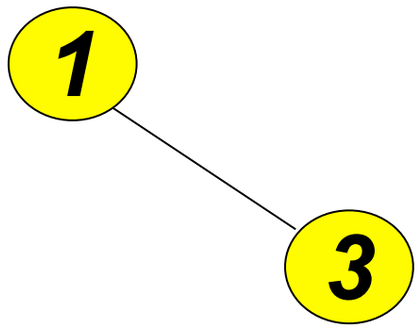
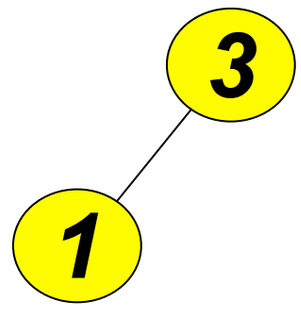
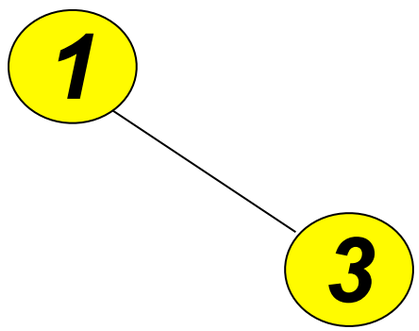
Dalle visite agli alberi es 1

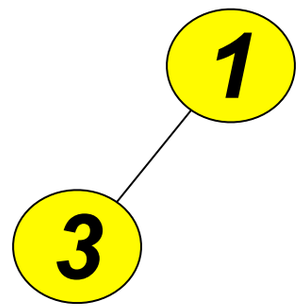
visita inordine: 1 3





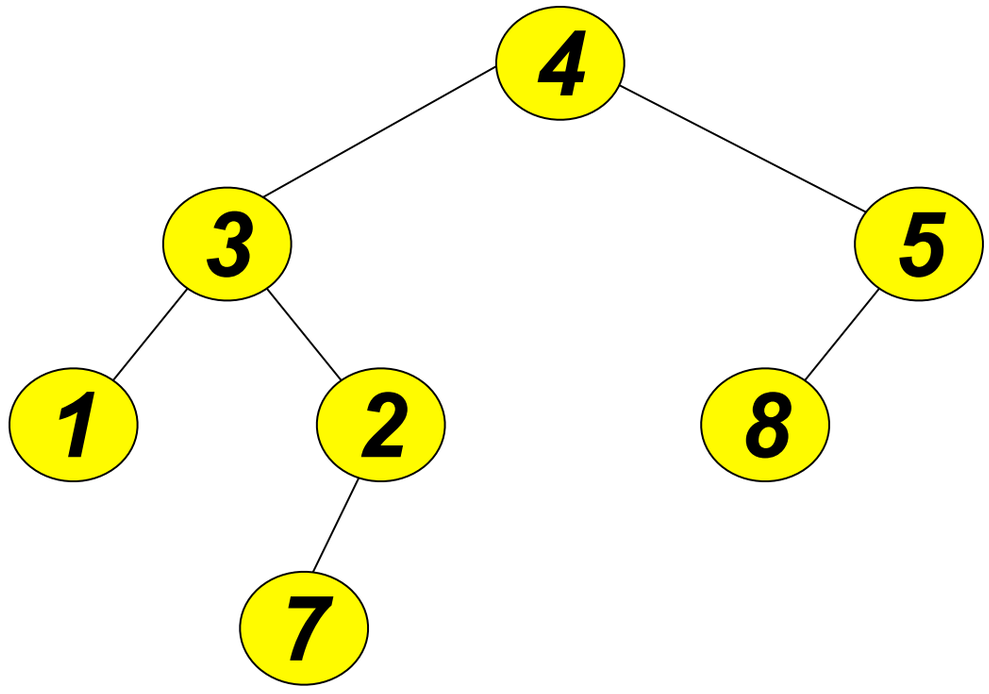
visita preordine: 1 3

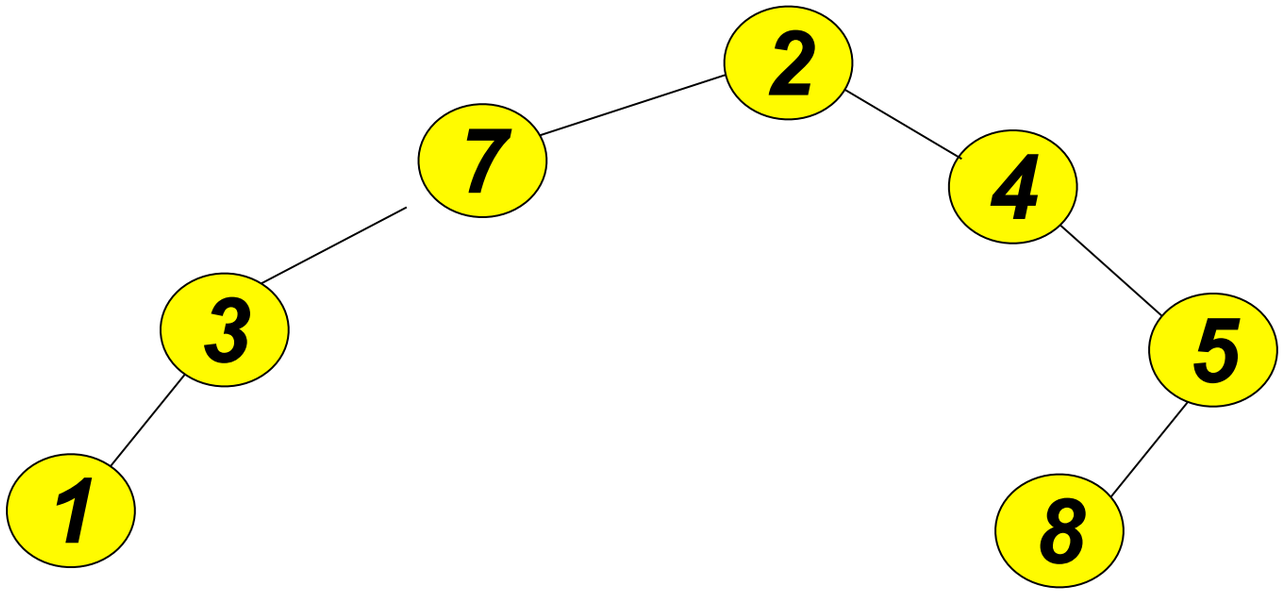




Dalle visite agli alberi es 2

visita inordine: 1 3 7 2 4 8 5



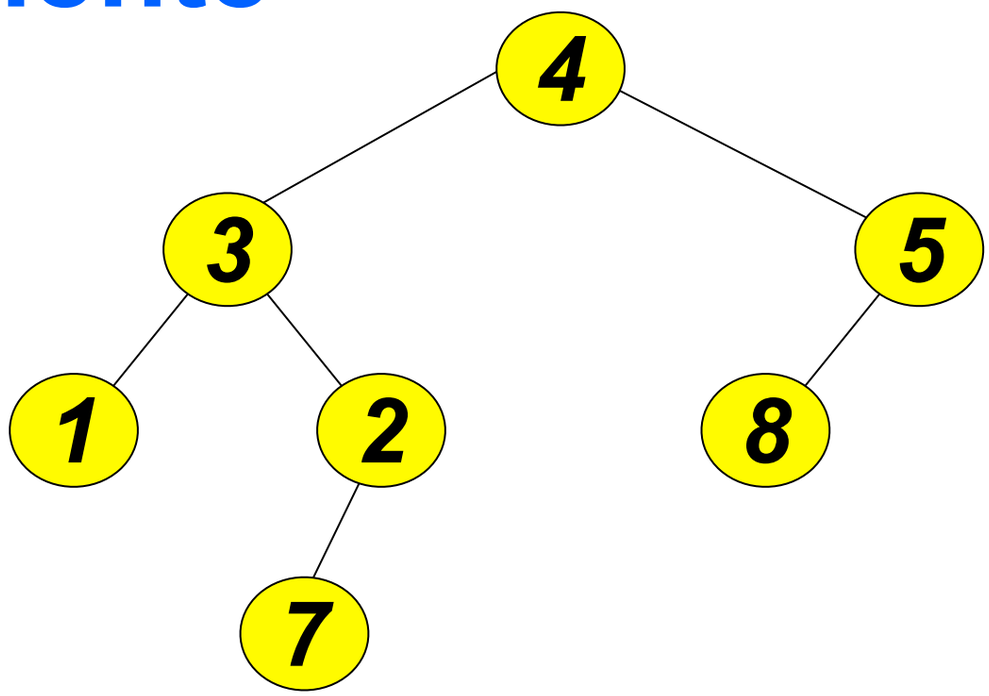


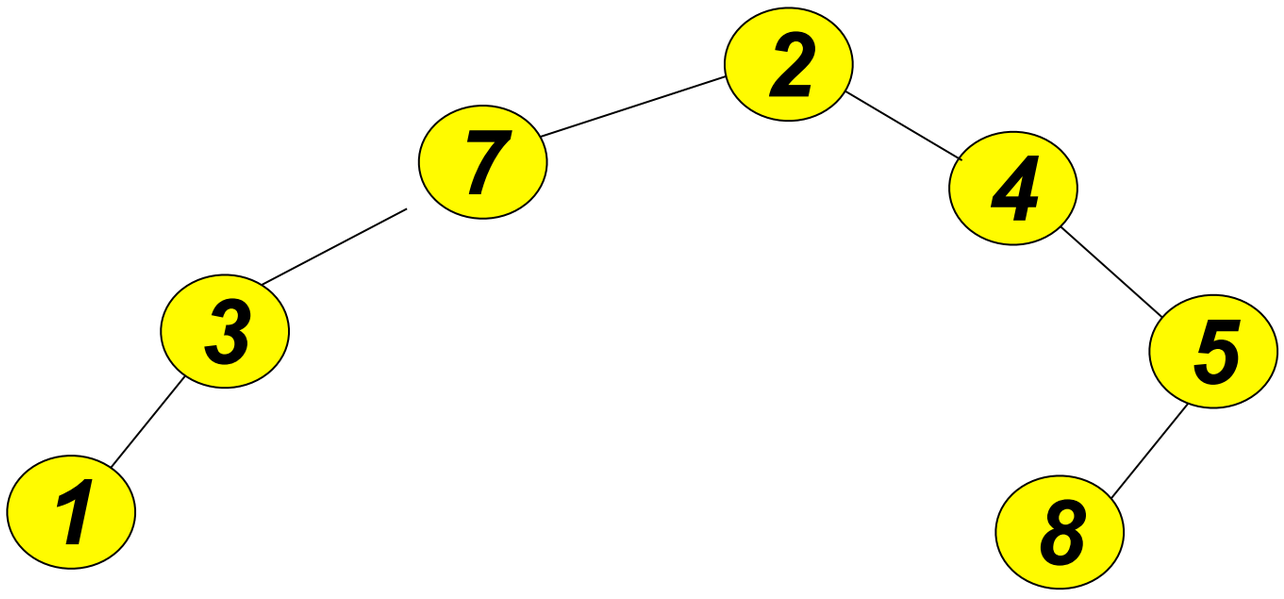
visita inordine: 1 3 7 2 4 8 5

Con due visite individuo l'albero univocamente

visita preordine: 4 3 1 2 7 5 8

visita inordine: 1 3 7 2 4 8 5





visita preordine: 2 7 3 1 4 5 8

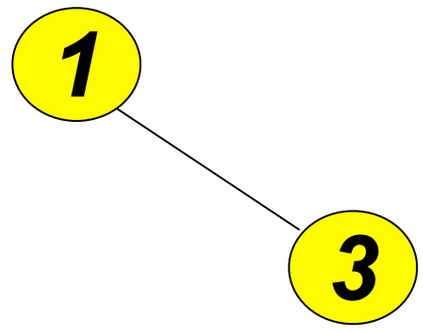
visita inordine: 1 3 7 2 4 8 5

Da 2 visite a un albero es 1

visita inordine: 1 3

visita preordine: 1 3

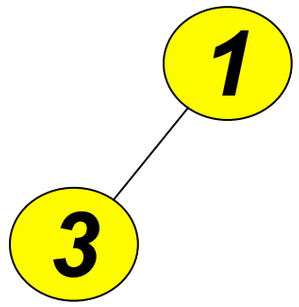
Dalla visita in preordine so che 1 è la radice, dalla inorder so che 3 è figlio destro.



Da 2 visite a un albero es 2

visita preordine: 1 3 **visita inordine: 3 1**

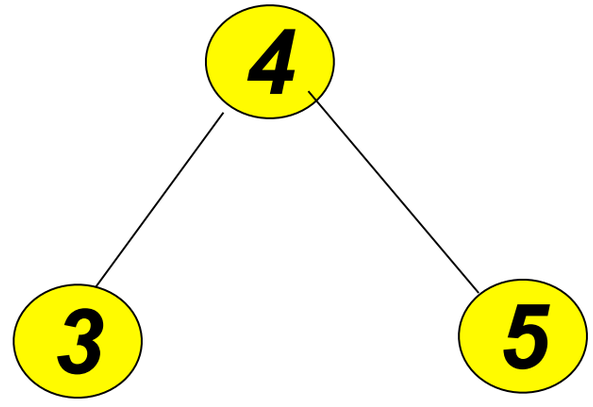
Dalla visita in preordine so che 1 è la radice, dalla inorder so che 3 è figlio sinistro.



Un albero da 2 visite: l'algoritmo

visita inordine: 1 3 7 2 4 8 5

visita preordine: 4 3 1 2 7 5 8



Nella visita in preordine il primo è la radice

Nella visita inordine il sottoalbero sinistro è visitato prima della radice, quindi
1 3 7 2 sono nodi del sottoalbero sinistro e 8 5 del destro.

Applicando ricorsivamente il ragionamento ai due sottoalberi, si ottiene che 3 è la radice del sottoalbero sinistro e 5 del destro.

Esecuzione continuata

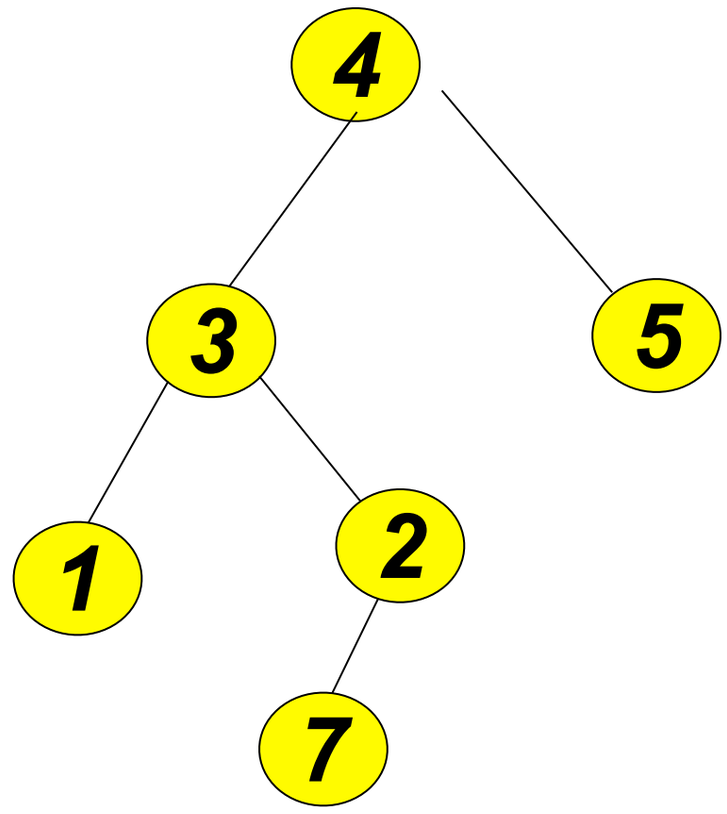
visita inordine: 1 3 7 2 4 8 5

visita preordine: 4 3 1 2 7 5 8

1 3 7 2 sono i nodi del sottoalbero sinistro e 8 5 quelli del destro.

Dalla visita inordine del sottoalbero sinistro ricaviamo che 1 è l'unico nodo nel sottoalbero sinistro di 3 e 7 2 in quello destro.

Poichè nella visita preordine 2 è visitato prima di 7, 2 è la radice e 7 sta nel sottoalbero sinistro perchè precede 2 nella visita inordine.

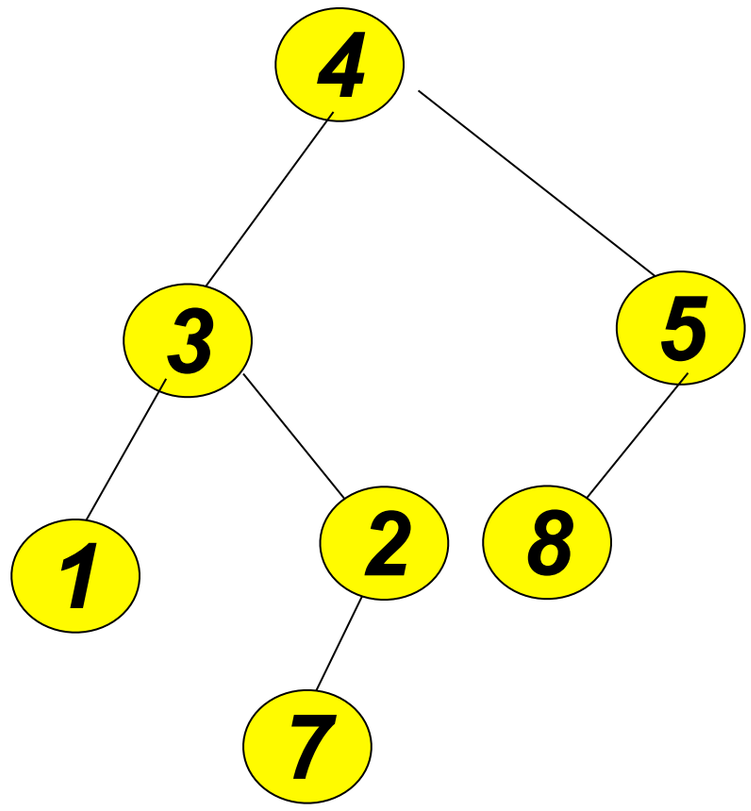


fine esecuzione

visita inordine: 1 3 7 2 4 8 5

visita preordine: 4 3 1 2 7 5 8

1 3 7 2 sono i nodi del sottoalbero sinistro e 8 5 quelli del destro.



Poichè nella visita inordine 8 è visitato prima di 5, 8 sta nel sottoalbero sinistro

L’algoritmo

Date le due visite inordine a: a_1 a_2…a_n e preordine b: b_1b_2…b_n

1. ricava la radice dell’albero come primo elemento della visita in preordine b_1
2. crea un nodo con etichetta b_1.
3. cerca b_1 in a, se $b_1 = a_i$ allora a_1,…,a_{i-1} sono nel sotto albero sinistro e a_{i+1},…,a_n sono nel destro.
Si applicano ricorsivamente i passi 1 - 3 alle sequenze a_1,…a_{i-1} e b_2,…b_i, (devo avere i-1 nodi) che sono le visite inorder e preorder del sotto albero sinistro e a quelle del sotto albero destro a_{i+1},…a_n e b_{i+1},…,b_n rendendo i nodi creati nelle due chiamate rispettivamente figlio sinistro e destro del nodo creato al passo 2.

Alberi da visite

CostrAB(inordine, preordine)

Input: due arrays inordine e preordine con le visite inorder e preorder rispettivamente,

prec:inordine e preordine sono non vuoti, di elementi distinti

output: il puntatore alla radice dell'albero binario T, la cui visita inorder è inordine e quella in preorder è preordine

inorder_lo = 0; inorder_hi = inordine.len - 1; preorder_lo= 0; preorder_hi=inorder_hi

return CoAB_aus(inordine, preordine,inorder_lo,inorder_hi,preorder_lo,preorder_hi)

Si utilizza una funzione di ricerca in un vettore

cerca(inordine,inorder_lo,inorder_hi,x)

Input: un vettore inordine, due indici che individuano l'intervallo di ricerca in inordine e la chiave x da cercare.

prec: inordine è non vuoto, di elementi distinti e lo≤hi

postc: restituisce la posizione in in di x

Lo pseudocodice

```
CoAB_aus(inordine, preordine,inorder_lo,inorder_hi,preorder_lo,preorder_hi)
 Input: due sequenze inordine e preordine con le visite inorder e preorder rispettivamente,
 e gli indici che individuano la porzione di inordine  su cui si costruisce l'albero,e gli indici che
 individuano la porzione di preordine  su cui si costruisce l'albero
 prec: inordine e preordine contengono elementi distinti
 output: dà in output il puntatore alla radice dell' albero binario T, la cui visita inorder è inordine
  e quella in preorder è preordine
  if inorder_lo > inorder_hi  then return NIL
 crea un nodo T, con T.key = preordine[preorder_lo] e campi puntatori a NIL
 if lo_preorder = hi_preorder  then return T
  j = cerca(inordine,inorder_lo,inorder_hi,T.key)
 j è  l'indice della radice in inordine, il sottoalbero sinistro contiene i nodi
 inordine[inorder_lo…j-1] e il destro i nodi inordine[j+1 ... inorder_hi].
 La porzione di preordine che contiene i nodi del sotto albero sinistro è compresa tra l'indice
  preorder_lo+1 e quello ottenuto sommando a preorder_lo il numero di elementi nel sotto albero
  sinistro, j-inorder_lo.  Invece i nodi nel sotto albero destro in preordine sono quelli a seguire,
 quindi dall'indice preorder_lo+1 + (j-inorder_lo) e preorder_hi
 T.left = CoAB_aus(inordine, preordine,inorder_lo,j-1,preorder_lo+1,preorder_lo + (j - inorder_lo))
 T.right = CoAB_aus(inordine, preordine,j+1,inorder_hi,preorder_lo + (j - inorder_lo)+1,preorder_hi)
     return T
```

Alberi da visite

Relazioni di ricorrenza per CoAB_aus.

$T(n) = T(0) + T(n-1) + cn$ che si semplifica con
$T(n) = T(n-1) + cn$
Sappiamo che $T(n) = O(n^2)$.
Verifichiamo che $T(n) \leq kn^2$, per una costante $k>0$, a partire da un certo n in poi. Usiamo l'induzione.
$T(n) = T(n-1) + cn \leq$ (per l'ipotesi induttiva)
$k(n-1)^2 + cn =$
$k(n^2 + 1 - 2n) + cn =$
$kn^2 + k - 2kn + cn$
Imponendo la disuguaglianza:
$kn^2 + k - 2kn + cn \leq kn^2$
$\Leftrightarrow$ $k - 2kn + cn \leq 0$

$\Leftrightarrow$ $2kn - k \geq cn$

Questa disuguaglianza è vera per $k = c$ e $n \geq 1$

Alberi da visite

Relazioni di ricorrenza per CostrAB_aus.

$T(n) = T(n/2) + T(n/2) + cn/2$ che si semplifica con
$T(n) = 2T(n/2) + cn$
Sappiamo che $T(n) = \Theta(n \lg n)$.