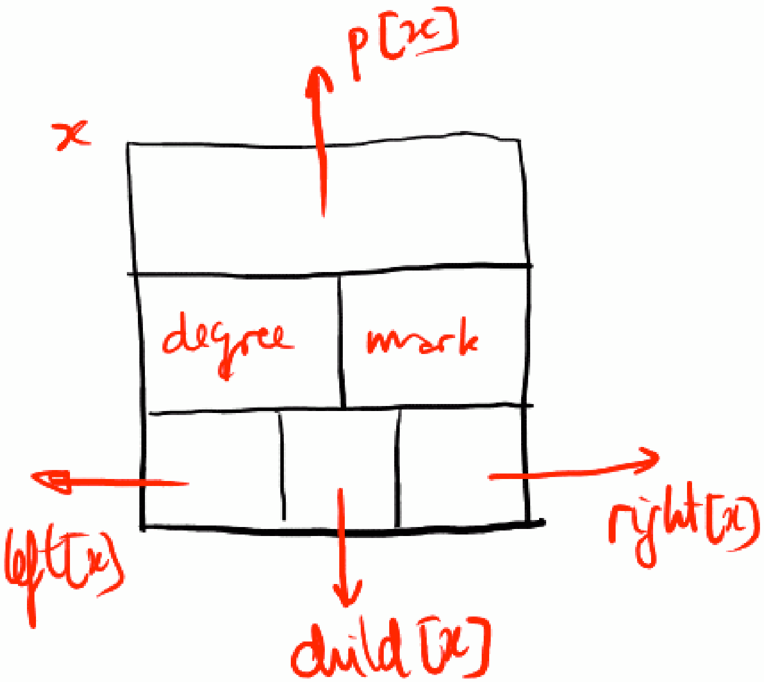
Heap di Fibonacci

- UNO HEAP DI FIBONACCI E' UNA COLLEZIONE DI ALBERI CON LA PROPRIETA' HEAP
- GLI ALBERI DI UNO HEAP DI FIBONACCI NON DEBBONO ESSERE NECESSARIAMENTE ALBERI BINOMIALI

Procedure	Binary heap (worst-case)	Binomial heap (worst-case)	Fibonacci heap (amortized)
MAKE-HEAP	$\Theta(1)$	$\Theta(1)$	$\Theta(1)$
INSERT	$\Theta(\lg n)$	$O(\lg n)$	$\Theta(1)$
MINIMUM	$\Theta(1)$	$O(\lg n)$	$\Theta(1)$
EXTRACT-MIN	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$ (*)
UNION	$\Theta(n)$	$O(\lg n)$	$\Theta(1)$
DECREASE-KEY	$\Theta(\lg n)$	$\Theta(\lg n)$	$\Theta(1)$ (*)
DELETE	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$ (*)

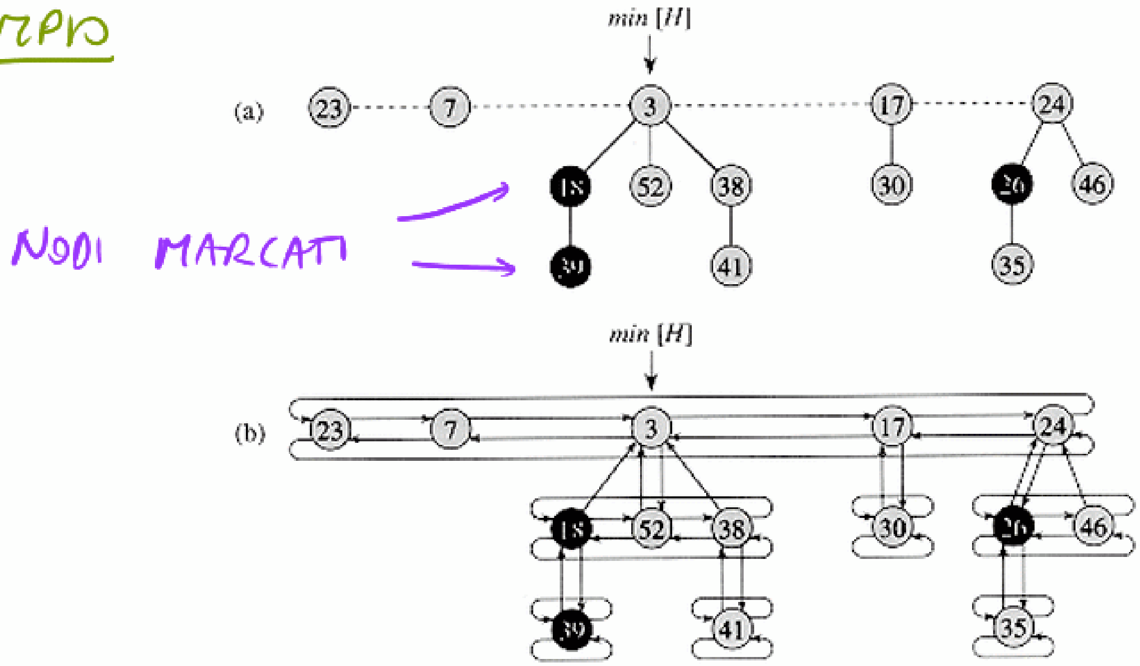
(*) Costi ammortizzati



RAPPRESENTAZIONE DI UN NODO

$p[x]$ – padre
$left[x]$ – fratello sinistro
$right[x]$ – fratello destro
$child[x]$ – (un) figlio
$degree[x]$ – numero di figli
$mark[x]$ – indica se il nodo x ha perduto un figlio dall'ultima volta in cui x è diventato figlio di un altro nodo

ESEMPIO



- IL PUNTATORE min[H] INDICA LA RADICE CONTENENTE LA CHIAVE MINIMA E DÀ ACCESSO ALLA STRUTTURA
- VIENE ANCHE MANTENUTO IL CAMPO n[H] CHE CONTIENE IL NUMERO DI NODI IN H

FUNZIONE POTENZIALE

$$\Phi(H) = t(H) + 2\,m(H)$$

DOVE

- $t(H)$ = # ALBERI NELLA LISTA DELLE RADICI DI H

- $m(H)$ = # NODI MARCATI IN H

- SIA $\{H_i\}_{i \in I}$ UNA COLLEZIONE FINITA DI HEAP. PONIAMO:

$$\Phi(\{H_i\}_{i \in I}) = \sum_{i \in I} \Phi(H_i)$$

MASSIMO GRADO DI UN NODO: $D(n)$

- L'ANALISI VERRÀ EFFETTUATA IN FUNZIONE DI UN UPPER BOUND $D(n)$ SUL MASSIMO GRADO DI UN NODO QUALUNQUE IN UNO HEAP CON n NODI
- DIMOSTREREMO CHE SI HA $D(n) = O(\lg n)$

- SE VENGONO ESEGUITE SOLO OPERAZIONI DEL TIPO:
    - MAKE-HEAP
    - INSERT
    - MINIMUM
    - EXTRACT-MIN
    - UNION

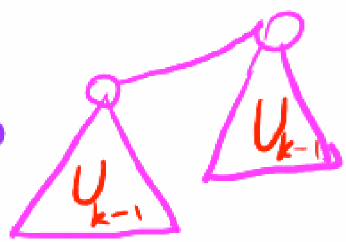
CIASCUNO HEAP DI FIBONACCI E' RAPPRESENTABILE COME COLLEZIONE DI ALBERI BINOMIALI NON ORDINATI

ALBERI BINOMIALI NON ORDINATI

DEFINIZIONE

PER OGNI $k \in \mathbb{N}$ ESISTE UN ALBERO BINOMIALE NON ORDINATO U_k DI GRADO k, DEFINITO IN BASE ALLA SEGUENTE RICORSIONE:

- U_0 E' L'ALBERO FORMATO DA UN SOLO NODO
- DATO U_{k-1} DEFINIAMO U_k COMBINANDO DUE COPIE DI U_{k-1} NELLA SEGUENTE MANIERA:

$U_k \Rightarrow$ 

(NON ORDINATO)

LEMMA (PROPRIETÀ DEGLI ALBERI BINOMIALI NON ORDINATI)

PER OGNI $k=0,1,2,\dots$ VALGONO LE SEGUENTI PROPRIETÀ:

1. U_k HA 2^k NODI
2. L'ALTEZZA DI U_k È k
3. U_k HA $\binom{k}{i}$ NODI A PROFONDITÀ i ($i=0,1,\dots,k$)
4. LA RADICE DI U_k HA GRADO k ED OGNI ALTRO NODO IN U_k HA GRADO $< k$.
   INOLTRE I FIGLI DELLA RADICE DI U_k SONO RADICI DI $U_0, U_1, U_2, \dots U_{k-1}$ (IN QUALCHE ORDINE).

■

- SI OSSERVI CHE DAL LEMMA PRECEDENTE SEGUE IMMEDIATAMENTE CHE $D(n) = O(\lg n)$ ALMENO QUANDO LO HEAP E' FORMATO SOLTANTO DA ALBERI BINOMIALI NON ORDINATI

- LA STRATEGIA DI MANTENIMENTO DEGLI HEAP DI FIBONACCI PREVEDE DI RITARDARE IL LAVORO IL PIÙ POSSIBILE

MAKE-FIBONACCI-HEAP( )

```
H := allocate_node( );
n[H] := 0;
min[H] := NIL;
return [H]
```

COMPLESSITA':

$$\left.\begin{aligned}\Delta t &= 0\\ \Delta m &= 0\end{aligned}\right\} \Longrightarrow \Delta\phi = 0$$

PERTANTO: $\hat{c} = c + \Delta\phi = c = O(1)$

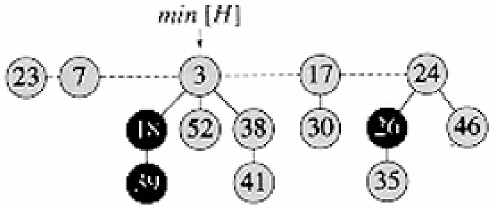
```
FIB-HEAP-INSERT(H, x)
degree[x] ← 0
p[x] ← NIL
child[x] ← NIL
left[x] ← x
right[x] ← x
mark[x] ← FALSE
concatenate the root list containing x with root list H
if min[H] = NIL or key[x] < key[min[H]]
   then min[H] ← x
n[H] ← n[H] + 1
```

COMPLESSITA'

$\Delta t = 1, \; \Delta m = 0 \implies \Delta \phi = 1$

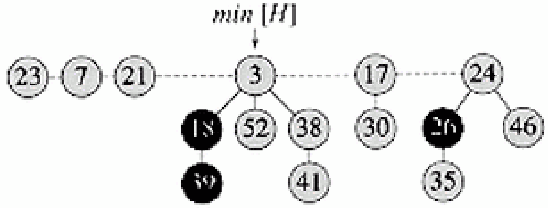
$\hat{c} = c + \Delta \phi = c + 1 = O(1)$

ESEMPIO



FIB-HEAP-INSERT(H, x)

CON key[x] = 21



MINIMUM(H)

return (min[H])

COMPLESSITA'

$\Delta\phi = 0$

$\hat{c} = c + \Delta\phi = c = O(1)$

```
FIB-HEAP-UNION(H1,H2)
H ← MAKE-FIB-HEAP()
min[H] ← min[H1]
concatenate the root list of H2 with the root list of H
if (min[H1] = NIL) or (min[H2] ≠ NIL and key[min[H2]] < key[min[H1]])
then min[H] ← min[H2]
n[H] ← n[H1] + n[H2]
free the objects H1 and H2
return H
```

COMPLESSITA':

$$\left.\begin{array}{l}\Delta t = 0 \\ \Delta m = 0\end{array}\right\} \Longrightarrow \Delta\phi = 0$$

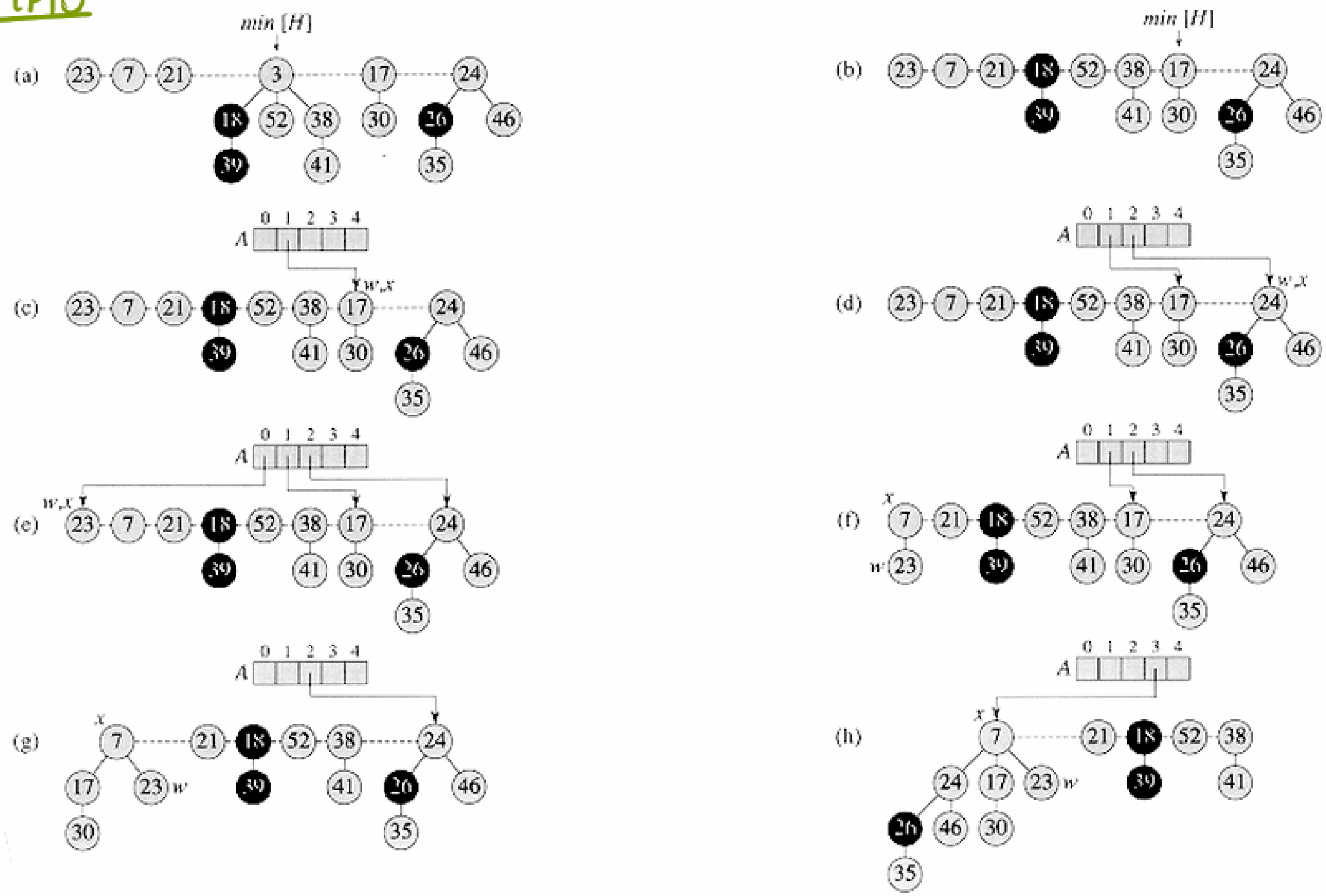
PERTANTO: $\hat{c} = c + \Delta\phi = c = O(1)$

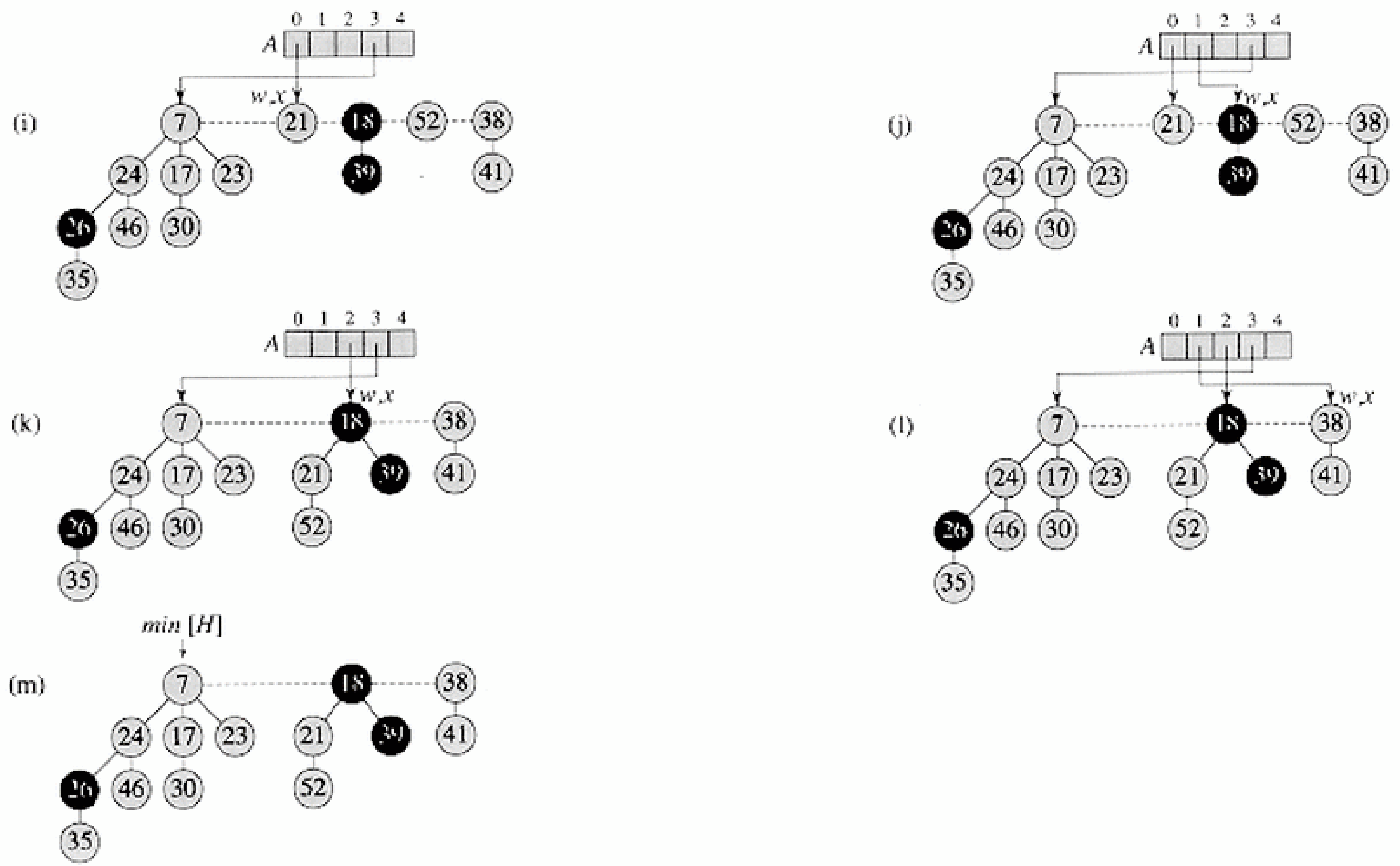
```
FIB-HEAP-EXTRACT-MIN(H)
z ← min[H]
if z ≠ NIL
    then for each child x of z
             do add x to the root list of H
                p[x] ← NIL
         remove z from the root list of H
         if z = right[z]
            then min[H] ← NIL
            else min[H] ← right[z]
                 CONSOLIDATE(H)
         n[H] ← n[H] - 1
return z
```

```
CONSOLIDATE(H)
for i ← 0 to D(n[H])
    do A[i] ← NIL
for each node w in the root list of H
    do x ← w
       d ← degree[x]
       while A[d] ≠ NIL
           do y ← A[d]
              if key[x] > key[y]
                 then exchange x ↔ y
              FIB-HEAP-LINK(H, y, x)
              A[d] ← NIL
              d ← d + 1
       A[d] ← x
min[H] ← NIL
for i ← 0 to D(n[H])
    do if A[i] ≠ NIL
          then add A[i] to the root list of H
               if min[H] = NIL or
                  key[A[i]] < key[min[H]]
                  then min[H] ← A[i]
```

```
FIB-HEAP-LINK(H, y, x)
remove y from the root list of H
make y a child of x, incrementing degree[x]
mark[y] ← FALSE
```

ESEMPIO




(i)
(j)
(k)
(l)
(m)
A
0 1 2 3 4
w,x
min [H]
7
21
18
52
38
24
17
23
39
41
26
46
30
35

COMPLESSITA' DI FIB-HEAP-EXTRACT-MIN

$$\text{COSTO REALE} = O\big(\underbrace{D(n)}_{\substack{\text{PROCESSAMENTO} \\ \text{FIGLI DI } min[H]}} + \underbrace{D(n) + t(H)}_{\substack{\text{PROCESSAMENTO} \\ \text{LISTA DI RADICI} \\ \text{IN CONSOLIDATE}}}\big)$$

$$= O(D(n) + t(H))$$

$$\Delta t \leq (D(n)+1) - t(H), \quad \Delta m \leq 0$$

$$\Delta \phi = \Delta t + 2\Delta m \leq D(n) + 1 - t(H)$$

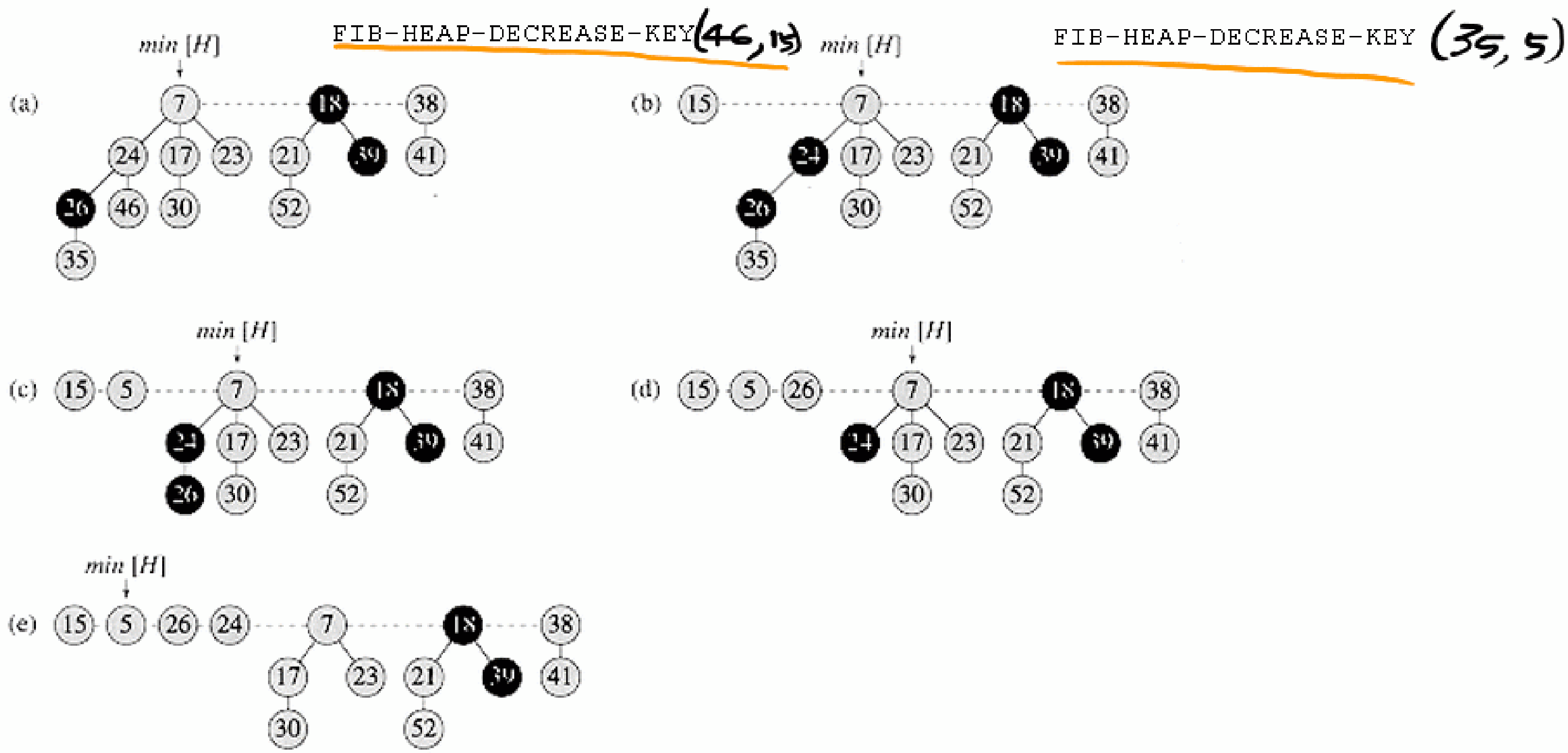
$$\hat{c} = c + \Delta\phi = O(D(n) + t(H)) - t(H) = O(D(n))$$

(PUR DI SCALARE OPPORTUNAMENTE IL POTENZIALE)

```
FIB-HEAP-DECREASE-KEY(H, x, k)
if k > key[x]
   then error "new key is greater than current key"
key[x] ← k
y ← p[x]
if y ≠ NIL and key[x] < key[y]
   then CUT(H, x, y)
        CASCADING-CUT(H, y)
if key[x] < key[min[H]]
   then min[H] ← x
```

```
CUT(H, x, y)
remove x from the child list of y, decrementing degree[y]
add x to the root list of H
p[x] ← NIL
mark[x] ← FALSE
```

```
CASCADING-CUT(H, y)
z ← p[y]
if z ≠ NIL
   then if mark[y] = FALSE
           then mark[y] ← TRUE
           else CUT(H, y, z)
                CASCADING-CUT(H, z)
```


FIB-HEAP-DECREASE-KEY(46, 15)
FIB-HEAP-DECREASE-KEY (35, 5)
min [H]
(a)
(b)
(c)
(d)
(e)
7
18
38
24
17
23
21
39
41
26
46
30
52
35
15
5

COMPLESSITA' DI FIB-HEAP-DECREASE-KEY

SUPPONIAMO CHE LA PROCEDURA CASCADING-CUT VENGA CHIAMATA d VOLTE

COSTO REALE $= O(d)$

$$\phi(H) = t(H) + 2m(H)$$

$$\phi(H') \leq (t(H) + d) + 2(m(H) - (d-1) + 1)$$

$$= t(H) + d + 2m(H) - 2d + 2 + 2$$

$$= t(H) + 2m(H) - d + 4$$

$$\hat{c} \leq O(d) + \phi(H') - \phi(H) = O(d) - d + 4 = O(1)$$

(PUR DI SCALARE OPPORTUNAMENTE IL POTENZIALE)

```
FIB-HEAP-DELETE(H, x)
FIB-HEAP-DECREASE-KEY(H, x, -∞)
FIB-HEAP-EXTRACT-MIN(H)
```

COMPLESSITA' AMMORTIZZATA = $O(D(n))$

ALCUNE PROPRIETÀ SUI NUMERI DI FIBONACCI

$$F_0 = 0 \,, \quad F_1 = 1$$

$$F_k = F_{k-1} + F_{k-2} \,, \quad \text{PER} \quad k \geq 2$$

LEMMA 1 $\quad F_{k+2} = 1 + \sum_{i=0}^{k} F_i \,, \quad$ PER $\quad k \geq 0$

DIM

CASO BASE $\quad k=0$

$$F_{k+2} = F_2 = 0+1 = 1 \,, \qquad 1 + \sum_{i=0}^{k} F_i = 1 + \sum_{i=0}^{0} F_i = 1 + 0 = 1$$

PASSO INDUTTIVO

$$F_{(k+1)+2} = F_{k+3} = F_{k+2} + F_{k+1} = 1 + \sum_{i=0}^{k} F_i + F_{k+1} = 1 + \sum_{i=0}^{k+1} F_i \,.$$

$\blacksquare$

LEMMA 2 $\quad F_{k+2} \geq \phi^k$, PER $k \geq 0$

(DOVE $\phi = \frac{1+\sqrt{5}}{2}$ E' IL RAPPORTO AUREO)

DIM.

CASO BASE

$k=0 \longrightarrow \quad F_2 = 1, \quad \phi^0 = 1$ ✓

$k=1 \longrightarrow \quad F_3 = 2, \quad \phi^1 = \phi < 2$

PASSO INDUTTIVO $\quad (k \geq 1)$

$$F_{k+3} = F_{k+1} + F_{k+2} \geq \phi^{k-1} + \phi^k = \phi^{k-1}(1+\phi) = \phi^{k-1} \cdot \phi^2 = \phi^{k+1}$$

IN QUANTO $\quad 1+\phi = \phi^2$ $\blacksquare$

STIMA DI $D(n)$

LEMMA 3 SIA x UN NODO IN UNO HEAP DI FIBONACCI E SIA $degree[x] = k$, SIANO $y_1, y_2, \ldots, y_k$ I FIGLI DI x NELL'ORDINE IN CUI SONO STATI INNESTATI IN x.
ALLORA $degree[y_i] \geq max(i-2, 0)$, PER $i = 1, \ldots, k$.

DIM. - PER $i=1$ SI HA: $degree[y_1] \geq 0 = max(1-2, 0)$

- PER $i \geq 2$, NOTIAMO CHE QUANDO y_i E' INNESTATO IN x (AL TEMPO T), IL NODO x HA GIA' $y_1, \ldots, y_{i-1}$ TRA I SUOI FIGLI, PER CUI $degree_T[y_i] = degree_T[x] \geq i-1$. DALL'ISTANTE T, y_i PUÒ AVERE PERDUTO AL PIÙ UN FIGLIO E QUINDI $degree[y_i] \geq i-2 = max(i-2, 0)$. ∎

LEMMA SIA x UN NODO IN UNO HEAP DI FIBONACCI E SIA $degree[x] = k$.

ALLORA $size(x) \geq F_{k+2}$, DOVE $size(x)$ E' IL NUMERO DI NODI NEL SOTTOALBERO RADICATO IN x.

DIM

PONIAMO
$$s_j = \min_{\substack{degree[z]=j \\ z \in H \\ H \in \mathcal{H}}} size[z]$$

CON $\mathcal{H}$ FAMIGLIA DI TUTTI GLI HEAP DI FIBONACCI

SI HA: $s_0 = 1$, $s_1 \geq 2$ E $s_{j+1} > s_j$ $(j \in \mathbb{N})$

DIMOSTRIAMO CHE $s_j \geq F_{j+2}$, PER $j = 0, 1, 2, \ldots$

CASO $j = 0$: $s_j = 1$, $F_{j+2} = 1$ ✓

CASO $j = 1$: $s_j \geq 2$, $F_{j+2} = F_3 = 2$ ✓

CASO $j \geq 2$: SIA z UN NODO IN UNO HEAP DI FIBONACCI TALE $degree[z] = j$, $size[z] = s_j$ E SIANO $y_1, y_2, \ldots, y_j$ I FIGLI DI z NELL'ORDINE IN CUI SONO STATI INNESTATI IN z.

$$s_j = size[z] = size[y_1] + size[y_2] + \ldots + size[y_j] + 1$$

$$\geq s_0 + s_{2-2} + \ldots + s_{j-2} + 1$$

$$= 2 + \sum_{i=2}^{j} s_{i-2} \geq 2 + \sum_{i=2}^{j} F_i = 1 + \sum_{i=0}^{j} F_i = F_{j+2}$$

POICHÉ $degree[x] = k$, SI HA

$$size[x] \geq s_k \geq F_{k+2} \qquad \blacksquare$$

COROLLARIO $size[x] \geq \phi^{degree[x]}$

COROLLARIO $D(n) \leq \lfloor \log_\phi n \rfloor$, DA CUI $D(n) = O(\log n)$.

DIM SIA x UN NODO IN UNO HEAP DI FIBONACCI CON n NODI.

SI HA: $n \geq size[x] \geq \phi^{degree[x]}$

PERTANTO $degree[x] \leq \lfloor \log_\phi n \rfloor$ E QUINDI

$$D(n) = \max_{\substack{x \in H \\ H \in \mathcal{H}_n}} degree[x] \leq \lfloor \log_\phi n \rfloor \qquad \blacksquare$$