allora il massimo elemento negativo si trova nei figli
  - Quindi basta scendere dalla radice
  - L'unico accorgimento sta nel fatto che se si incontra un flag invertito si deve spingerlo verso il basso per garantire che i vari valori di MAX_{neg} siano aggiornati.

Trovare la coppia MAX

Si possono quindi trovare gli elementi della coppia MAX

- L'elemento negativo della coppia MAX (π_j) si torva nella struttura dati:
  - Se l'elemento di un nodo non è uguale al suo valore MAX_{neg}, allora il massimo elemento negativo si trova nei figli
  - Quindi basta scendere dalla radice
  - L'unico accorgimento sta nel fatto che se si incontra un flag invertito si deve spingerlo verso il basso per garantire che i vari valori di MAX_{neg} siano aggiornati.
- Il secondo elemento della coppia (π_i) viene trovato tramite un vettore ausiliario:

Trovare la coppia MAX

Si possono quindi trovare gli elementi della coppia MAX

- L'elemento negativo della coppia MAX (π_j) si torva nella struttura dati:
  - Se l'elemento di un nodo non è uguale al suo valore MAX_{neg}, allora il massimo elemento negativo si trova nei figli
  - Quindi basta scendere dalla radice
  - L'unico accorgimento sta nel fatto che se si incontra un flag invertito si deve spingerlo verso il basso per garantire che i vari valori di MAX_{neg} siano aggiornati.
- Il secondo elemento della coppia (π_i) viene trovato tramite un vettore ausiliario:
  - Si usa un vettore di puntatori di n elementi

Trovare la coppia MAX

Si possono quindi trovare gli elementi della coppia MAX

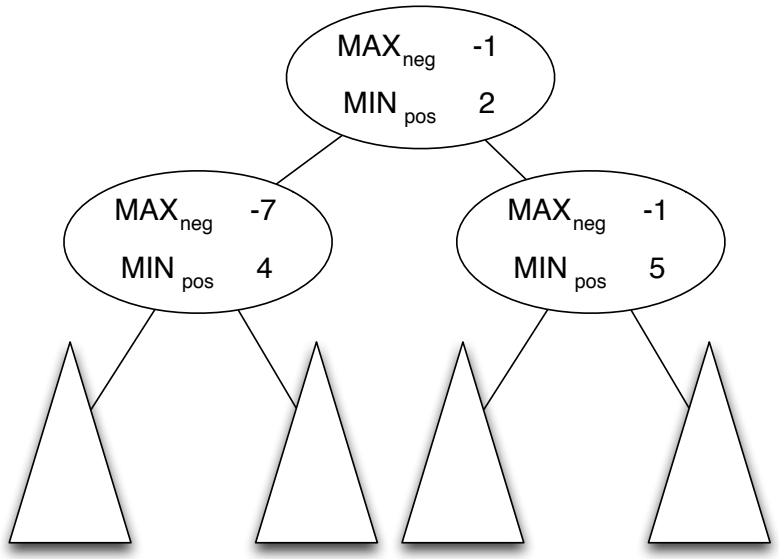
- L'elemento negativo della coppia MAX (π_j) si torva nella struttura dati:
  - Se l'elemento di un nodo non è uguale al suo valore MAX_{neg}, allora il massimo elemento negativo si trova nei figli
  - Quindi basta scendere dalla radice
  - L'unico accorgimento sta nel fatto che se si incontra un flag invertito si deve spingerlo verso il basso per garantire che i vari valori di MAX_{neg} siano aggiornati.
- Il secondo elemento della coppia (π_i) viene trovato tramite un vettore ausiliario:
  - Si usa un vettore di puntatori di n elementi
  - Ogni puntatore mappa il nodo che lo contiene

Trovare la coppia MAX

Si possono quindi trovare gli elementi della coppia MAX

- L'elemento negativo della coppia MAX (π_j) si torva nella struttura dati:
  - Se l'elemento di un nodo non è uguale al suo valore MAX_{neg}, allora il massimo elemento negativo si trova nei figli
  - Quindi basta scendere dalla radice
  - L'unico accorgimento sta nel fatto che se si incontra un flag invertito si deve spingerlo verso il basso per garantire che i vari valori di MAX_{neg} siano aggiornati.
- Il secondo elemento della coppia (π_i) viene trovato tramite un vettore ausiliario:
  - Si usa un vettore di puntatori di n elementi
  - Ogni puntatore mappa il nodo che lo contiene
  - Questo vettore non cambia durante la computazione, e quindi garantisce di trovare qualsiasi elemento in tempo costante

Splay Tree: trovare la coppia MAX
Esempio



Splay Tree con valori MAX e MIN per ogni nodo

Trovare l'inversione MAX

Come individuare gli indici degli elementi dell'inversione MAX

- Se nello splay tree non esistono flag reversed attivi allora gli indici i e j dell'inversione MAX $\rho(i, j)$ possono essere ottenuti direttamente dalle posizioni dei nodi corrispondenti alla coppia MAX.

Trovare l'inversione MAX

Come individuare gli indici degli elementi dell'inversione MAX

- Se nello splay tree non esistono flag reversed attivi allora gli indici i e j dell'inversione MAX $\rho(i, j)$ possono essere ottenuti direttamente dalle posizioni dei nodi corrispondenti alla coppia MAX.
- La presenza di un flag indica però che ci siano indici da aggiornare, obbligando a lavoro ulteriore per ottenere i valori i e j.

Trovare l'inversione MAX

Come individuare gli indici degli elementi dell'inversione MAX

L'indice di un nodo rispetto allo stato attuale della permutazione può essere calcolato usando l'indice del nodo padre, e le dimensioni dei sottalberi di destra e sinistra.

- Perciò l'indice corrente di un nodo può essere calcolato ogni volta che il flag reversed viene spinto verso il basso.

Trovare l'inversione MAX

Come individuare gli indici degli elementi dell'inversione MAX

L'indice di un nodo rispetto allo stato attuale della permutazione può essere calcolato usando l'indice del nodo padre, e le dimensioni dei sottalberi di destra e sinistra.

- Perciò l'indice corrente di un nodo può essere calcolato ogni volta che il flag reversed viene spinto verso il basso.
- La dimensione del sottoalbero con radice in un nodo è facilmente mantenuto:

Trovare l'inversione MAX

Come individuare gli indici degli elementi dell'inversione MAX

L'indice di un nodo rispetto allo stato attuale della permutazione può essere calcolato usando l'indice del nodo padre, e le dimensioni dei sottalberi di destra e sinistra.

- Perciò l'indice corrente di un nodo può essere calcolato ogni volta che il flag reversed viene spinto verso il basso.
- La dimensione del sottoalbero con radice in un nodo è facilmente mantenuto:
  - Se il nodo è un figlio destro, l'indice è calcolato come la somma tra gli indici dei nodi padre più uno, più la dimensione del sottoalbero di sinistra.

Trovare l'inversione MAX

Come individuare gli indici degli elementi dell'inversione MAX

L'indice di un nodo rispetto allo stato attuale della permutazione può essere calcolato usando l'indice del nodo padre, e le dimensioni dei sottalberi di destra e sinistra.

- Perciò l'indice corrente di un nodo può essere calcolato ogni volta che il flag reversed viene spinto verso il basso.
- La dimensione del sottoalbero con radice in un nodo è facilmente mantenuto:
  - Se il nodo è un figlio destro, l'indice è calcolato come la somma tra gli indici dei nodi padre più uno, più la dimensione del sottoalbero di sinistra.
  - Se il nodo è un figlio sinistro, l'indice è calcolato come uno meno la differenza degli indici dei nodi padre, più la dimensione del sottoalbero di destra.

Trovare l'inversione MAX

Come individuare gli indici degli elementi dell'inversione MAX

L'indice di un nodo rispetto allo stato attuale della permutazione può essere calcolato usando l'indice del nodo padre, e le dimensioni dei sottalberi di destra e sinistra.

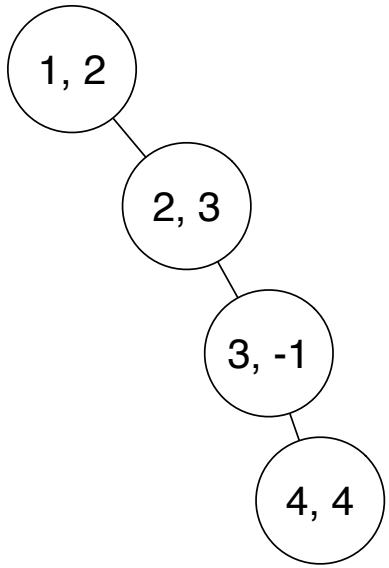
- Perciò l'indice corrente di un nodo può essere calcolato ogni volta che il flag reversed viene spinto verso il basso.
- La dimensione del sottoalbero con radice in un nodo è facilmente mantenuto:
  - Se il nodo è un figlio destro, l'indice è calcolato come la somma tra gli indici dei nodi padre più uno, più la dimensione del sottoalbero di sinistra.
  - Se il nodo è un figlio sinistro, l'indice è calcolato come uno meno la differenza degli indici dei nodi padre, più la dimensione del sottoalbero di destra.
  - L'indice della radice è semplicemente la dimensione del suo sottoalbero di sinistra

Trovare l'inversione MAX

Come individuare gli indici degli elementi dell'inversione MAX

Dunque attraversando l'albero dalla radice per cercare il massimo elemento negativo π_j, gli indici vengono ricalcolati. L'altro elemento della coppia MAX (π_i) è poi trovato usando il vettore di lookup il suo indice aggiornato può essere ottenuto attraversando verso la radice (usando i puntatori dei genitori) e scendendo verso lo stesso path, pushando i flag reversed e ricalcolando gli indici per ogni nodo.

Splay Tree: trovare gli indici della coppia MAX
Esempio 1



$\pi = 2\ 3\ -1\ -4$ inversione tra $\rho(2,3)$

Splay Tree: trovare gli indici della coppia MAX
Esempio 2

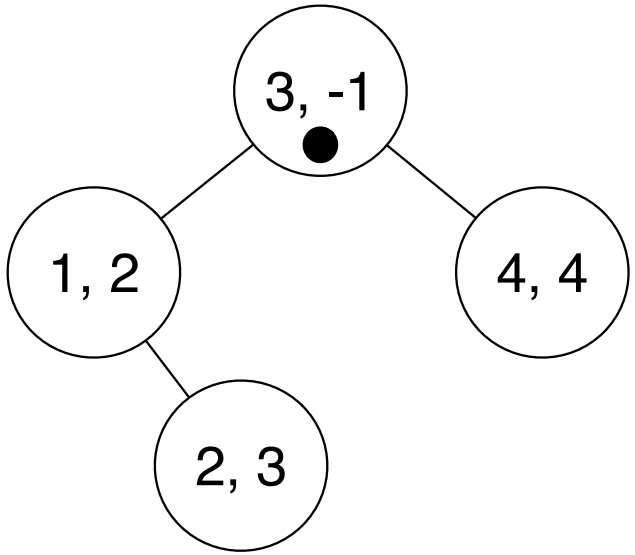
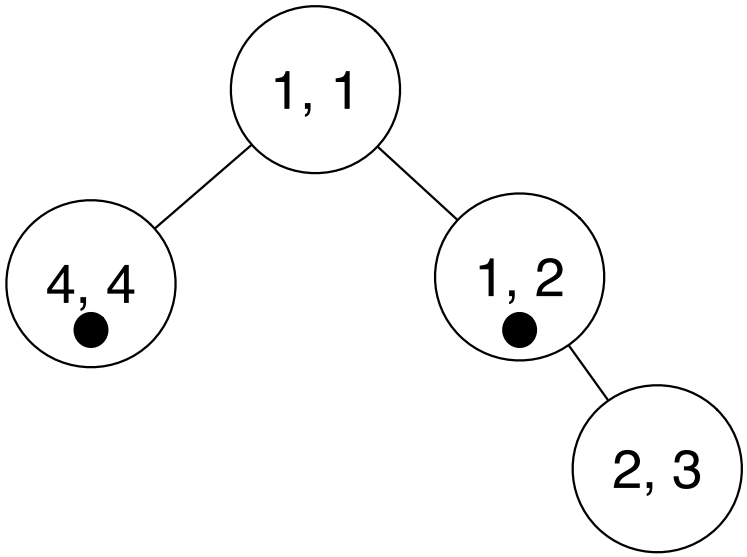


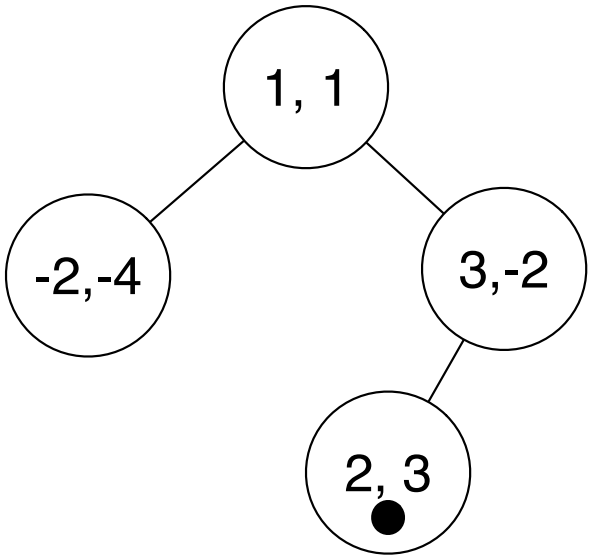
Figura: $\pi = 1 \ -3 \ -2 \ -4$

Splay Tree: trovare gli indici della coppia MAX
Esempio 3



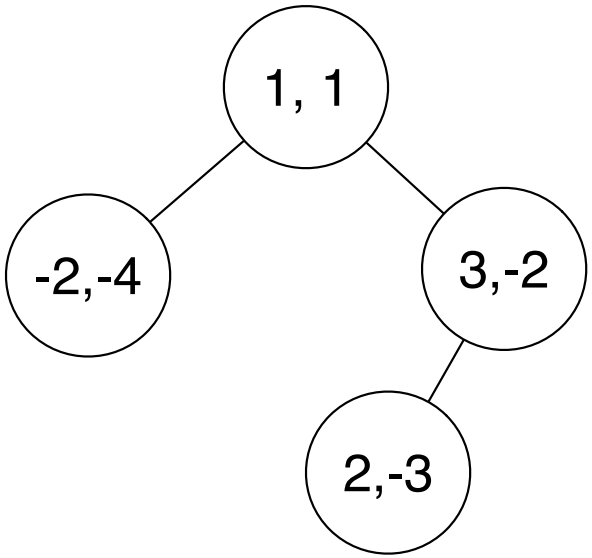
$\pi = 1 \ -3 \ -2 \ -4$ con push del flag verso i figli

Splay Tree: trovare gli indici della coppia MAX
Esempio 4



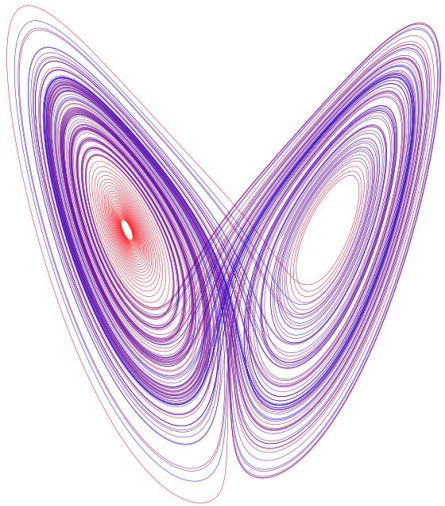
$$\pi = 1 \ -3 \ -2 \ -4$$

Splay Tree: trovare gli indici della coppia MAX
Esempio 5



$\pi = 1 \ -3 \ -2 \ -4$ Gli indici sono ricalcolati correttamente

Complessità



La complessità computazionale richiesta per trovare la coppia MAX:

- Push dei flag reversed: $O(1)$ per nodo

Visione d'insieme
Complessità computazionale: trovare la coppia MAX

La complessità computazionale richiesta per trovare la coppia MAX:

- Push dei flag reversed: $O(1)$ per nodo
- Trovare il massimo elemento negativo π_j e il suo indice aggiornato j: $O(\log n)$

Visione d'insieme
Complessità computazionale: trovare la coppia MAX

La complessità computazionale richiesta per trovare la coppia MAX:

- Push dei flag reversed: $O(1)$ per nodo
- Trovare il massimo elemento negativo π_j e il suo indice aggiornato j: $O(\log n)$
- Trovare l'elemento π_i della coppia MAX: $O(1)$

La complessità computazionale richiesta per trovare la coppia MAX:

- Push dei flag reversed: $O(1)$ per nodo
- Trovare il massimo elemento negativo π_j e il suo indice aggiornato j: $O(\log n)$
- Trovare l'elemento π_i della coppia MAX: $O(1)$
- Trovare l'indice i di π_i aggiornato: $O(\log n)$

La complessità computazionale richiesta per trovare la coppia MAX:

- Push dei flag reversed: $O(1)$ per nodo
- Trovare il massimo elemento negativo π_j e il suo indice aggiornato j: $O(\log n)$
- Trovare l'elemento π_i della coppia MAX: $O(1)$
- Trovare l'indice i di π_i aggiornato: $O(\log n)$
- Dunque complessivamente trovare la coppia MAX richiede $O(\log n)$

Visione d'insieme
Complessità computazionale: effettuare l'inversione MAX

La complessità computazionale richiesta per effettuare l'inversione MAX:

- Mantenere MAX_{neg}, MIN_{pos}, MIN_{neg}, MAX_{pos}: $O(1)$ per le operazioni di split e join sullo splay tree

Visione d'insieme
Complessità computazionale: effettuare l'inversione MAX

La complessità computazionale richiesta per effettuare l'inversione MAX:

- Mantenere MAX_{neg}, MIN_{pos}, MIN_{neg}, MAX_{pos}: $O(1)$ per le operazioni di split e join sullo splay tree
- Ogni rotazione per i due splay: $O(1)$ per ogni rotazione

In altre parole, si è data prova del seguente

Teorema 1

Per ogni permutazione con segno di dimensione n esiste una struttura dati tale che:

- Consenta di verificare l'esistenza di un'inversione orientata in $O(1)$
- Consente di effettuare un'inversione MAX (o MIN), senza perdere traccia della permutazione, in tempo $O(\log n)$
- E ha dimensione $O(n)$

E del seguente

Teorema 2

In assenza di inversioni unsafe per ogni passo di ordinamento, l'algoritmo MAX produce una sequenza di ordinamento ottima in tempo $O(n\log n)$

Componenti Bad

Fallimenti degli algoritmi

Sia l'algoritmo MAX che l'algoritmo RAND possono bloccarsi su una permutazione completamente positiva (diversa dall'identità) prodotta da un'inversione unsafe.
Si studiano tre soluzioni

- Ripartenze casuali

Fallimenti degli algoritmi

Sia l'algoritmo MAX che l'algoritmo RAND possono bloccarsi su una permutazione completamente positiva (diversa dall'identità) prodotta da un'inversione unsafe.
Si studiano tre soluzioni

- Ripartenze casuali
- Recovery di Tannier e Sagot

Fallimenti degli algoritmi

Sia l'algoritmo MAX che l'algoritmo RAND possono bloccarsi su una permutazione completamente positiva (diversa dall'identità) prodotta da un'inversione unsafe.
Si studiano tre soluzioni

- Ripartenze casuali
- Recovery di Tannier e Sagot
- Recovery da un'inversione unsafe

RAND

L'algoritmo RAND propone già una strategia per evitare dei fallimenti.
Evidenze sperimentali dimostrano che l'algoritmo RAND funziona per la maggior parte dei casi, però esistono delle permutazioni per cui l'algoritmo si blocca:

Esempio

Per la permutazione $(3\ 1\ -4\ -2)$ sia con un'inversione MAX che con un'inversione MIN si ottiene la permutazione $(3\ 1\ 2\ 4)$, che è positiva ma non è l'identità

Recovery da un'inversione unsafe
Approccio di Tannier e Sagot

Approccio *postmortem*:

- Si tiene traccia di tutte le inversioni effettuate

Recovery da un'inversione unsafe
Approccio di Tannier e Sagot

Approccio *postmortem*:

- Si tiene traccia di tutte le inversioni effettuate
- Quando si arriva ad una permutazione positiva ma diversa dall' identiá rollback fino alla più recende inversione unsafe

Recovery da un'inversione unsafe
Approccio di Tannier e Sagot

Approccio *postmortem*:

- Si tiene traccia di tutte le inversioni effettuate
- Quando si arriva ad una permutazione positiva ma diversa dall' identiá rollback fino alla più recende inversione unsafe
- Il tutto viene fatto mediante un grafo overlay che tiene traccia dei breakpoint rimanenti e del loro orientamento.

Recovery da un'inversione unsafe
Approccio innovativo

Similmente all'approccio di Tannier e Sagot si usa un grafo ausiliario per tenere traccia delle inversioni applicate; il grafo però è minimale.
Dette p_1 e p_i rispettivamente la prima permutazione che genera il fail e l'i-esima permutazione, il recovery comporta tre step:

- Trovare la più recente inversione unsafe, μ_i

Recovery da un'inversione unsafe
Approccio innovativo

Similmente all'approccio di Tannier e Sagot si usa un grafo ausiliario per tenere traccia delle inversioni applicate; il grafo però è minimale.
Dette p_1 e p_i rispettivamente la prima permutazione che genera il fail e l'i-esima permutazione, il recovery comporta tre step:

- Trovare la più recente inversione unsafe, μ_i
- Sostituire μ_i con un'inversione safe

Recovery da un'inversione unsafe
Approccio innovativo

Similmente all'approccio di Tannier e Sagot si usa un grafo ausiliario per tenere traccia delle inversioni applicate; il grafo però è minimale.
Dette p_1 e p_i rispettivamente la prima permutazione che genera il fail e l'i-esima permutazione, il recovery comporta tre step:

- Trovare la più recente inversione unsafe, μ_i
- Sostituire μ_i con un'inversione safe
- Aggiungere le inversioni senza annullare quelle che sono già state applicate dopo μ_i.

Recovery da un'inversione unsafe

Trovare l'inversione unsafe più recente

Definizioni:

- $\pi \cdot \rho$: nuova permutazione risultato dell'applicazione dell'inversione ρ alla permutazione π

Recovery da un'inversione unsafe

Trovare l'inversione unsafe più recente

Definizioni:

- $\pi \cdot \rho$: nuova permutazione risultato dell'applicazione dell'inversione ρ alla permutazione π
- $\pi \cdot S$: risultato dell'applicazione di una sequenza di inversioni S alla permutazione π

Recovery da un'inversione unsafe

Trovare l'inversione unsafe più recente

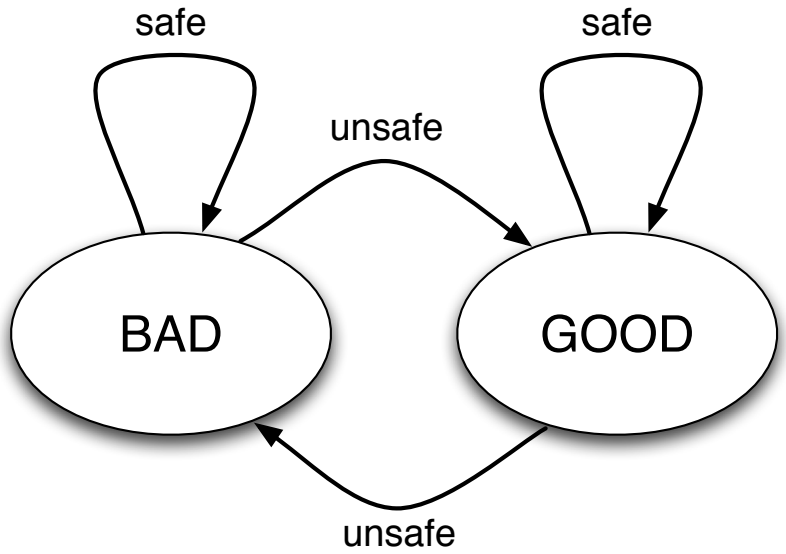
Definizioni:

- $\pi \cdot \rho$: nuova permutazione risultato dell'applicazione dell'inversione ρ alla permutazione π
- $\pi \cdot S$: risultato dell'applicazione di una sequenza di inversioni S alla permutazione π
- Annullare la sequenza di inversioni $S = \rho_1, \rho_2, ..., \rho_n$ dalla permutazione $\pi \cdot S$ significa effettuare le inversioni in senso opposto, con risultato: $\pi \cdot S \cdot \rho_n \cdot ... \cdot \rho_1 = \pi$

Trovare la più recente unsafe inversion 2
Nota 1

Nota 1

Quando si annulla una serie di inversioni da S_i, la più recente inversione unsafe μ_i è la prima inversione che modifica un elemento da bad component a good.



Trovare la più recente unsafe inversion 3
Algoritmo

Dunque l'algoritmo per tornare all'inversione unsafe più recente è:

- Annulla la sequenza di inversioni S_i una alla volta
- Per ogni inversione controlla se esistono componenti bad
- Fermati quando trovi un'inversione unsafe
- Sostituisci con una safe

$$\underbrace{\Pi_{in} \cdot \rho_1 \cdot \rho_2 \cdot \ldots \cdot \mu_{j-1}}_{S_1} \cdot \underbrace{\mu_j}_{v_j} \cdot \underbrace{\ldots \cdot \rho_n}_{S_2}$$

Trovare la più recente unsafe inversion
Costo complessivo

Complessivamente, si hanno questi costi:

Costo delle operazioni sull'albero ausiliario

Detta S_1 la sequenza di inversioni applicate prima di μ_j, e detta S_2 la sequenza di inversioni applicate dopo μ_j. Ogni inversione safe in in S_2 che è annullata prima di incontrare μ_j ha un costo in tempo di $O(\log\, n)$, quindi il costo totale per trovare l'inversione unsafe più recente è di $O(n + |S_2|\log n)$

Analisi dell'esecuzione complessiva
Costo esecuzione di MAX-RECOVER e RAND-RECOVER

Gli algoritmi comprensivi della fase di recovery prendono il nome di MAX-RECOVER e RAND-RECOVER.

- L'esecuzione normale richiede $O(n \log n)$

Analisi dell'esecuzione complessiva
Costo esecuzione di MAX-RECOVER e RAND-RECOVER

Gli algoritmi comprensivi della fase di recovery prendono il nome di MAX-RECOVER e RAND-RECOVER.

- L'esecuzione normale richiede $O(n \log n)$
- In caso di fault, la ricerca degli elementi q_i e μ_i richiede $O(n + |S_2|\log n)$

Analisi dell'esecuzione complessiva
Congettura sul tempo di esecuzione

Dati statistici hanno dimostrato che

Teorema

Il tempo di esecuzione di MAX-RECOVER o RAND-RECOVER è di $O(n \log n + kn)$ dove k è il numero totale di inversioni unsafe effettuate dall'algoritmo.

E' stato empiricamente dimostrato che anche su grandi permutazioni casuali con n variabile il valore medio e la devianza di k rimangono costanti ($k \approx 0.5$) e si congettura che questo sia il reale valore di esecuzione dell'algoritmo.

Analisi dell'esecuzione complessiva
TUTTAVIA...

Tuttavia esiste almeno una classe certa di permutazioni per la quale il tempo di esecuzione degli algoritmi diventa $O(n^2)$, ovvero:

Permutazione tuttavia

Costruire una permutazione di lunghezza n partendo dalla permutazione identità lunga n mod 5 come primo blocco, seguita da $n/5$ copie del blocco $i, (i+3), (i+1), -(i+4), -(i+2), (i+5)$ ciascuno dei quali condivide il suo primo elemento con l'ultimo del blocco precedente

Conclusioni

Evidenze sperimentali
Risultati dei test di esecuzione - 1

- Sono stati eseguiti test sugli algoritmi MAX, RAND, MAX-RECOVER e RAND-RECOVER

Evidenze sperimentali
Risultati dei test di esecuzione - 1

- Sono stati eseguiti test sugli algoritmi MAX, RAND, MAX-RECOVER e RAND-RECOVER
- Ciascun algoritmo è stato testato con permutazioni casuali di lunghezza 100, 200, 500, 1000, 5000, 10k, 20k.

Evidenze sperimentali
Risultati dei test di esecuzione - 1

- Sono stati eseguiti test sugli algoritmi MAX, RAND, MAX-RECOVER e RAND-RECOVER
- Ciascun algoritmo è stato testato con permutazioni casuali di lunghezza 100, 200, 500, 1000, 5000, 10k, 20k.
- Per ciascuna lunghezza ciascun algoritmo è stato testato su 1M permutazioni casuali differenti

Evidenze sperimentali
Risultati dei test di esecuzione - 1

- Sono stati eseguiti test sugli algoritmi MAX, RAND, MAX-RECOVER e RAND-RECOVER
- Ciascun algoritmo è stato testato con permutazioni casuali di lunghezza 100, 200, 500, 1000, 5000, 10k, 20k.
- Per ciascuna lunghezza ciascun algoritmo è stato testato su 1M permutazioni casuali differenti
- Non vengono fornite informazioni sul tipo di casualità impiegata.

Evidenze sperimentali - 2
Tassi di fallimento

Questa tabella mostra i tassi percentuali di fallimento per gli algoritmi MAX e RAND. Mediamente gli algoritmi producono una sequenza di ordinamenti per il 61% dei tentativi. RAND-RESTART riesce a portarsi intorno all' 82% ma in questa tabella non viene indicato il prezzo pagato in termini di k ripartenze.

Length	100	200	500	1,000	2,000	5,000	10,000	20,000
MAX	39.5%	38.9%	39.0%	39.1%	39.3%	39.3%	39.3%	39.2%
RAND	39.0%	39.2%	39.5%	39.5%	39.6%	39.5%	39.6%	39.5%
RAND-RESTART	17.2 %	17.1 %	16.8 %	16.4 %	16.3 %	16.2 %	16.0 %	16.0 %

Percentuale fallimenti

Evidenze sperimentali - 3
Numero medio di recover per MAX-RECOVER

Questa tabella mostra i valori medi del parametro k per ciascuna esecuzione, ed un indice di dispersione (deviazione standard) riferiti al 1M di permutazioni testate. Si vede che mediamente gli algoritmi sono soggetti ad un numero di ripartenze pari alla metà della lunghezza della permutazione.

Length	100	200	500	1,000	2,000	5,000	10,000	20,000
AVE(k)	0.513	0.518	0.522	0.524	0.524	0.525	0.524	0.525
SD(k)	0.765	0.770	0.772	0.774	0.773	0.775	0.774	0.777

Numero medio di recover per MAX-RECOVER

Evidenze sperimentali - 4
Numero di fallimenti per RAND-RECOVER

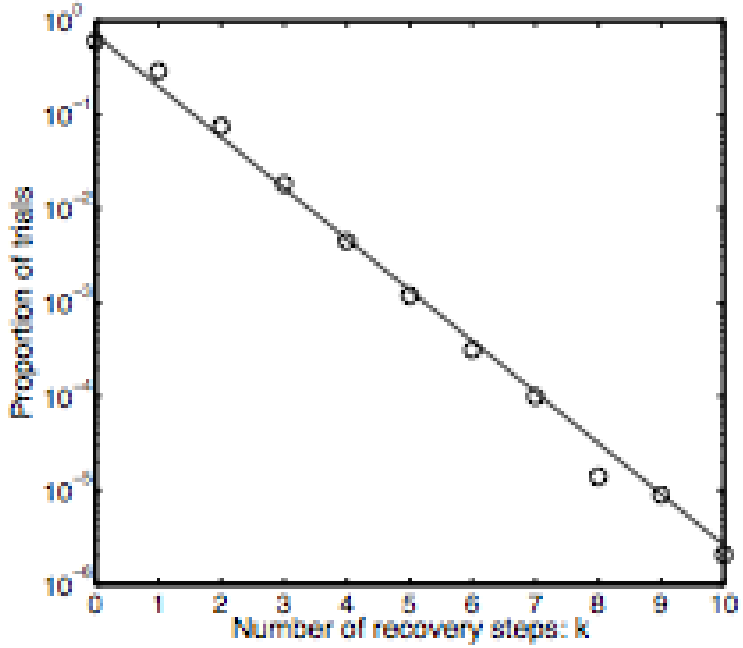
Passando a RAND-RECOVER media e varianza scendono di poco. E' interessante però il fatto che al crescere della dimensione delle permutazioni non cambi la media dei fallimenti (ancora: nulla è detto sul tipo di casualità impiegata).

Length	100	200	500	1,000	2,000	5,000	10,000	20,000
AVE(k)	0.485	0.489	0.492	0.493	0.495	0.495	0.495	0.499
SD(k)	0.690	0.694	0.697	0.697	0.698	0.698	0.698	0.699

Numero medio di recover per RAND-RECOVER

Evidenze sperimentali - 5
Distribuzione di k

Questo grafico infine mostra la distribuzione del parametro k per permutazioni casuali di 10000 elementi. Si accenna ad una somiglianza con la funzione esponenziale inversa.



Proportion of trials - Number of recovery steps