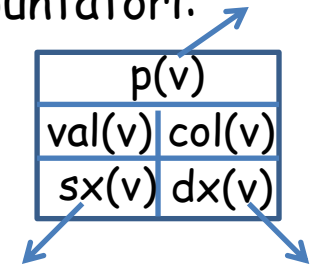
Esercitazione
18 novembre 2025

Algoritmi su alberi

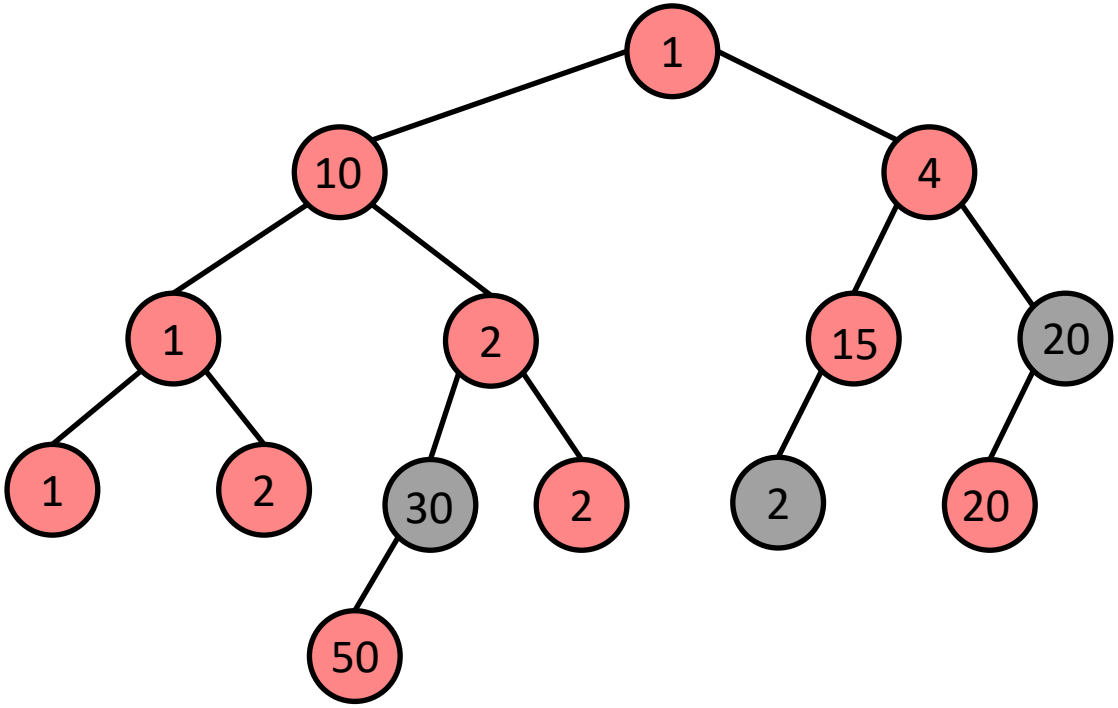
problema 1

Input: un albero binario T di n nodi
ogni nodo v ha:
-valore val(v)>0
-colore col(v)∈{R,N}

T rappresentato con record e puntatori:



Output: valore del cammino rosso di tipo radice-nodo di valore massimo

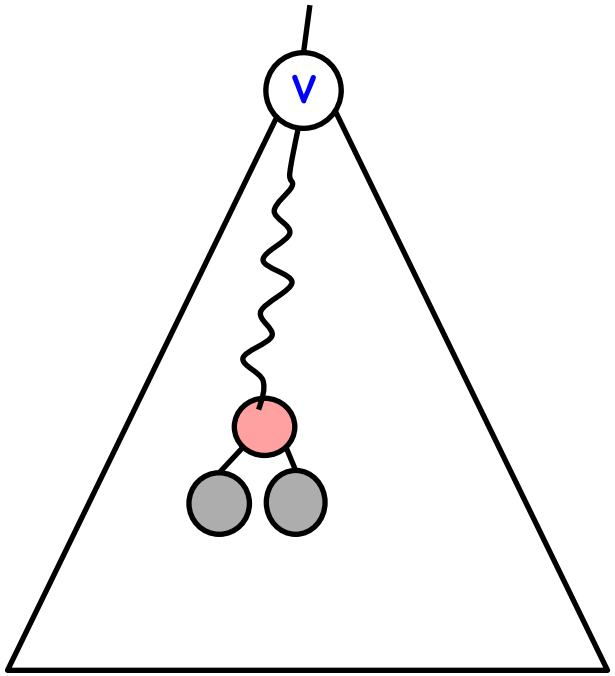


Def.: il valore di un cammino è la somma di valori dei nodi del cammino

Def.: un cammino è rosso se tutti i suoi nodi sono di colore rosso.

Output: 20

Idea



MaxRosso(v)

Restituisce il valore del cammino rosso di valore massimo di tipo v-discendente di v

-info che vengono "dal basso", calcolate rispetto al sottoalbero con radice v;
-possono essere usate per "passare informazioni" al padre di v

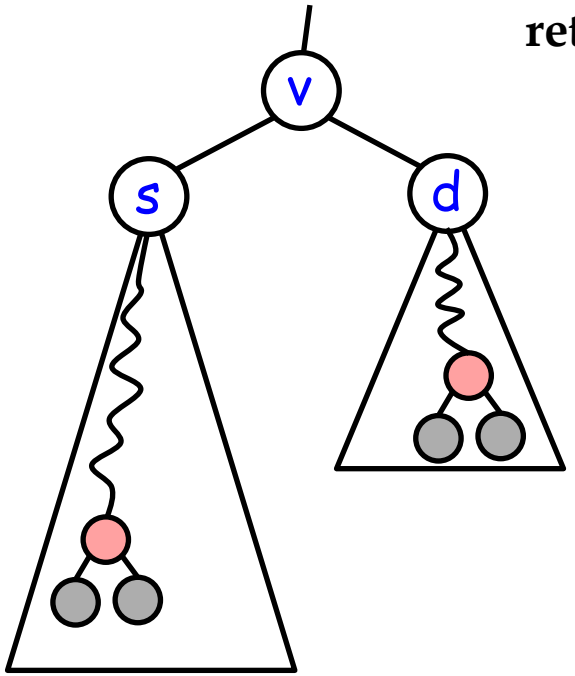
Idea

MaxRosso(v)

if v=null **then return** 0

if col(v)=N **return** 0

return val(v)+max{MaxRosso(sx(v)), MaxRosso(dx(v))}



MaxRosso(v)

Restituisce il valore del cammino rosso di valore massimo di tipo v-discendente di v

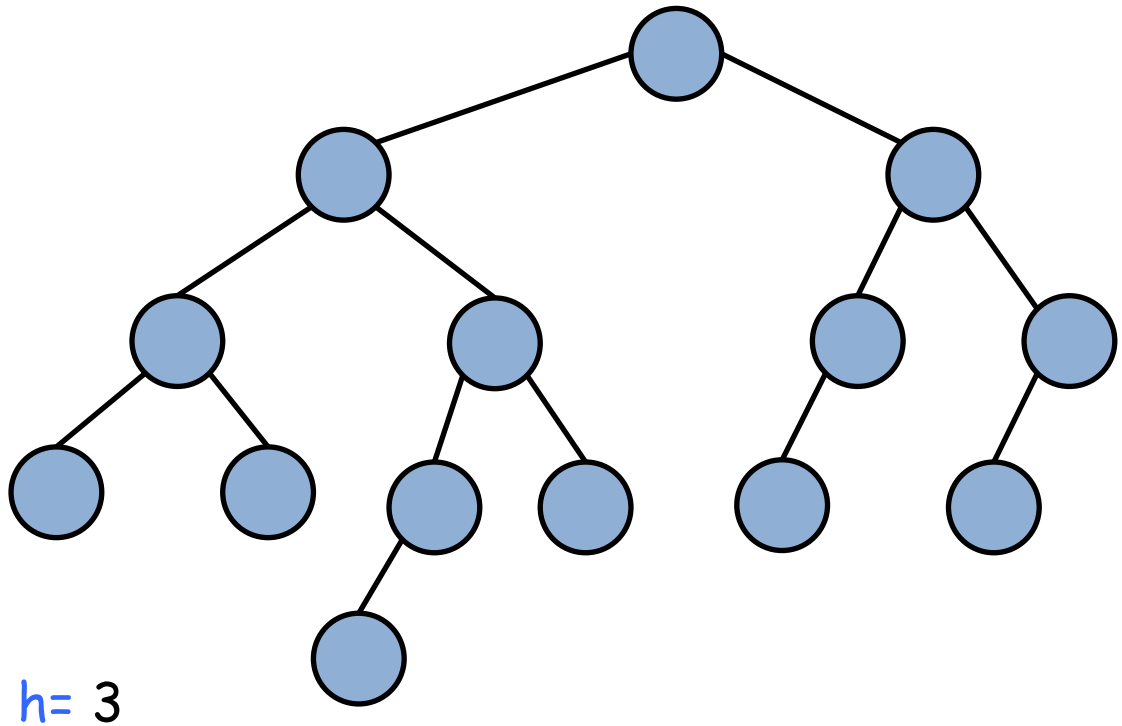
-info che vengono "dal basso", calcolate rispetto al sottoalbero con radice v;
-possono essere usate per "passare informazioni" al padre di v

Complessità:
O(n)

problema 2

Input: - un albero binario T di n nodi
(rappresentato con record e puntatori)
- un intero $h \geq 0$

Output: numero di nodi di T con profondità almeno h

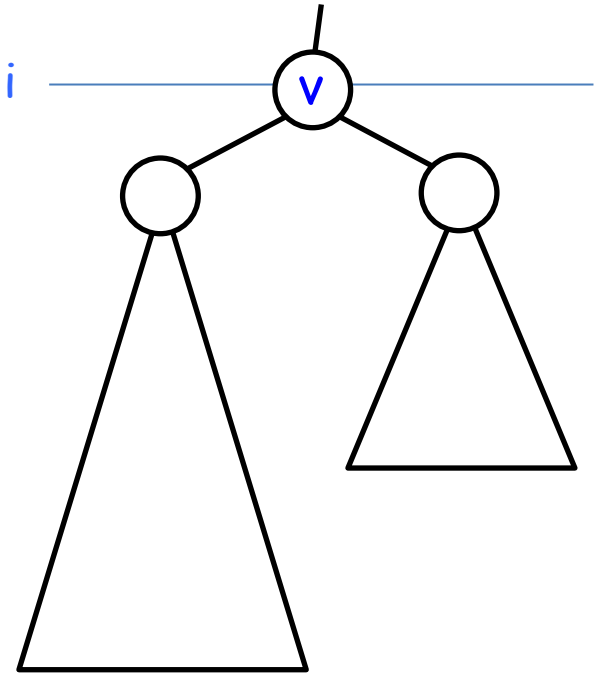


h= 3

Output: 7

profondità di un nodo:
distanza (#di archi) dalla radice.

Idea



info che vengono "dall'alto"

ContaProf(*v*, *h*, *i*)

restituisce il numero di nodi nel sottoalbero radicato in v che hanno prof ≥ h.
assume che v ha profondità i

info che vengono "dal basso"

ContaProf(*v*, *h*, *i*)

if *v*=null **then return** 0

if $i \geq h$

then return 1+ ContaProf(sx(*v*), *h*, *i*+1)
+ContaProf(dx(*v*), *h*, *i*+1)

else return ContaProf(sx(*v*), *h*, *i*+1)
+ContaProf(dx(*v*), *h*, *i*+1)

chiamata iniziale:

ContaProf(*r*, *h*, 0)

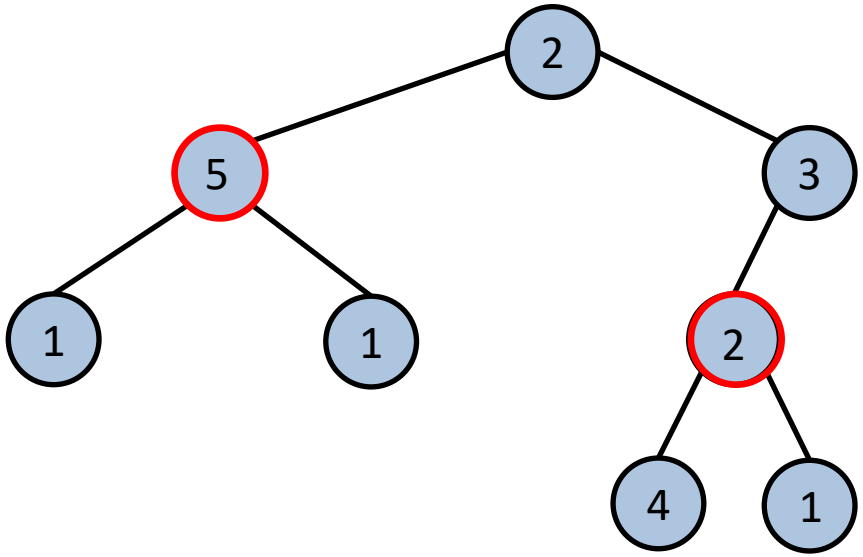
Complessità:
O(n)

problema 3

Input: - un albero binario T di n nodi
(rappresentato con record e puntatori)
- ogni nodo v ha un valore val(v)>0

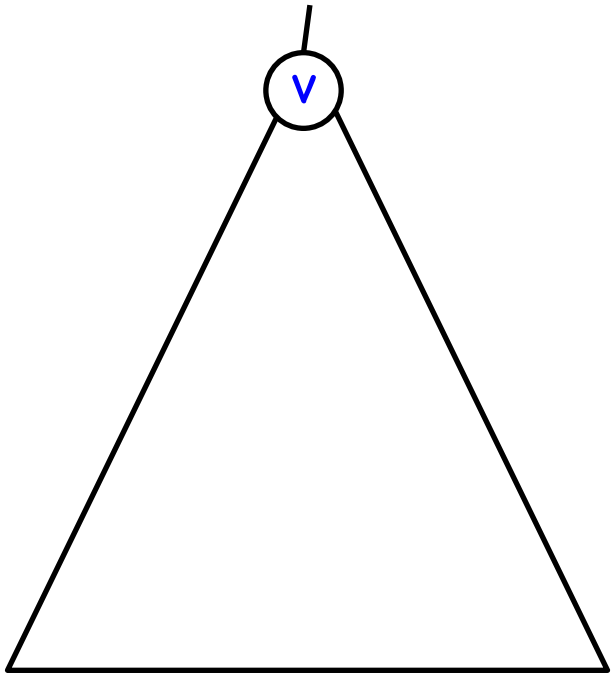
Output: numero di nodi che soddisfano

somma dei valori degli antenati del nodo = somma dei valori dei discendenti del nodo } Δ



Output: 2

Idea

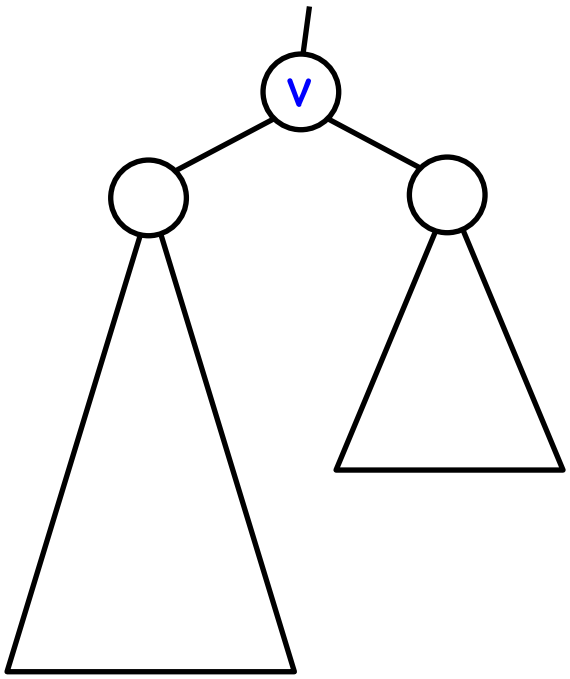


parametri problema | info "dall'alto" | info "dal basso"

alg(v,… , …, …, …) → …

output finale:

alg(*radice*,… , …,…, …)



Bilanciati(v, SA)

restituisce (SD,k)
-SD: somma valore discendenti di v (v incluso);
-k: numero nodi nel sottoalbero radicato in v che soddisfano (Δ).
assume:
-SA: somma valori antenati di v (v escluso)

Bilanciati(v, SA)

if v=null **then return** (0,0)

SA=SA+val(v)

(SD_s,k_s)=Bilanciati(sx(v),SA)

(SD_d,k_d)=Bilanciati(dx(v),SA)

SD=SD_s+SD_d+val(v)

if SD=SA

then return (SD, 1+ k_s+ k_d)

else return (SD, k_s+ k_d)

CalcoBilanciati(r)

(SD,k)=Bilanciati(r,0)

return k

Complessità:
O(n)

Esercizio

Sia T un albero binario di n nodi con radice r in cui ogni nodo ha un valore non negativo associato. La *profondità* di un nodo v è il numero di archi del cammino da v alla radice. I nodi che si incontrano lungo tale cammino (v compreso) sono detti *antenati* di v. Diremo che un nodo v è *generazionalmente profondo* se la sua profondità è strettamente maggiore del valore di un suo antenato di valore minimo.

Si assuma che T è mantenuto attraverso una struttura collegata e che ogni nodo v abbia associato i seguenti campi: puntatori al padre e ai figli ($v.p$, $v.s$, $v.d$) e valore del nodo ($v.val$). Si progetti un algoritmo con complessità temporale $O(n)$ che, preso T, restituisca il numero di nodi generazionalmente profondi di T. Si fornisca lo pseudocodice dettagliato dell'algoritmo.

Esercizio

Sia T un albero binario di n nodi con radice r. La *profondità* di un nodo v è il numero di archi del cammino da v alla radice. Un nodo u è un *discendente* di v se u si trova nel sottoalbero di T radicato in v. Si assuma che T è mantenuto attraverso una struttura collegata e che ogni nodo v mantenga i puntatori al padre e ai figli ($v.p$, $v.s$, $v.d$). Si progetti un algoritmo con complessità temporale $O(n)$ che, preso T, restituisca il nodo v di profondità minima la cui profondità è maggiore o uguale al numero dei suoi discendenti. Si fornisca lo pseudocodice dettagliato dell'algoritmo e possibilmente non si usino variabili globali e passaggi di parametri per riferimento.