Mergeable Heaps

Supportano le operazioni

MAKE-HEAP()

INSERT(H, x)

MINIMUM(H)

EXTRACT-MIN(H)

UNION(H_1, H_2)

DECREASE-KEY(H, x, k)

DELETE(H, x)

Considereremo due implementazioni

- Heap binomiali
- Heap di Fibonacci

MAKE-HEAP()	creates and returns a new heap containing no elements.
INSERT(H, x)	inserts node x, whose *key* field has already been filled in, into heap H.
MINIMUM(H)	returns a pointer to the node in heap H whose key is minimum.
EXTRACT-MIN(H)	deletes the node from heap H whose key is minimum, returning a pointer to the node.
UNION(H_1, H_2)	creates and returns a new heap that contains all the nodes of heaps H_1 and H_2. Heaps H_1 and H_2 are "destroyed" by this operation.
DECREASE-KEY(H, x, k)	assigns to node x within heap H the new key value k, which is assumed to be no greater than its current key value.
DELETE(H, x)	deletes node x from heap H.

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	$\Theta(\lg n)$	$\Theta(\lg n)$	$O(\lg n)$ (*)

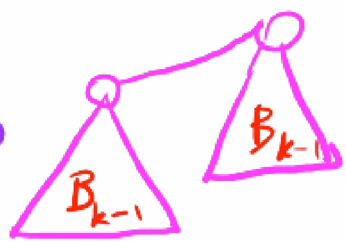
(*) Costi ammortizzati

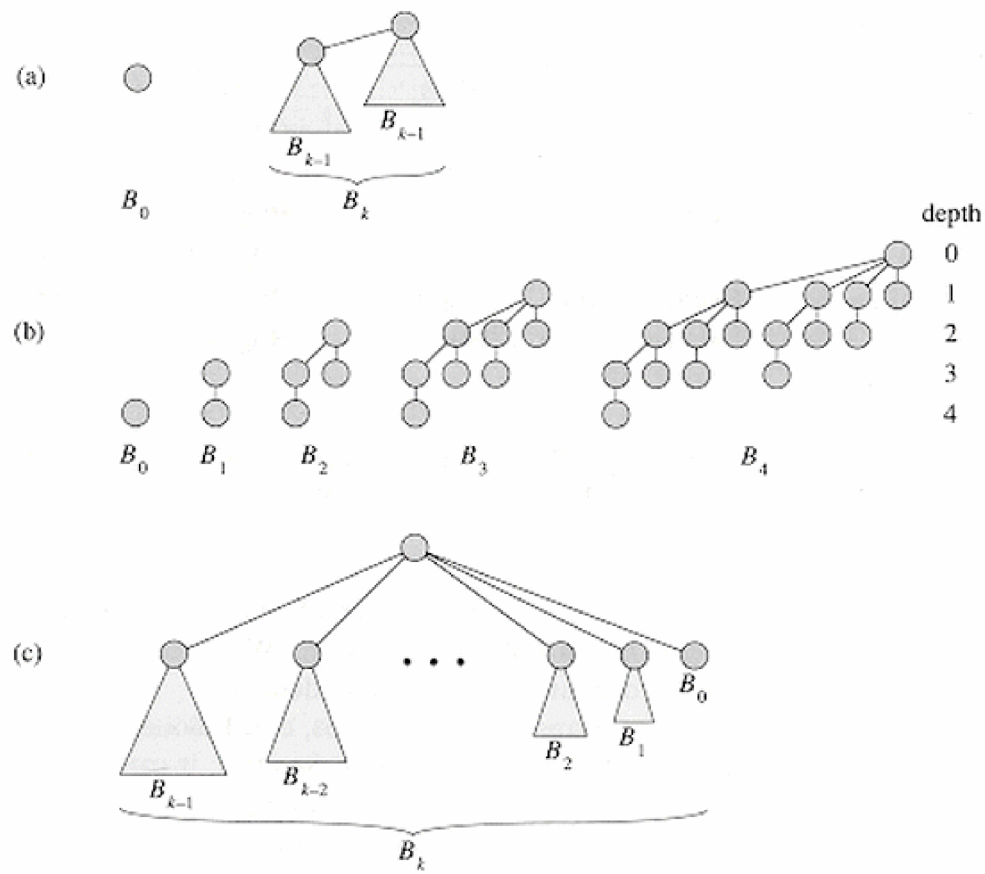
ALBERI BINOMIALI

DEFINIZIONE

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

- B_0 E' L'ALBERO FORMATO DA UN SOLO NODO
- DATO B_{k-1} DEFINIAMO B_k COMBINANDO DUE COPIE DI B_{k-1} NELLA SEGUENTE MANIERA:

$B_k \Rightarrow$ 


(a)
B_0
B_{k-1}
B_{k-1}
B_k
(b)
depth
0
1
2
3
4
B_0
B_1
B_2
B_3
B_4
(c)
B_{k-1}
B_{k-2}
B_2
B_1
B_0
B_k

LEMMA (PROPRIETÀ DEGLI ALBERI BINOMIALI)

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

1. B_k HA 2^k NODI
2. L'ALTEZZA DI B_k È k
3. B_k HA $\binom{k}{i}$ NODI A PROFONDITÀ i ($i = 0, 1, \ldots, k$)
4. LA RADICE DI B_k HA GRADO k ED OGNI ALTRO NODO IN B_k HA GRADO $< k$.
   INOLTRE I FIGLI DELLA RADICE DI B_k SONO RADICI DI $B_{k-1}, B_{k-2}, \ldots, B_2, B_1, B_0$, NELL'ORDINE DATO.

DIM.

PER INDUZIONE

CASO BASE $k=0$

1. B_0 HA $1 = 2^0$ NODI
2. L'ALTEZZA DI B_0 E' 0
3. B_0 HA $1 = \binom{0}{0}$ NODI A PROFONDITA' 0
4. LA RADICE DI B_0 HA GRADO 0

PASSO INDUTTIVO

SUPPONIAMO CHE IL LEMMA SIA VERO PER $k-1$, CON $k \geq 1$

1. B_k HA $\#(B_{k-1}) + \#(B_{k-1}) = 2 \cdot 2^{k-1} = 2^k$ NODI
2. L'ALTEZZA DI B_k E' UGUALE A

   $\text{height}(B_{k-1}) + 1 = (k-1) + 1 = k$

3. SIA $1 \leq i \leq k-1$.

   IL NUMERO DI NODI DI B_k A PROFONDITA' i È UGUALE A $\binom{k-1}{i} + \binom{k-1}{i-1} = \binom{k}{i}$.

   INOLTRE B_k HA
   - $1 = \binom{k}{0}$ NODI A PROFONDITA' 0
   - $\binom{k-1}{k-1} = \binom{k}{k} = 1$ NODI A PROFONDITA' k

4. • IL GRADO DELLA RADICE DI B_k E' UGUALE A
$degree(B_{k-1}) + 1 = (k-1) + 1 = k$

• INOLTRE CIASCUN ALTRO NODO APPARTIENE AD UN B_{k-1} E PERTANTO PER IPOTESI INDUTTIVA HA GRADO $\leq k-1$, CIOE' $< k$

• IL PRIMO FIGLIO DELLA RADICE DI B_k E' RADICE DI B_{k-1}. INOLTRE, PER INDUTTIVA, I SUCCESSIVI $k-1$ FIGLI DELLA RADICE DI B_k SONO RADICI DI $B_{k-2}, \dots, B_1, B_0$.

■

COROLLARIO

SIA B UN ALBERO BINOMIALE CON n NODI.
ALLORA OGNI NODO IN B HA GRADO AL PIÙ $\lg n$.

DIM.

PER QUALCHE $k \in \mathbb{N}$ SI HA $B = B_k$.
QUINDI $n = 2^k$. OGNI NODO IN B_k HA GRADO
$\leq k$, MA $k = \lg n$, DA CUI LA TESI. ■

HEAP BINOMIALI

DEFINIZIONE

UNO HEAP BINOMIALE H E' UN INSIEME DI ALBERI BINOMIALI TALE CHE

- CIASCUN ALBERO BINOMIALE IN H GODE DELLA PROPRIETÀ min-heap
- PER OGNI $k \in \mathbb{N}$, H CONTIENE AL PIÙ UN SOLO ALBERO BINOMIALE DI GRADO k

CONSEGUENZE IMMEDIATE:

- IN CIASCUN ALBERO BINOMIALE IN UNO HEAP BINOMIALE, LA RADICE CONTIENE LA CHIAVE MINIMA DELL'ALBERO
- UNO HEAP BINOMIALE H CON n NODI E' FORMATO DA AL PIÙ $\lfloor \lg n \rfloor + 1$ ALBERI BINOMIALI

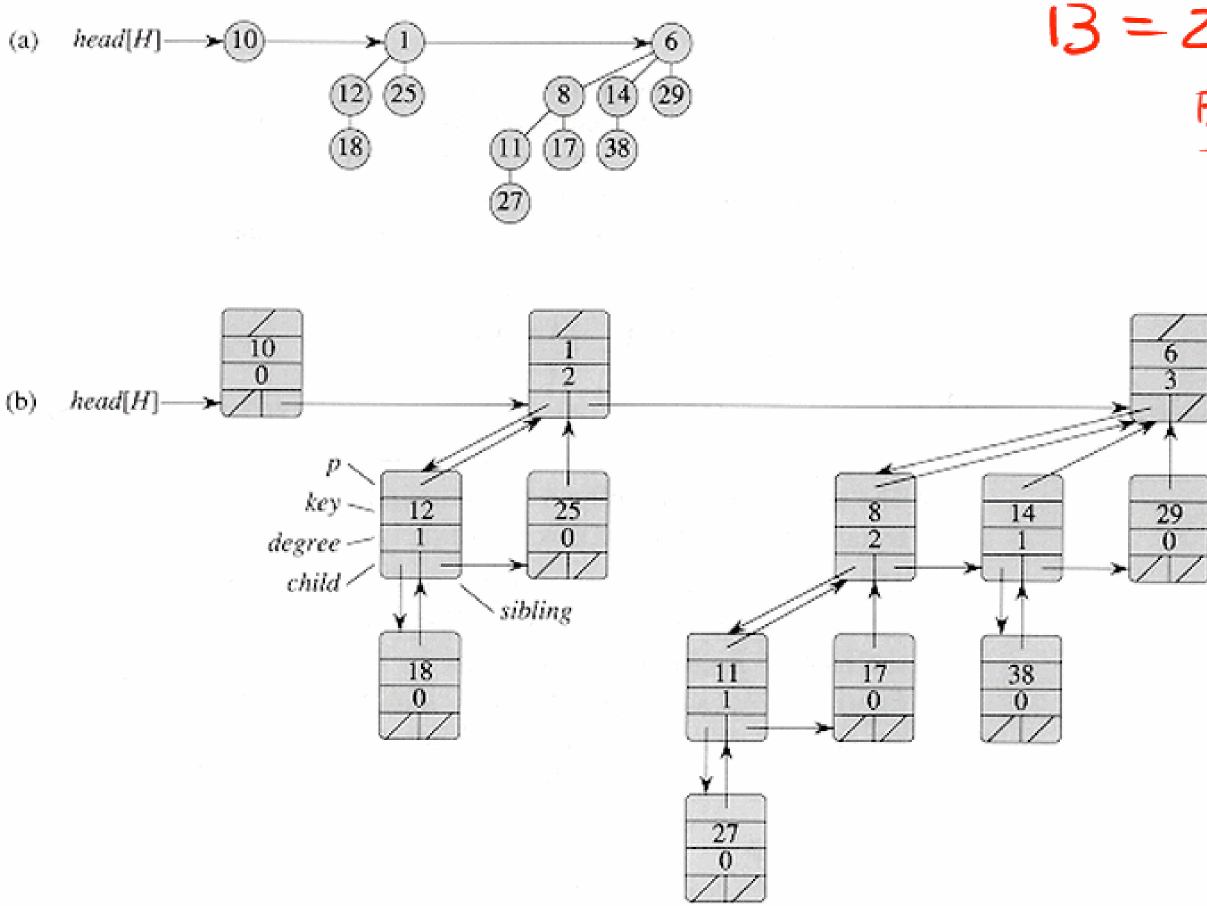
INFATTI, SIA B_k L'ALBERO BINOMIALE DI GRADO MASSIMO IN H. SI HA $2^k \leq n$, DA CUI $k \leq \lg n$ E QUINDI $k \leq \lfloor \lg n \rfloor$.

PERTANTO IL NUMERO DI ALBERI BINOMIALI IN H E' AL PIÙ $k+1 \leq \lfloor \lg n \rfloor + 1$

ESEMPIO DI HEAP BINOMIALE

$$13 = 2^3 + 2^2 + 2^0$$

$B_3 \quad B_2 \quad B_0$



```
MAKE_BINOMIAL_HEAP()
    H := allocate_heap();
    head[H] := NIL;
    return H;
```

COMPLESSITA'

$O(1)$

```
BINOMIAL-HEAP-MINIMUM(H)
y ← NIL
x ← head[H]
min ← ∞
while x ≠ NIL
      do if key[x] < min
            then min ← key[x]
                 y ← x
         x ← sibling[x]
return y
```

COMPLESSITA'

O(LUNGHEZZA DELLA LISTA DELLE RADICI)

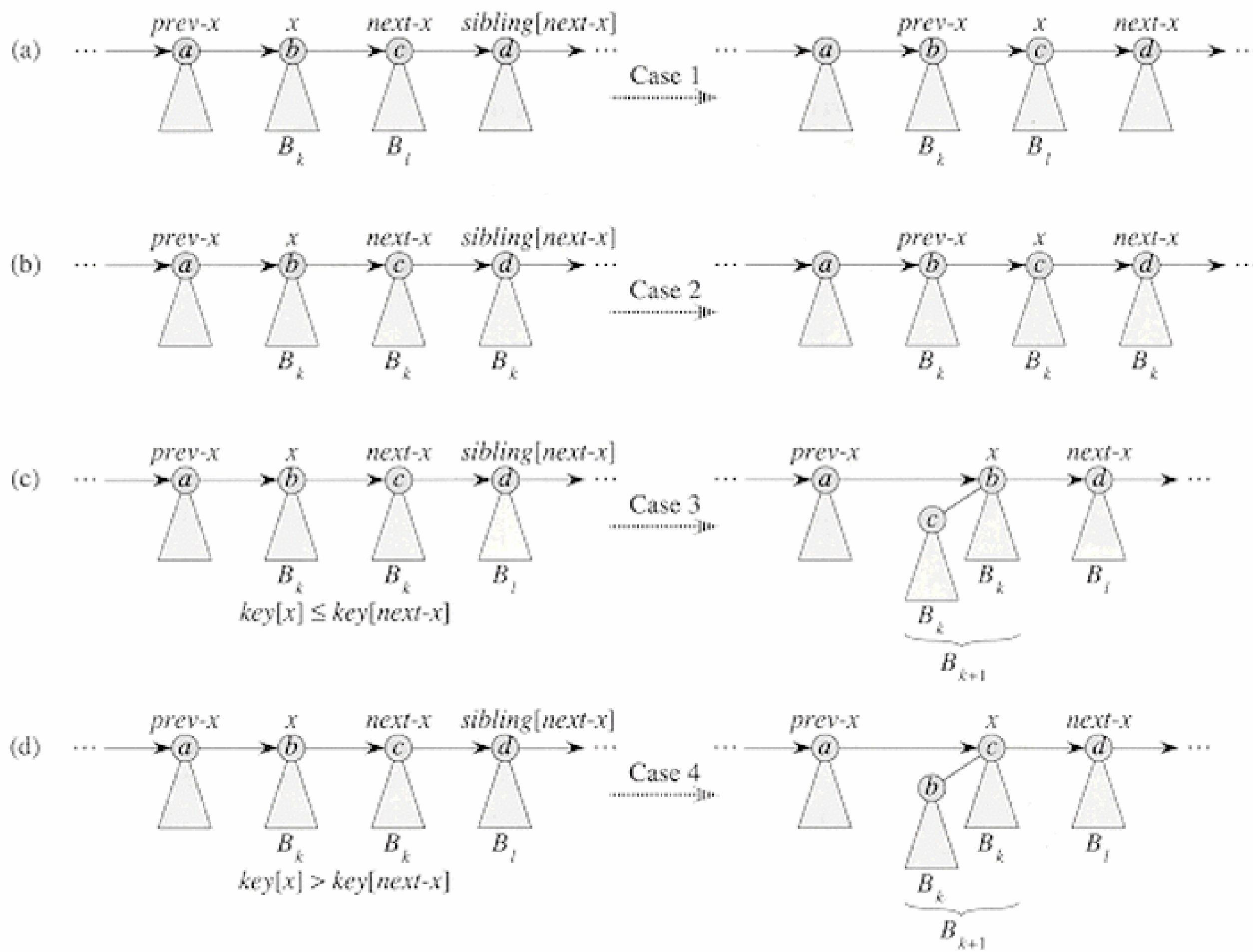
$= O(\lg n)$

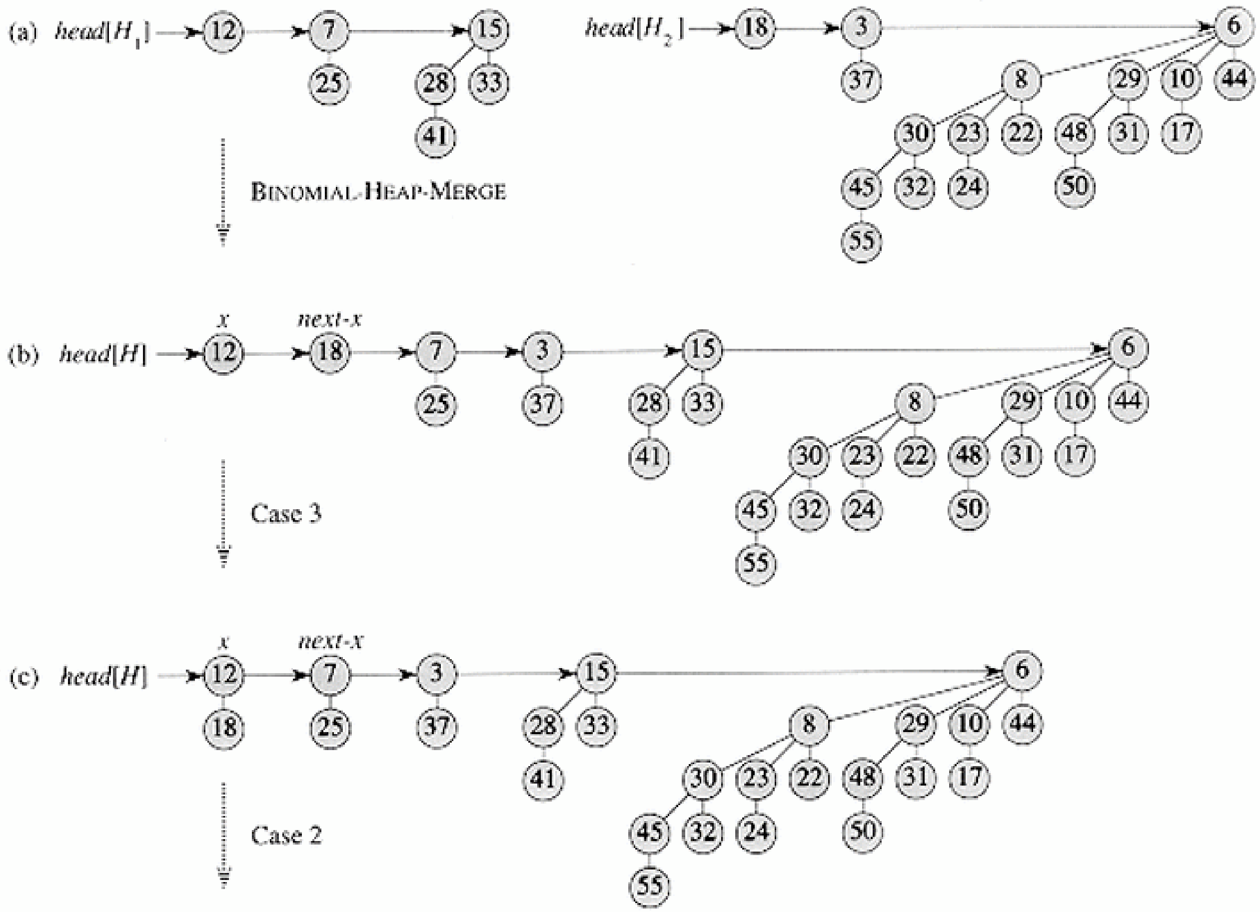
```
BINOMIAL-LINK(y,z)

p[y] ← z
sibling[y] ← child[z]
child[z] ← y
degree[z] ← degree[z]+1
```

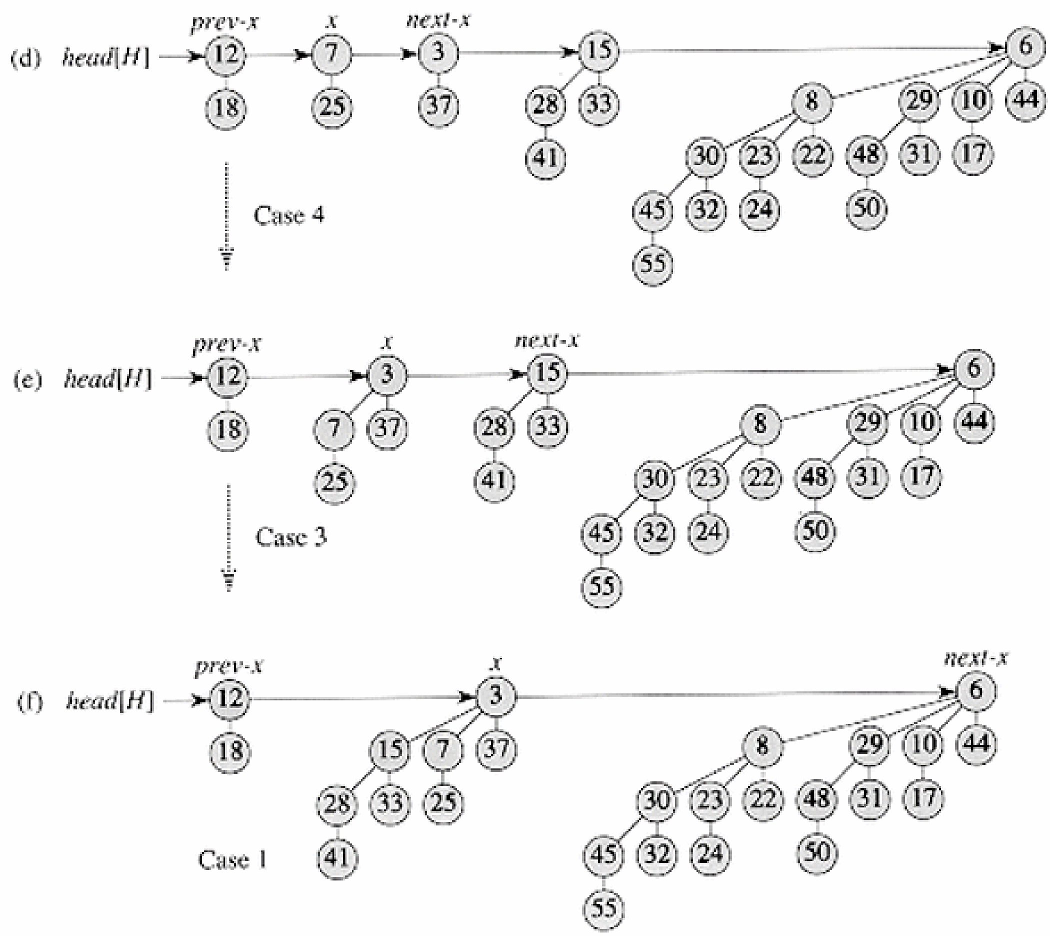
COMPLESSITA'

$O(1)$

```
BINOMIAL-HEAP-UNION(H1, H2)
H ← MAKE-BINOMIAL-HEAP()
head[H] ← BINOMIAL-HEAP-MERGE(H1, H2)
free the objects H1 and H2 but not the lists they point to
if head[H] = NIL
   then return H
prev-x ← NIL
x ← head[H]
next-x ← sibling[x]
while next-x ≠ NIL
     do if (degree[x] ≠ degree[next-x]) or
               (sibling[next-x] ≠ NIL
               and degree[sibling[next-x]] = degree[x])
          then prev-x ← x                              ▷ Cases 1 and 2
               x ← next-x                              ▷ Cases 1 and 2
          else if key[x] ≤ key[next-x]
                  then sibling[x] ← sibling[next-x]        ▷ Case 3
                       BINOMIAL-LINK(next-x, x)             ▷ Case 3
                  else if prev-x = NIL                      ▷ Case 4
                          then head[H] ← next-x             ▷ Case 4
                          else sibling[prev-x] ← next-x     ▷ Case 4
                       BINOMIAL-LINK(x, next-x)             ▷ Case 4
                       x ← next-x                           ▷ Case 4
        next-x ← sibling[x]
return H
```


(a)
prev-x
x
next-x
sibling[next-x]
Case 1
(b)
Case 2
(c)
key[x] ≤ key[next-x]
Case 3
(d)
key[x] > key[next-x]
Case 4


(a) head[H1]
head[H2]
BINOMIAL-HEAP-MERGE
(b) head[H]
x
next-x
Case 3
(c) head[H]
x
next-x
Case 2


(d) head[H]
prev-x
x
next-x
Case 4
(e) head[H]
prev-x
x
next-x
Case 3
(f) head[H]
prev-x
x
next-x
Case 1

COMPLESSITA' DI BINOMIAL-HEAP-UNION

$n_1 = \#\text{nodi}(H_1) \rightarrow \#\text{alberi}(H_1) \leq \lfloor \lg n_1 \rfloor + 1$

$n_2 = \#\text{nodi}(H_2) \rightarrow \#\text{alberi}(H_2) \leq \lfloor \lg n_2 \rfloor + 1$

$n = n_1 + n_2$

$$\#\text{alberi}(H_1 \cup H_2) \leq \lfloor \lg n_1 \rfloor + \lfloor \lg n_2 \rfloor + 2 \leq 2\lfloor \lg n \rfloor + 2$$

COMPLESSITA' DI BINOMIAL-HEAP-MERGE: $O(\lg n)$

SI OSSERVI CHE AD OGNI ITERAZIONE DEL CICLO WHILE 9-21

- IL PUNTATORE x AVANZA DI UNA POSIZIONE,

OPPURE

- VIENE RIMOSSA UNA RADICE DALLA LISTA DELLE RADICI

INOLTRE, CIASCUNA ITERAZIONE DEL CICLO WHILE PRENDE TEMPO COSTANTE.

PERTANTO LA COMPLESSITA' DI BINOMIAL-HEAP-UNION E' $O(\lg n)$

```
BINOMIAL-HEAP-INSERT(H, x)

H' ← MAKE-BINOMIAL-HEAP()
p[x] ← NIL
child[x] ← NIL
sibling[x] ← NIL
degree[x] ← 0
head[H'] ← x
H ← BINOMIAL-HEAP-UNION(H,H')
```

SIA $n = \#nodi(H)$.
ALLORA LA COMPLESSITA' DI BINOMIAL-HEAP-INSERT E' $O(\log n)$

```
BINOMIAL-HEAP-EXTRACT-MIN(H)
1  find the root x with the minimum key in the root list of H,
and remove x from the root list of H
2  H' ← MAKE-BINOMIAL-HEAP()
3  reverse the order of the linked list of x's children,
and set head[H'] to point to the head of the resulting list
4  H ← BINOMIAL-HEAP-UNION(H, H')
5  return x
```

<u>COMPLESSITA'</u> ($n = \#$ nodi (H))

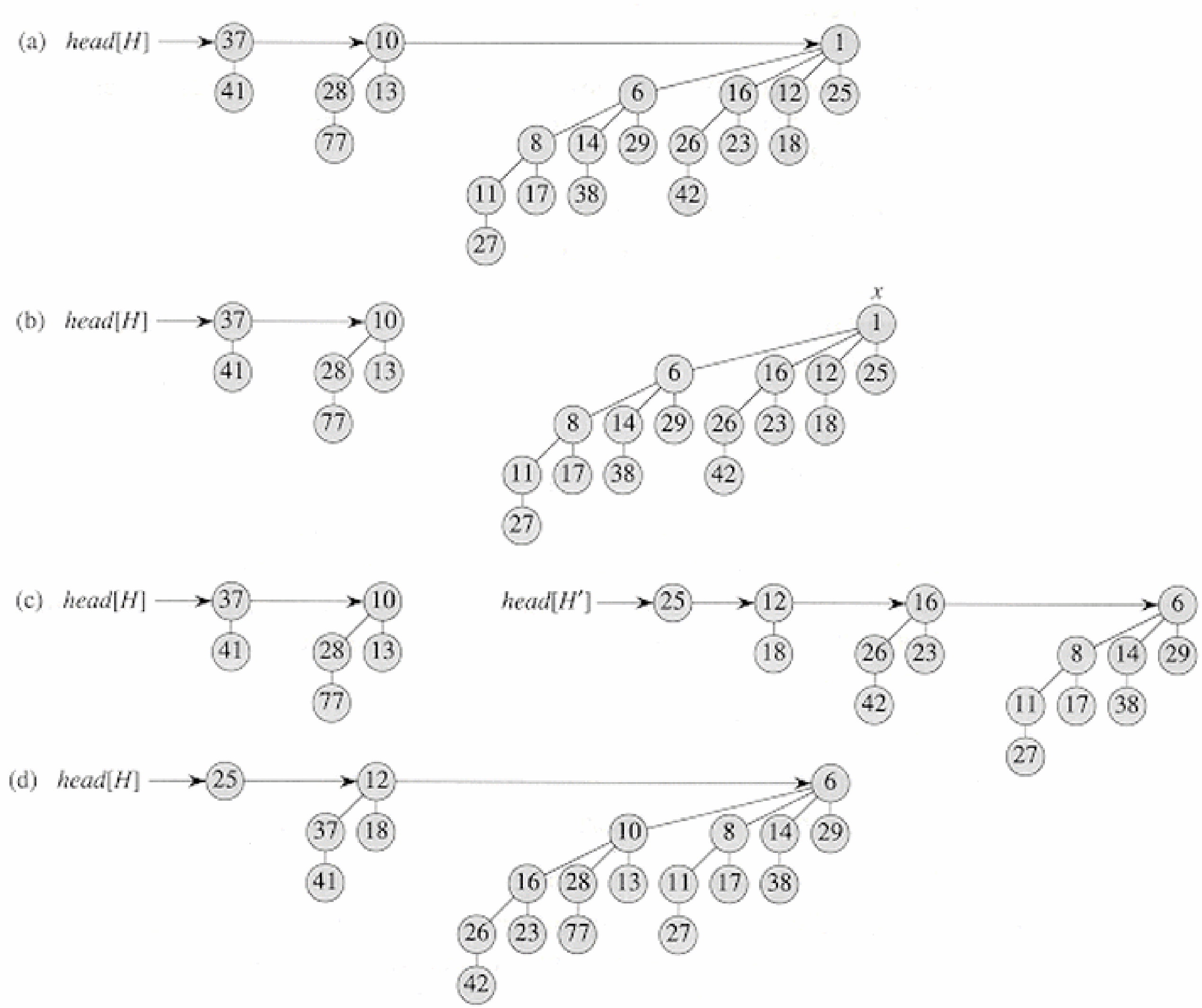
1 → $O(\lg n)$

2 → $O(1)$

3 → $O(\lg n)$

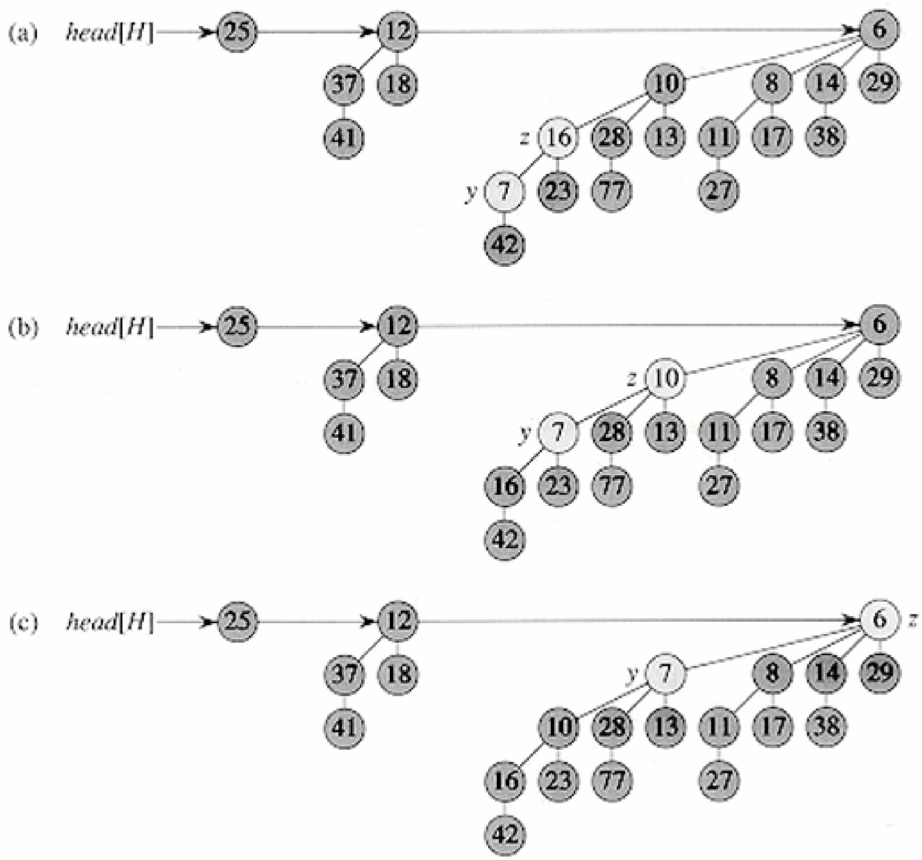
4 → $O(\lg n)$

QUINDI BINOMIAL-HEAP-EXTRACT-MIN HA COMPLESSITA' $O(\lg n)$


(a) head[H]
(b) head[H]
x
(c) head[H]
head[H′]
(d) head[H]

```
BINOMIAL-HEAP-DECREASE-KEY (H,x,k)
if k > key[x]
    then error "new key is greater than current key"
key[x] ← k
y ← x
z ← p[y]
while z ≠ NIL and key[y] < key[z]
    do exchange key[y] ↔ key[z]
       ▷ If y and z have satellite fields, exchange them, too.
       y ← z
       z ← p[y]
```

COMPLESSITA': $O(\lg n)$



(a)
head[H]
z
y
(b)
head[H]
z
y
(c)
head[H]
z
y

```
BINOMIAL-HEAP-DELETE(H,x)

1 BINOMIAL-HEAP-DECREASE-KEY(H,x,-∞)

2 BINOMIAL-HEAP-EXTRACT-MIN(H)
```

COMPLESSITA': $O(\lg n)$, CON $n = \#$ nodi(H)