

FIBONACCI 1

- USA LA FORMULA DI BINET
- $\Theta(1)$
- NON È CORRETTO

FIBONACCI 2

- RICORSIVA

$$\left\{ \begin{array}{l} T(m) = 2 + T(m-1) + T(m-2) \quad \text{se } m \geq 3 \\ T(1) = T(2) = 1 \end{array} \right.$$

$$\text{e } m = 1$$

FIBONACCI 3

- ITERATIVA
- UTILIZZO DI ARRAY
- $\Theta(n) \rightarrow 2m$

FIBONACCI 4

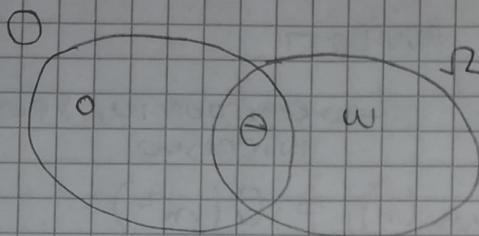
- ITERATIVA
- 3 VARIABILI
- $\Theta(n) \rightarrow 4n - 5$

FIBONACCI 5

- MATEMATICO
- ITERATIVO
- $\Theta(n) \rightarrow 2m + 1$

FIBONACCI

- MATRICI
- RICORSIVO
- EQUAZIONE DI CORRENZA: $T(n) \leq T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + c$
- $\Theta(\log n)$



RICERCA SEQUENZIALE

- ITERATIVA
- $T_{\text{seq}}(n) = 1 = \Theta(1) \rightarrow$ PRIMA POSIZIONE
- $T_{\text{seq}}(n) = n = \Theta(n) \rightarrow$ ULTIMA POSIZIONE
- $T_{\text{iterativo}}(n) = T_{\text{seq}}(n) = \Theta(n)$

RICERCA BINARIA

- ITERATIVA
 - PARTIRE DAL CENTRO E DIVIDERE IN SOVRAPPARTI
- $T_{\text{bin}}(n) = 1 = \Theta(1) \rightarrow$ ELEMENTO CENTRALE
- $T_{\text{bin}}(n) = \lceil \log n \rceil + 1 = \Theta(\log n) \rightarrow$ NON ESISTEGG
- $T_{\text{iterativo}}(n) = T_{\text{bin}}(n) = \Theta(\log n)$

SELECTION SORT

- ITERAZIONE \rightarrow 2 CICLI ANIDATTI

- $T_{\text{best}}(n) = \Theta(n^2)$

↳ CONFRONTO E SCAMBIO

- $T_{\text{worst}}(n) = \Theta(n^2)$

- $T_{\text{medio}}(n) = \Theta(n^2)$

INSERTION SORT 1

- ITERAZIONE \rightarrow 2 CICLI ANIDATTI

↳ CONFRONTO, SPOSTO, INSERISCO

- $T_{\text{best}}(n) = T_{\text{worst}}(n) = T_{\text{medio}}(n) = \Theta(n^2)$

INSERTION SORT 2 \rightarrow ORDINARE IN ORDINE CRESCENTE

- ITERAZIONE

- $T_{\text{best}}(n) = \Theta(n) \rightarrow$ già ordinato \rightarrow 1 SOLO CICLO

- $T_{\text{worst}}(n) = \Theta(n^2) \rightarrow$ si trovava in ordine decrescente

- $T_{\text{medio}}(n) = \Theta(n^2)$

INSERTION SORT 3

- UTILIZZO RICERCA BINARIA

- 2 CICLI

- $T_{\text{best}}(n) = \Theta(n \log n)$

- $T_{\text{worst}}(n) = \Theta(n^2)$

- $T_{\text{medio}}(n) = \Theta(n^2)$

MERGE SORT

- RICORSIVO
- CON IL Merge PONDE 2 SEQUENZE ORDINATE

Lo uso di UN ARRAY AUSILIARIO

$$T_{\text{BEST}}(n) = T_{\text{Worst}}(n) = T_{\text{medio}}(n) = \Theta(n \log n)$$

HEAP SORT → ORDINAMENTO IN LOCO

- UTILIZZO DI HEAP BINARIO

$$\text{Lo } h = \Theta(\log n)$$

Lo ALBERO BINARIO QUASI COMPLETO

- USO DEL ~~Max~~Heap

Lo CONTROLLA DALL'ALTO IN BASSO

$$T_{\text{BEST}}(n) = T_{\text{Worst}}(n) = T_{\text{medio}}(n) = \Theta(n \log n)$$

QUICKSORT

- DIVIDE et IMPERA

- UTILIZZO DI UN PERNO → DOPPIO SCORRIMENTO



S_x → D_x MAGGIORI
D_x → S_x MINORI

- PARTIZIONE IN LOCO

$$T_{\text{BEST}}(n) = \Theta(n \log n)$$

$$T_{\text{Worst}}(n) = \Theta(n^2)$$

$$T_{\text{medio}}(n) = \Theta(n \log n)$$

INTEGER SORT \rightarrow ORDINAMENTO LINEARE

- USA ARRAY DI APPoggIO
 - $\Theta(n)$
 - Lo \rightarrow per ogni indice memorizza quanti numeri ha trovato

BUCKET SORT

- È UN ALGORITMO STABILE
 - $\Theta(n)$
 - Lo preserva in output l'ordine posizionale recuperato che avevano gli elementi uguali in input

RADIX SORT

- CONFRONTO GLI ELEMENTI DALLE UNITA' IN PIAZZA
- $\Theta(n)$

D - HEAP

- ORDINAMENTO DI TIPO MIN-HEAP
- ALTEZZA: $h = \Theta(\log_d n)$
 - muoviAlto(v)
 - $T(n) = O(\log_d n)$
 - muoviBasso(v)
 - $T(n) = O(d \log_d n)$
 - findMin()
 - $T(n) = O(1)$
 - insert(elem e, chavi k)
 1. lo posiziona come foglia
 2. applica muoviAlto
 - $T(n) = O(\log_d n)$
 - delete(elem e)
 - POSSONO ESSERE CHIAMATI muoviBasso o muoviAlto
 - $T(n) = O(d \log_d n)$
 - deleteMin()
 - ESEGUE muoviBasso
 - $T(n) = O(d \log_d n)$

- ~~decreasekey~~ (elem e, chiave A)
 - DECREMENTO DELLA QUANTITÀ A
 - ESEGUE ~~muoi alto~~
 - $T(n) = O(\log d n)$

- ~~increasekey~~ (elem e, chiave A)
 - AUMENTA DELLA QUANTITÀ A
 - ESEGUE ~~muoi basso~~
 - $T(n) = O(d \log d n)$

- merge (key d-ario c₁, key d-ario c₂)
 - $T(n) = \Theta(n)$

HEAD BINOMIALE

- ORDINAMENTO A MIN-HEAP

- FORMATO DA ALBERI BINOMIALI

↳ $B_0, B_1, B_2 \dots B_i$

- *ristruttura()*

- SI FONDONO DUE B_i PER FORMARE UN ALBUM B_{i+1}

- *findMin()*

- $O(\log n)$

- *insert(elem e, chiave k)*

- UTILIZZA LA PROCEDURA *ristruttura*

- $O(\log n)$

- *deleteMin()*

- TROVA L'ALBERO B_h CON LA RADICE A CHIAVE MINIMA

↳ SI TOGLIE LA RADICE

↳ SI SPERZZA FORMANDO h
ALBERI BINOMIALI

- RIPRISANO DELL'UNICITÀ DIVIDENDO I DOPPIOM

- $O(\log n)$

- *decreaseKey(elem e, chiave Δ > 0)*

- DECREMENTA DI Δ

- RIPRISTINO DELLE PROPRIETÀ SPERZANDO IL
NODO VERSO L'ALTO (min-heap)

- $O(\log n)$

- delete (elem e)

- RICHIAMA increasekey (e, ∞)
- RICHIAMA deleteMin()
- $O(\log n)$



COSÌ DA METTERLO
NUOVA CHIAVE PER POI
ELIMINARLO

- increasekey (elem e, chiave $\Delta > 0$)

- RICHIAMA delete(e)
- RICHIAMA insert (e, $k + \Delta$)
- $O(\log n)$

↳ CHIAVE DI e

- merge (H_1, H_2)

- UNISCE GLI ALBERI H_1 E H_2
- FUSIONE DEI DOPPIUMI
- $O(\log n)$