29/11

Domanda A (7 punti) Dare la definizione della classe $\Theta(f(n))$. Mostrare che la ricorrenza

$$\left[T(n) = \frac{3}{4}T(n/3) + T(2n/3) + 2n\right] \quad \left[a\,T\left(\frac{n}{b}\right) + f(n)\right]$$

MT

ha soluzione in $\Theta(n)$.

$$c\,d(n) \le f(n) \le \dots$$

$$[T(n) \le cn] \to \frac{3}{4}c\left(\frac{n}{3}\right) + c\left(\frac{2n}{3}\right) + 2n \le c(n)$$

$$\frac{3}{4}T\left(\frac{n}{3}\right) + T\left(\frac{2n}{3}\right) + 2n \le c\left(\frac{n}{3}\right)$$

$$\frac{3}{4}c\left(\frac{n}{3}\right) + c\left(\frac{2n}{3}\right) + 2n \le c\left(\frac{n}{3}\right) \dots$$

$$\le \; (\dots) \quad \text{RISOLUZIONE}$$

$$\left[\frac{11}{12}cn\right] + \underbrace{2n}_{\forall n \ge 1} \le c\left(\frac{n}{3}\right) \qquad \left[\begin{array}{l}\forall c \ge 0 \\ \forall n \ge 0\end{array}\right]$$

$$\left[\frac{11}{12}cn \le c\,\frac{n}{3}\right]$$

$$c \to 24 \checkmark$$

$$\Theta \begin{cases} T(n) \le c(n) \\ T(n) \ge d(n) \end{cases}$$

$$T(n) = \frac{3}{4}T(n/3) + T(2n/3) + 2n$$

2

SOTTO 1
CERTA ...
SOPRA 2

$T(n) \geq d(n) \leftarrow$

$T(n) \geq d(n) \leftarrow$ RISOLVI PER UN "d"

VALE n, d

$\frac{3}{4} d\left(\frac{n}{3}\right) + d\left(\frac{2n}{3}\right) + 2n \geq d(n)$

$T(n) \rightarrow d(n)$

$\frac{3}{4} d \frac{n}{3} + 2d\frac{n}{3} + 2n \geq dn$

$T(n) \geq d(n)$

$\frac{dn}{4} + 2d\frac{n}{3} + 2n \geq dn$

$12 \quad \frac{3dn + 8dn + 24n \geq 12dn}{12}$

$11dn + 24n \geq 12dn$

LA SOLUZIONE ... $\leftarrow \begin{bmatrix} d \geq 24 \\ \forall n \geq 0 \end{bmatrix} \leftarrow \frac{24n}{n} \geq \frac{dn}{n}$

CASO 1
CASO 2
CASO 3

$\lim_{n \to \infty} \frac{f(n)}{\log_n^{b_a}}$

$\infty \rightarrow 1$

$K \rightarrow 2$

$\rightarrow \emptyset \rightarrow 3$

$\emptyset$ (INFINIT.)

REG.

$$a f\left(\frac{n}{b}\right) \le k f(n) \equiv \text{CASO 3}$$

$\Uparrow$

CALCOLOSO ...

(ALGEBRICAMENTE)

Domanda B (6 punti) Calcolare la lunghezza della longest common subsequence (LCS) tra le stringhe *store* e *shoes*, calcolando tutta la tabella $L[i, j]$ delle lunghezze delle LCS sui prefissi usando l'algoritmo visto in classe.

LCS → STORE / SHOES

ABCD / BC → 2

LEGENDA

~~ = MATCH (NON CONTIGUO)

→ ≤ ≥ ! (NO) MATCH

⊔ = MATCH

		S	H	O	E	S
	0	0	0	0	0	
S	0	1	1	1	1	1
T	0	1	1	1	1	1
O	0	1	1	2	2	2
R	0	1	1	2	2	2
E	0	1	1	2	3	3

[PROG. DINAMICA]

[SOL.]

↑ MATRICE

SOLUZIONE → BACKTRACKING

↖ = MATCH

↑ = NO MATCH DA ME, MA PRIMA SI

← = MATCH VICINO

TOP-DOWN ↓

	s	h	o	e	s
0	0	0	0	0	0
s 0	1	1	1	1	1
t 0	1	1	1	1	1
o 0	1	1	2	2	2
r 0	1	1	2	2	2
e 0	1	1	2	3	3

→ TORNARE INDIETRO ≡ TROVA DOVE È STATO IL MATCH!

Esercizio 2 (9 punti) Supponiamo di avere un numero illimitato di monete di ciascuno dei seguenti valori: 50, 20, 1. Dato un numero intero positivo n, l'obiettivo è selezionare il più piccolo numero di monete tale che il loro valore totale sia n. Consideriamo l'algoritmo greedy che consiste nel selezionare ripetutamente la moneta di valore più grande possibile.

(a) Fornire un valore di n per cui l'algoritmo greedy *non* restituisce una soluzione ottima.

(b) Supponiamo ora che i valori delle monete siano 10, 5, 1. In questo caso l'algoritmo greedy restituisce sempre una soluzione ottima: dimostrare che ogni insieme ottimo M^* di monete di valore totale n contiene la scelta greedy.

[50] - 20 - 1 [MONETE]

n = NUMERO

min Σ (INSIEME MINIMO) = n

GREEDY — MIN / MAX (SCELTA INTELLIGENTE)

[INSIEME DI MONETE]

ALGORITMO → PRENDI QUELLA CON VAL. MAX → NON OTTIMA! DIMOSTRA (INPUT FALLISCE)

MONETE → . . . . .

1 2 3 4 $\boxed{5}$ → $\boxed{5}$

→ PIÙ PICCOLO INSIEME È 5

[50 20 1]

↑

INTERMEDIO

20

5 (NON AVREBBE SENSO)

n = 25 = 20 + (5 · 1)

$50 = \underset{1}{20}\ \underset{1}{20}\ \underset{2}{5}$.

(a) Per esempio $n = 60$, perché la soluzione ottima è 3 monete da 20, mentre l'algoritmo greedy restituisce 11 monete (una da 50 e 10 da 1).

60 → 2 / 3 / 5 / 10 / 20 ....

GREEDY = INPUT COSTRUITO

(MULTIPLO) → 3 MONETE DA 20 (✓ GIUSTO)

→ $\underbrace{50}$ + (11 · 1)

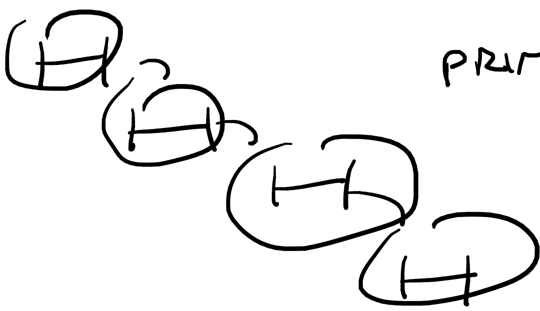
GRANDE (MAX)

PIÙ MONETE

GREEDY → OTTIMO

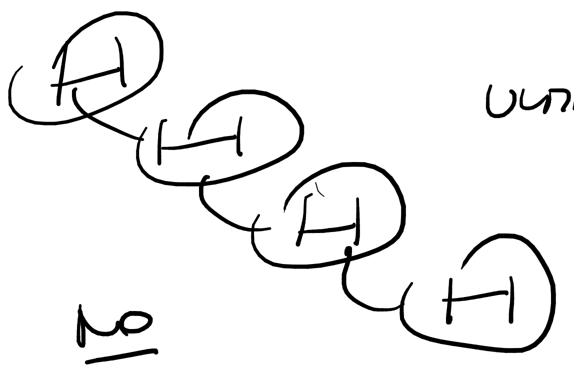
NON OTTIMO = FORNIRE CONTROESEMPI!

SELEZ. ATTIVITÀ →



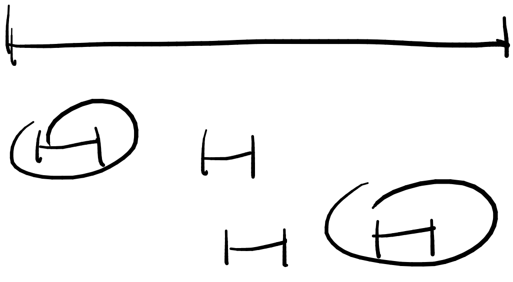
PRIMA ↓

OTTIMO!



ULTIMA ↑

✓ NO



→ NON OTTIMO (SELECT MAX)

(b) Supponiamo ora che i valori delle monete siano 10, 5, 1. In questo caso l'algoritmo greedy restituisce sempre una soluzione ottima: dimostrare che ogni insieme ottimo $M^{\star}$ di monete di valore totale n contiene la scelta greedy.

$M^* = OPT$

$\underbrace{10}_{\text{MAX}} \quad 5 \quad 1$

$M^* = M^* \cup \{\underbrace{OPT}_{\text{MAX}}\}$

MAX $\quad M^*(+1) = \text{MAX}$

$= M^* \cup \{OPT\}$

$\cup \{OPT\_LOCAL\}$

CUT AND PASTE → MIN

MAX + MIN → NO MAX

MIN + MIN → NO MAX

(b) Sia $M^{\star}$ una soluzione ottima. Sia x il valore maggiore tra 10, 5, e 1 che sia non superiore a n. Se $M^{\star}$ contiene una moneta di valore x, la proprietà è dimostrata. Altrimenti, sia $M \subseteq M^{\star}$ un insieme di (2 o più) monete di valore totale x (si osservi che tale insieme esiste sempre quando i valori delle monete sono 10, 5, 1); consideriamo $M' = M^{\star} \setminus M \cup X$, dove X è l'insieme contenente una moneta di valore x. M' è un insieme di monete di valore totale n e di cardinalità inferiore a quella di $M^{\star}$: assurdo, quindi questo secondo caso non può verificarsi, e quindi $M^{\star}$ contiene necessariamente una moneta di valore x.

ALGORITMO → TI DÀ SEMPRE IL MAX GIUSTO!

S_0 → SOL → NON PARTE DI M^*

$M^* = M^* \setminus \{M \cup X\}$

X = SCELTA ATTUALE

CON MONETE PRESE DA M^*,

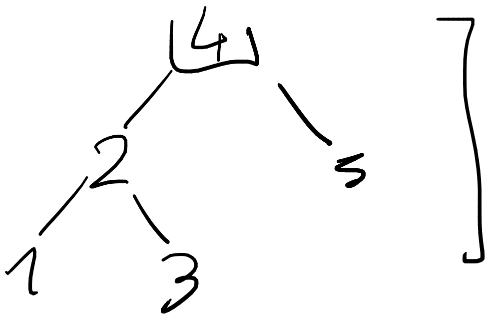
LA SOL. NON FUNZIONA!

OTTIMA! $= \underbrace{20 \cdot 3}_{M^*}$ $\qquad \underbrace{50 + (10 \cdot 1)}_{M^* = M^* \setminus x}$

Esercizio 1 (7 punti) Realizzare una procedura `BST(A)` che dato un array `A[1..n]` di interi, ordinato in modo crescente, costruisce un albero binario di ricerca di altezza minima che contiene gli elementi di `A` e ne restituisce la radice. Fornire un'argomentazione che supporti il fatto che l'albero ha altezza minima. Per allocare un nuovo nodo dell'albero si utilizzi una funzione `mknode(k)` che dato un intero `k` ritorna un nuovo nodo con `x.key=k` e figlio destro e sinistro `x.left = x.right = nil`. Valutarne la complessità.

- ARRAY A → [1, 2, 3, 4, 5]

- BST →

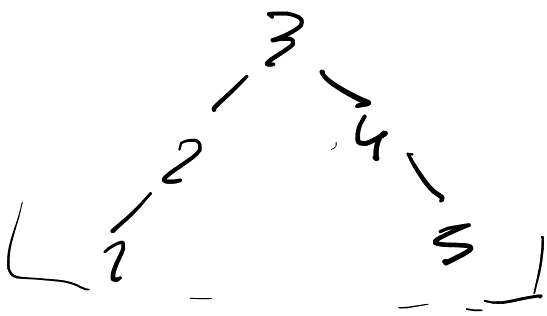


ALT. MINIMA

1, 2, $\boxed{3}$, 4, 5 → BST(A)

PIVOT

ORD. CRESCENTE



BST (A)

RETURN BST-REC (A, 1, n)

BST_REC (A, p, r)

1 2 [3] 4 5

$\underbrace{1}_{p}$ 2 $\underbrace{[3]}_{q}$ 4 $\underbrace{5}_{r}$

$q = \lfloor p + r \rfloor / 2$ [FLOOR]

x = MKNODE (n) → A[q] = x

x.L = BST_REC (A, p, q − 1)

x.R = BST_REC (A, q + 1, R) ...

```
BST(A)
   return BST-rec(A,1,n)

BST-rec(T,A,p,q)
   if p <= q
      m = floor(p+q/2)
      x=mknode(A[m])
      x.l = BST-rec(A,p,m-1)
      x.r = BST-rec(A,m+1,q)
    else
      x = nil

    return x
```

$\lceil \quad \rceil$

CEILING

Si può dimostrare, per induzione su n, che l'altezza dell'albero generato è $\lceil \log_2(n+1) \rceil$, quindi è la minima possibile.

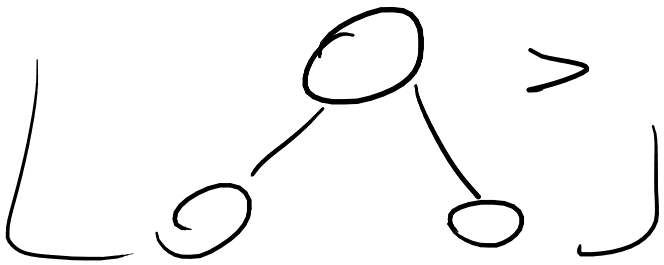
$$T(n) = 2T\left(\frac{n}{2}\right) + c$$

$\lfloor \quad \rfloor$ FLOOR

$$\approx = \Theta(n) \approx O(\lceil \log_2(n) + 1 \rceil)$$

Domanda B (6 punti) Dare la definizione di max-heap. Dato l'array `A` con elementi `7, 1, 17, 0, 5, 4, 22,` si specifichi il max-heap ottenuto applicando ad `A` la procedura `BuildMaxHeap`. Si descriva sinteticamente come si procede per arrivare al risultato (ad esempio illustrando come opera `BuildMaxHeap` e riportando il contenuto dello heap nei passi intermedi).

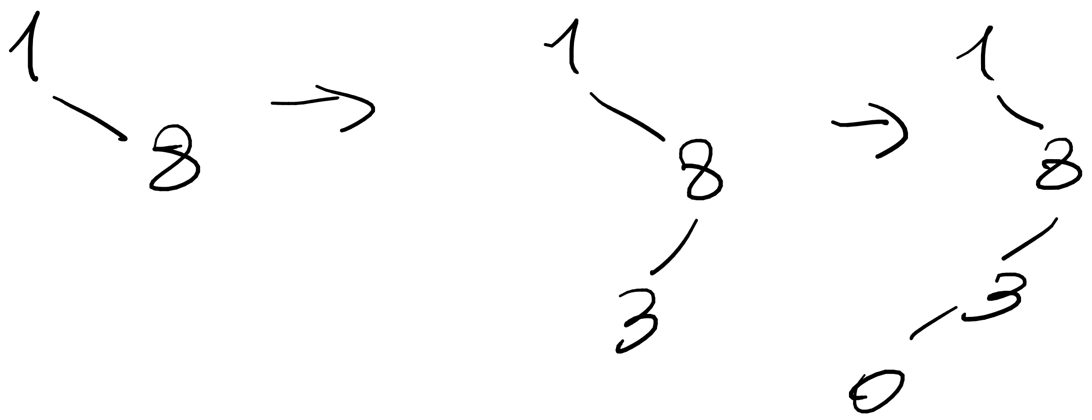
MAX-HEAP



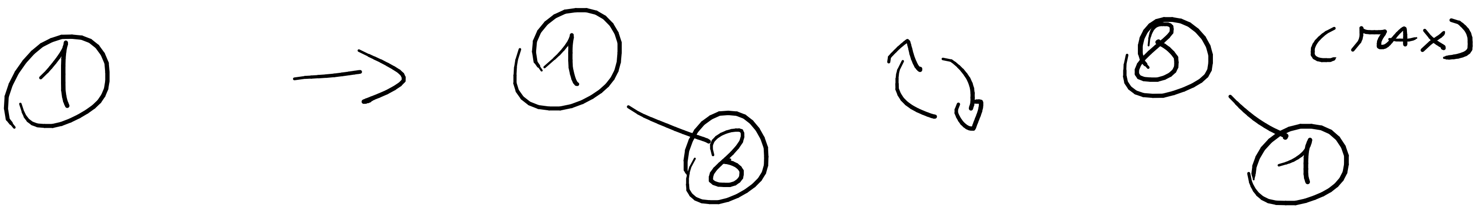
BST != HEAP

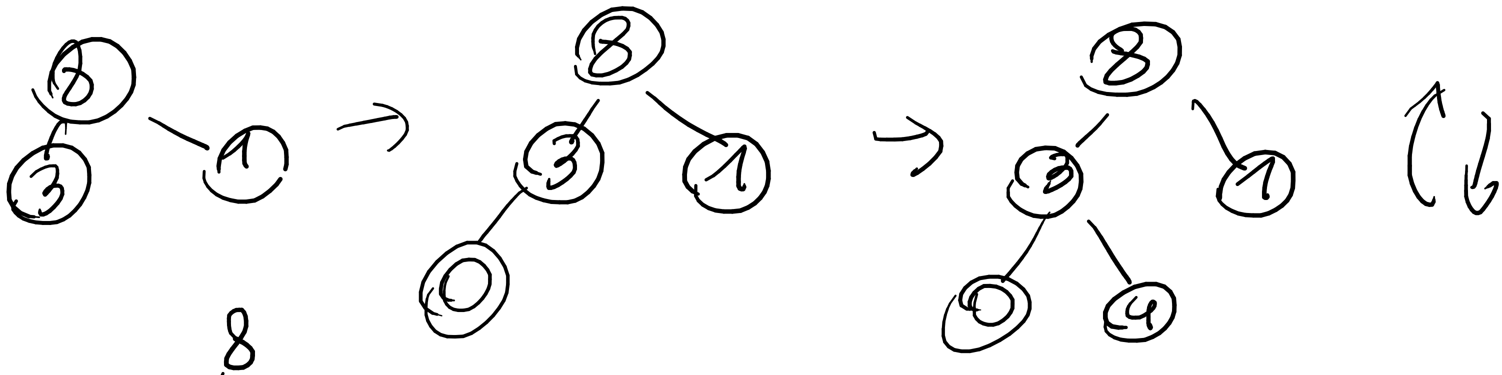
[1, 8 3 0 4

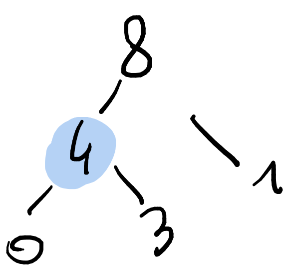
(BST)



1 8 3 0 4 → MAX HEAPIFY

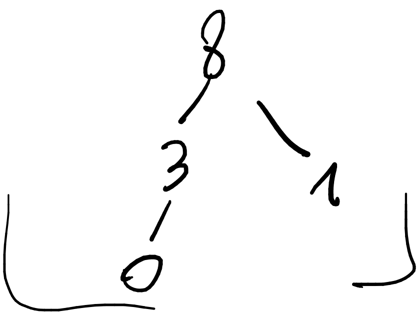






(HEAP) ...

DELETE 4



→ STABILISCE LA PROPR. DELLA STRUCT.

Domanda B (6 punti) Si consideri una tabella hash di dimensione $m = 8$, e indirizzamento aperto con doppio hash basato sulle funzioni $h_1(k) = k \; mod \; m$ e $h_2(k) = 1 + 2(k \; mod \; (m-2))$. Si descriva in dettaglio come avviene l'inserimento della sequenza di chiavi: 13, 29, 19, 27, 8.

$h_1(k) = k \bmod m \qquad h_2(k) = 1 + 2(k \bmod (m-2))$

$(h_1(k) + i(h_2(k))) \bmod m$

$13 \rightarrow 13 \bmod 8 = 5 \qquad (8 \cdot 1) + 5 = 13$

Doppio hash $= h(k,i) = (h_1(k) + i * h_2(k)) \; mod \; m$ *(da sapere!)*

0
1
2 ATTUATA → COLLISIONE!
3
4
5 → 13 (29) !

$29 \bmod 8 \rightarrow 3 \cdot 8 = 24 + \underline{5} \rightarrow \bmod$

$[h_1(k) + 1 \cdot h_2(k)] \bmod m =$

$[29 \bmod 8 + 1 \cdot (1 + 2(29 \bmod 7))]$