Lezione 6/4/2020

Monday, April 6, 2020 10:58 AM

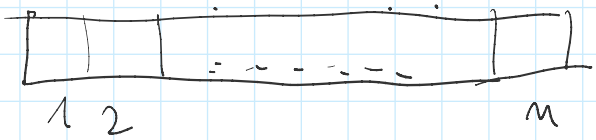
Alberi

- alberi di decisione — limiti inferiori
- alberi per descrivere la struttura ricorsiva di un algoritmo — MergeSort Torneo
- alberi di ricorsione per risolvere eq. ricorrenze
- Heap per realizzare
  - code di priorità
  - HeapSort

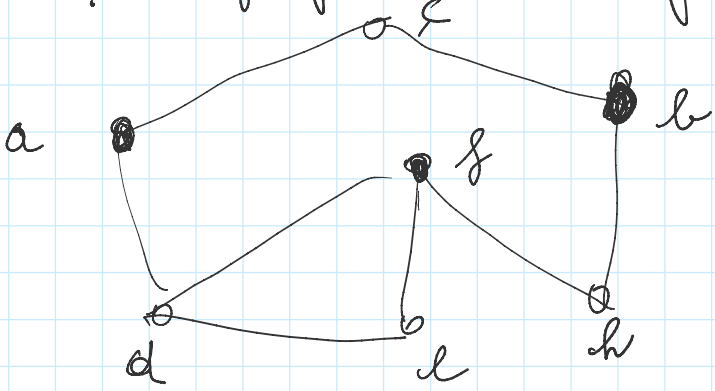
I dati su cui l'alg. opera sono degli alberi ?

- classificazioni
- gerarchie

dati + relazioni



Albero? Grafo connesso privo di cicli



$G = (V, E)$

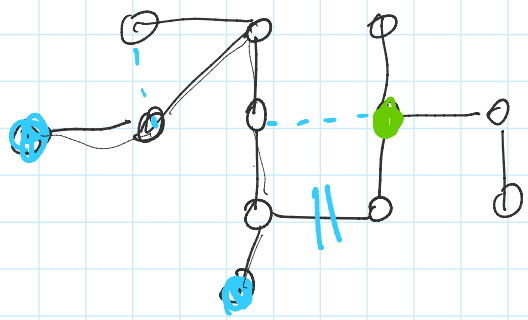
a c b h f e d a

d e f d

a c b d h f d a

connesso: se per ogni coppia di nodi esiste un cammino che li unisce

privo di cicli: (un ciclo è un cammino che inizia e termina su un nodo) non ha cicli



albero libero

grafo connesso privo di cicli

n nodi

n-1 archi

12 nodi

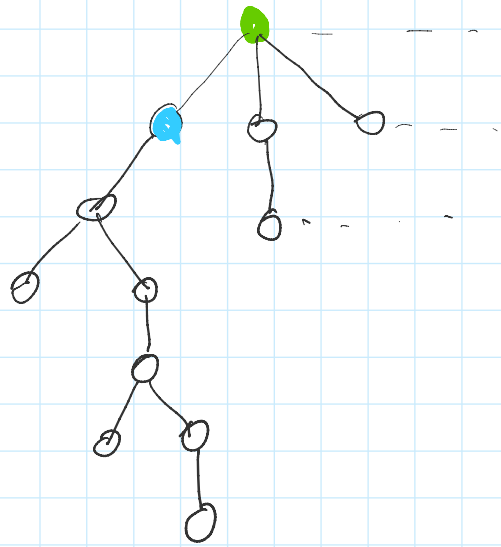
11 archi

n - 1 archi

11 archi

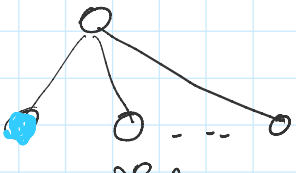
- se tolgo un arco → l'albero non è più connesso
- se aggiungo un arco → formo un ciclo

Albero con radice : ordinamento per livelli



profondità 1
altezza 6

Alberi ordinati



Alberi binari (k-ari) $K=2$.

definizione ricorsiva dell'albero T

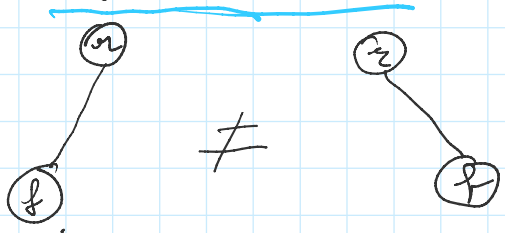
- T è vuoto
- T è composto da 3 sottoinsiemi
  - radice

- T è composto da 3 sottoinsiemi
  - radice
  - sottoalbero sinistro (alb. binario)
  - sottoalbero destro (alb. binario)

alberi ordinati

f è il primo figlio

alberi binari



figlio sin figlio des

Noi studieremo la famiglia degli **alberi binari**.

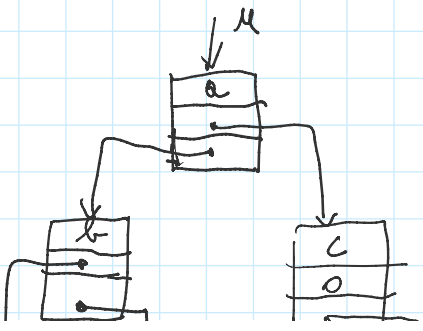
Metodo automatico per trasformare alberi ordinati ⇒ alberi binari

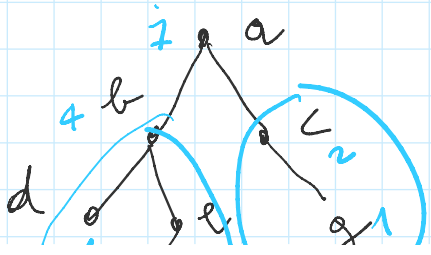
n° figli variabile

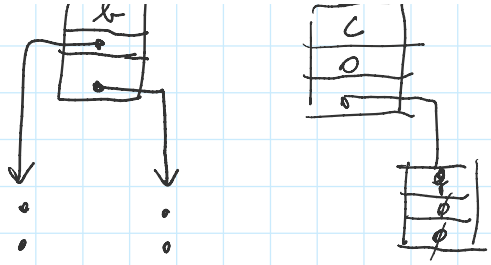
Struttura più complicata

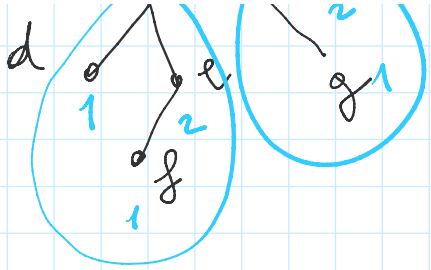
Memorizzazione alberi binari

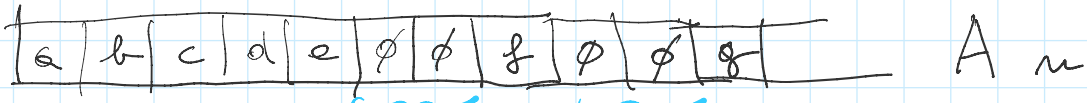
nodo u
- u.dato
- u.sx puntatore al sott. sin
- u.dx " al sott destro



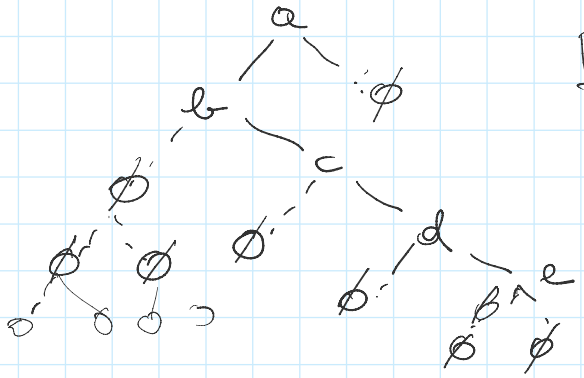


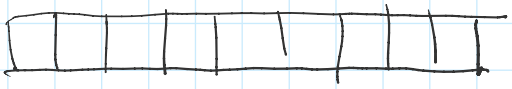






A n



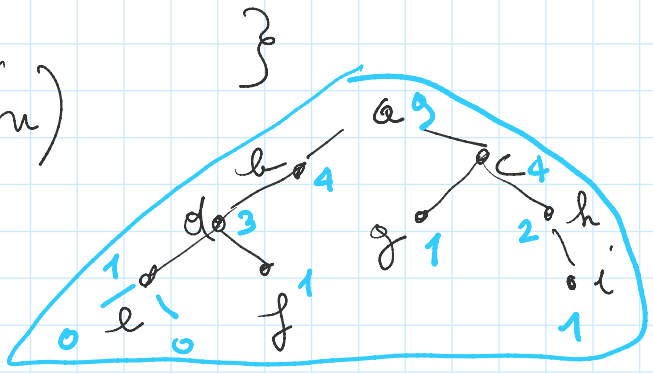


Quanto può essere lo spreco di memoria in funzione di n?

Non conosciamo la dimensione dell'albero. Calcoliamo la dimensione = numero di nodi di un albero binario di puntatore u.

```
Dimensione (u):
condizione terminaz.   if (u == Null) return 0;        Θ(1)
                       else {
chiamate ricorsive     dim_sx = Dimensione (u.sx);
su sottoalb.           dim_dx = Dimensione (u.dx);
Ricombinazione         return (dim_sx + dim_dx + 1);
                       }
```

$\Theta(n)$



$\Theta(1)$

$u = u.sx$

$u = u.sx$

$u = u.sx$

nodo e

$u = u.sx = \text{NULL}$

$u = u.sx = \text{NULL}$

$1 + \emptyset + \emptyset = 1$

$$T(n) = \Theta(1) \quad \text{se } n = 0$$

$$T(n) = 2\,T\left(\frac{n}{2}\right) + \Theta(1) \quad ?$$

$$T(n) = T(n_s) + T(n_d) + \Theta(1)$$

n_s = n° nodi sott. sin

n_d = n° nodi sott. des.

$n = n_s + n_d + 1$

$$T(n) = T(n_s) + T(n_d) + 1$$

<u>trial and error</u> fai un'ipotesi e dimostra che è vera.

$$T(n) = \Theta(n) = c\,n$$

dimostriamo per induzione generalizzata su n.

base : $n = 1$ $\quad \Theta(1)$

ipotesi induttiva :

posto vero per tutti gli $n' < n$

dimostro vero per n.

vero per n_s

vero per n_d

$$T(n) = \Theta(n_s) + \Theta(n_d) + 1 =$$

$$= \Theta(n_s + n_d + 1) = \Theta(n)$$

$O(n)$ $\quad$ oss. bil ho ~~$O(\log n)$~~

$$T(n) = 2T\left(\frac{n}{2}\right) + 1 = \Theta(n)$$

$n^{\log_2 2} = n^1 \qquad f(n) = n^0$

caso 1

Schema di DetI che si applica a alberi binari

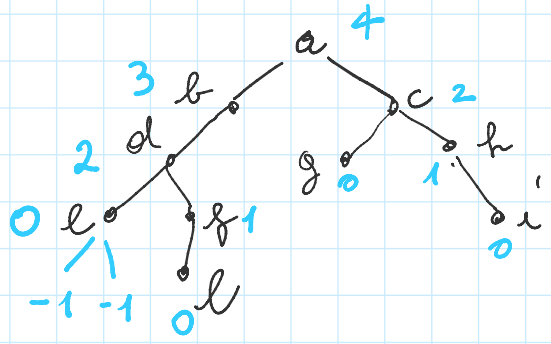
```
Decomponibile (u):
if (u == null) return Decomponibile (null);
else {
    risSx = Decomponibile (u.sx);
    risDx = Decomponibile (u.dx);
```

```
    risdx = Decomponibile(u.dx);
    return ricombina(rissx, risdx);
}
```

Problema: calcolo l'altezza di un albero T di puntatore u.

altezza di T = altezza della radice di T

altezza di un nodo = è la distanza in numero di archi della foglia più lontana

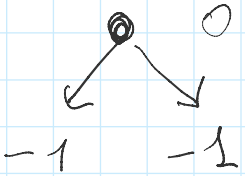


x nodo non foglia

$\max(alt_{sx}, alt_{dx}) + 1$

x è foglia $\rightarrow$ 0

sottoalbero = ϕ $\Rightarrow$ -1



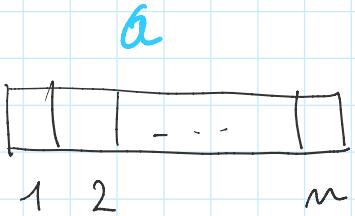

```
Altezza(u):
if (u == null) return -1;   Θ(1)
```

```
if (u == null) return -1;        Θ(1)
else {
    altsx = Altezza(u.sx);
    altdx = Altezza(u.dx);
    return (max(altsx, altdx) + 1);}      Θ(1)
```

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

Stampa di un array

```
for(i = 1; i<=n; i++)
    stampa(a[i]);
```



Stampa di tutti gli elementi di un albero binario.

occorre stabilire un ordine di percorrenza. Visite

1) • Visita anticipata, Pre-visita
PREVISIT

2) • Visita simmetrica, In visita
INVISIT

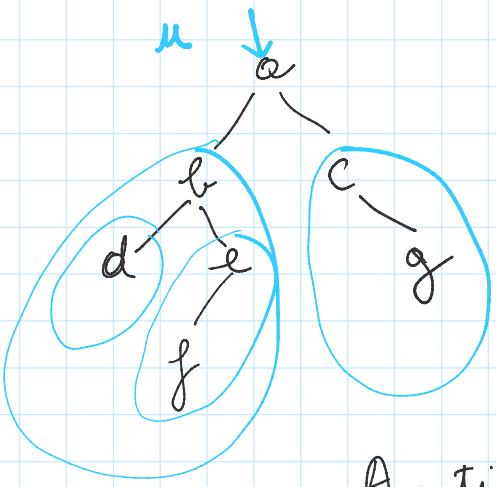
3) • Visita differita o posticipata
POSTVISIT

1) PRE VISIT

1) Esamina la radice
2) Visita il sottoalb. sinistro ricorsivamente
3) Visita il ~~sotto~~ alb. destro ricorsivamente

2) INVISITA
- Visita il sott. sin. ric
- Esamina la radice
- Visita il sott albero des. ric.

3) POSTVISITA
- - - -
- - -
- Esamina la radice



PRE: a b d e f c g
IN: d b f e a c g
POST: d f e b g c a

```
Anticipata(u):
if (u != NULL) {
    print u.dato;
```

$t(u) = \Theta(n)$

$t(n) = \Theta(n)$

```
        fermi u.dato;
        Anticipato (u.sx);
        Anticipato (u.dx);
    }
```