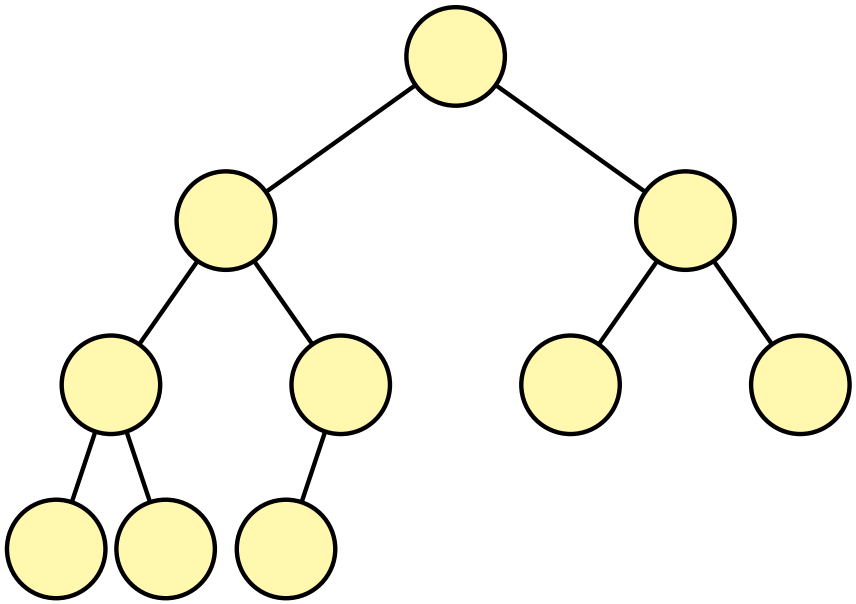
Famiglia

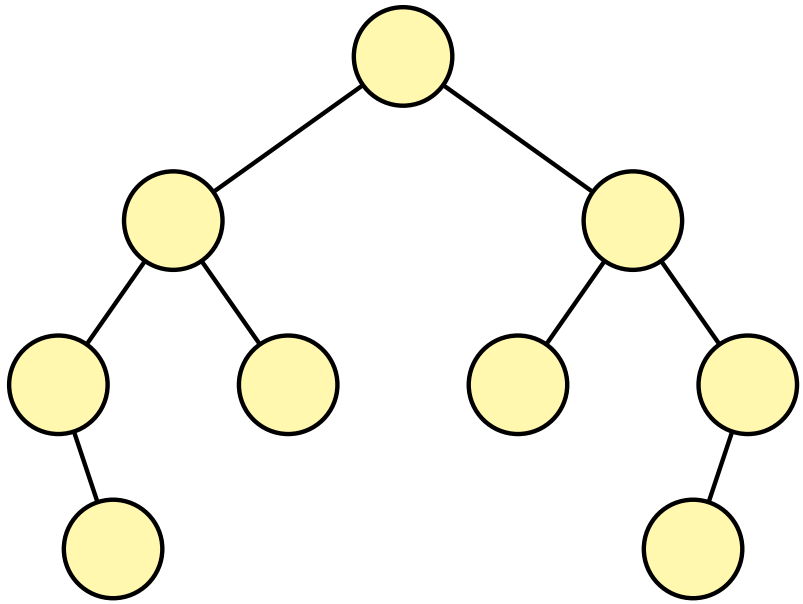
Trovare i limiti superiore e inferiore più stretti possibili per la seguente famiglia di equazioni di ricorrenza, per valori di a interi positivi.

$$T(n) = \begin{cases} aT(\lfloor n/2 \rfloor) + n^{a-1} & n \geq 2 \\ 1 & n < 2 \end{cases}$$

Alberi simmetrici

Scrivere un algoritmo **boolean** isSymmetric(Tree T) che prenda in input un albero binario T non vuoto e restituisca **true** se T è simmetrico, **false** altrimenti. Un albero binario è simmetrico se il sottoalbero sinistro della radice è un'immagine *speculare* del sottoalbero destro della radice. L'albero binario a sinistra non è simmetrico, mentre lo è quello di destra.





Discutere correttezza e complessità computazionale dell'algoritmo

Grafi – Pozzo universale

- Un pozzo universale è un nodo con out-degree uguale a zero e in-degree uguale a $n - 1$.
- Dato un grafo orientato G rappresentato tramite matrice di adiacenza, scrivere un algoritmo che opera in tempo $\Theta(n)$ in grado di determinare se G contiene un pozzo universale.
- È possibile ottenere la stessa complessità con liste di adiacenza?

Maggioranza

Scrivere una funzione

boolean hasMajority(**int**[ ] A, **int** n)

che prenda in input un vettore ordinato A contenente n interi e restituisca **true** se A contiene un valore di maggioranza, ovvero un valore che compare più di $n/2$ volte; restituisca **false** altrimenti. Soluzioni di costo computazionale lineare non verranno prese in considerazione.

Discutere correttezza e complessità dell'algoritmo proposto.

Analisi – Problema – $T(m) + T(n-m)$

Si consideri la seguente equazione di ricorrenza:

$$T(n) = \begin{cases} T(m) + T(n-m) + 1 & n > m \\ 1 & n \leq m \end{cases}$$

dove m è una costante intera positiva.

Utilizzando il teorema delle ricorrenze lineari di ordine costante, ottenere una stima della complessità.

Tramite il metodo della sostituzione, dimostrare che tale stima è corretta, un limite superiore ed un limite inferiore per $T(n)$. Fare particolare attenzione ai casi base.

Spoiler alert!

Famiglia (24/07/20)

Data la forma della ricorrenza, è possibile utilizzare il Master Theorem, versione base.

- Per $a = 1$, abbiamo $\alpha = \log_2 1 = 0$, $\beta = 0$; siamo nel secondo caso, $T(n) = \Theta(\log n)$.
- Per $a = 2$, abbiamo $\alpha = \log_2 2 = 1$, $\beta = 1$; siamo nel secondo caso, $T(n) = \Theta(n \log n)$.
- Per $a = 3$, abbiamo $\alpha = \log_2 3 < 2$, $\beta = 2$; siamo nel terzo caso, $T(n) = \Theta(n^2)$.
- In generale, è possibile dimostrare che $\log_2 a < a - 1$ per tutti i valori interi $a \geq 3$; siamo nel terzo caso e si ottiene $T(n) = \Theta(n^{a-1})$.

Alberi simmetrici (24/07/20)

Un sottalbero è simmetrico se i suoi sottoalberi destro e sinistro sono speculari.

Due sottoalberi t_1, t_2 sono speculari se e solo se:

- il sottoalbero sinistro di t_1 è speculare al sottoalbero destro di t_2
- il sottoalbero destro di t_1 è speculare al sottoalbero sinistro di t_1.

Il caso base è dato due nodi **nil** (si ritorna **true**) o da uno nodo **nil** e un nodo non **nil** (si ritorna **false**).

La procedura effettua una visita su entrambi gli alberi, e quindi ha complessità $\Theta(n)$.

Alberi simmetrici (24/07/20)

boolean isSymmetric(TREE T)

return isMirror($T.left, T.right$)

boolean isMirror(TREE t_{L}, TREE t_{R})

```
if t_L == nil and t_R == nil then
    return true
else if t_L ≠ nil and t_R ≠ nil then
    return isMirror(t_L.right, t_R.left) and isMirror(t_L.left, t_R.right)
else
    return false
```

Grafi – Pozzo universale

- Un pozzo universale è un nodo con out-degree uguale a zero e in-degree uguale a $n - 1$.
- Dato un grafo orientato G rappresentato tramite matrice di adiacenza, scrivere un algoritmo che opera in tempo $\Theta(n)$ in grado di determinare se G contiene un pozzo universale.
- È possibile ottenere la stessa complessità con liste di adiacenza?

Maggioranza (03/07/20)

Se esiste un valore di maggioranza, questo deve essere il valore contenuto nella posizione mediana $\lfloor (n+1)/2 \rfloor$.

- Se n è dispari, questo è l'elemento centrale. Qualunque maggioranza deve includere questo elemento e altri $(n-1)/2$ elementi; infatti, a sinistra e destra di tale elemento ci stanno esattamente $(n-1)/2$ elementi.
- Se n è pari, sia l'elemento in $n/2$ che l'elemento in $n/2+1$ devono appartenere alla maggioranza, per lo stesso ragionamento di cui sopra.

Dato il valore mediano m, si effettua ricerca dicotomica modificata che assuma l'esistenza di m e restituisca l'indice più basso in cui m si presenta.

Conoscendo l'indice più basso in cui m si presenta, è necessario vedere se l'elemento in posizione $A[\mathit{first} + \lfloor n/2 \rfloor]$ è uguale ad esso, nel qual caso esiste una maggioranza di elementi uguali, essendo il vettore ordinato

Maggioranza (03/07/20)

```
int hasMajority(int[] A, int n)
int candidate = A[⌊(n + 1)/2⌋])
int first = searchFirst(A, 1, n, candidate)
return A[first] == A[first + ⌊n/2⌋]
```

```
int searchFirst(int[] A, int i, int j, int v)
if i == j then
    return i
else
    int m = ⌊(i + j)/2⌋
    if v ≤ A[m] then
        return searchFirst(A, i, m, v)
    else
        return searchFirst(A, m + 1, j, v)
```

Ricorrenza $T(n) = T(m) + T(n-m) + 1$

- Ricordate che m è un valore costante;
- Il numero di chiamate ricorsive è $\lfloor n/m \rfloor$;
- Ogni chiamata costa $T(m) + 1$, ancora un valore costante.

Supponiamo quindi che $T(n) = \Theta(n)$.

Ricorrenza $T(n) = T(m) + T(n-m) + 1 = O(n)$

Passo induttivo: supponiamo per ipotesi induttiva che $T(k) \leq ck$ per ogni $k < n$. Proviamo che $\exists c > 0 : T(n) \leq cn$.

$$\begin{aligned} T(n) &= T(m) + T(n-m) + 1 \\ &\leq 1 + cn - cm + 1 \\ &\leq cn - cm + 2 \\ &\overset{?}{\leq} cn \end{aligned}$$

La disequazione è vera per $c \geq 2/m$; poichè $m \geq 1$, $c \geq 2$ è un valore accettabile per ogni m.

Ricorrenza $T(n) = T(m) + T(n-m) + 1 = O(n)$

Caso base: consideriamo tutti i casi per cui $1 \leq i \leq m$:

$$T(i) = 1 \leq c \cdot i \Leftrightarrow c \geq \frac{1}{i}$$

Le disequazioni su c derivanti dal caso base possono essere riassunte dal fatto che $c \geq 1$, poichè $1/i \leq 1$ per tutti i valori $i \geq 1$.

Considerando quindi sia la disequazione sul passo ricorsivo che sul caso base, otteniamo che $c \geq 2$.

Ricorrenza $T(n) = T(m) + T(n-m) + 1 = \Omega(n)$

Passo induttivo: supponiamo per ipotesi induttiva che $T(k) \geq ck$ per ogni $k < n$. Proviamo che $\exists c > 0 : T(n) \geq cn$.

$$\begin{aligned} T(n) &= T(m) + T(n-m) + 1 \\ &\geq 1 + cn - cm + 1 \\ &\geq cn \end{aligned}$$

L'ultima disequazione è banalmente vera per ogni $c \leq 2/m$.

Caso base: consideriamo tutti i casi per cui $1 \leq i \leq m-1$:

$$T(i) = 1 \geq c \cdot i \Leftrightarrow c \leq \frac{1}{i}$$

Il valore $\frac{1}{m}$ è il più piccolo fra tutte le disequazioni trovate; per soddisfarle tutte, compresa quella derivante dal passo induttivo, basterà porre $c = 1/m$.