بسم الله الرحمن الرحیم

طراحی و تحلیل الگوریتم‌ها

مبحث پانزدهم

مباحث پیشرفته در ساختمان داده‌ها

هیپ‌های دوجمله‌ای

Binomial Heaps

کاظم فولادی قلعه

دانشکده مهندسی، دانشکدگان فارابی

دانشگاه تهران

http://courses.fouladi.ir/algorithm

هیپ

HEAP

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

درخت دوجمله‌ای

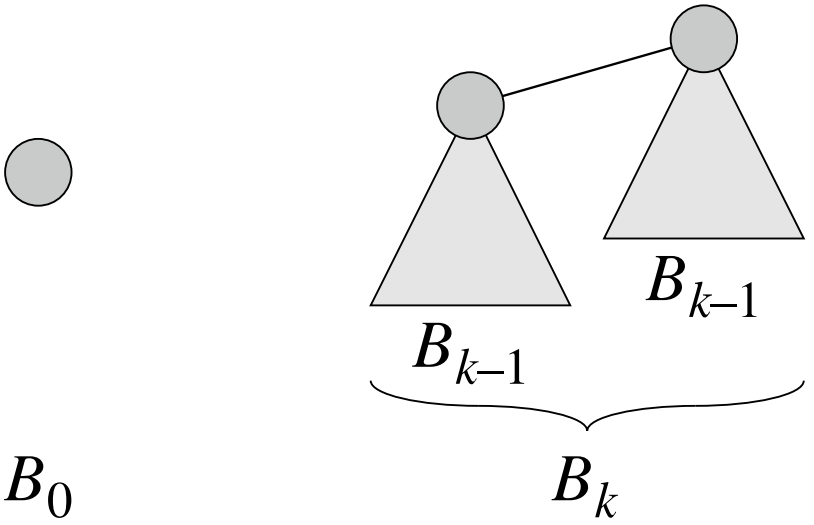
BINOMIAL TREE

درخت دوجمله‌ای
Binomial Tree

یک درخت مرتب B_k (که موقعیت قرارگیری فرزندان هر گره در آن مهم است) که به صورت بازگشتی تعریف می‌شود:

- اگر $k = 0$ باشد، یک گره‌ی تنها داریم.
- اگر $k > 0$ باشد، B_k به صورت زیر ساخته می‌شود:

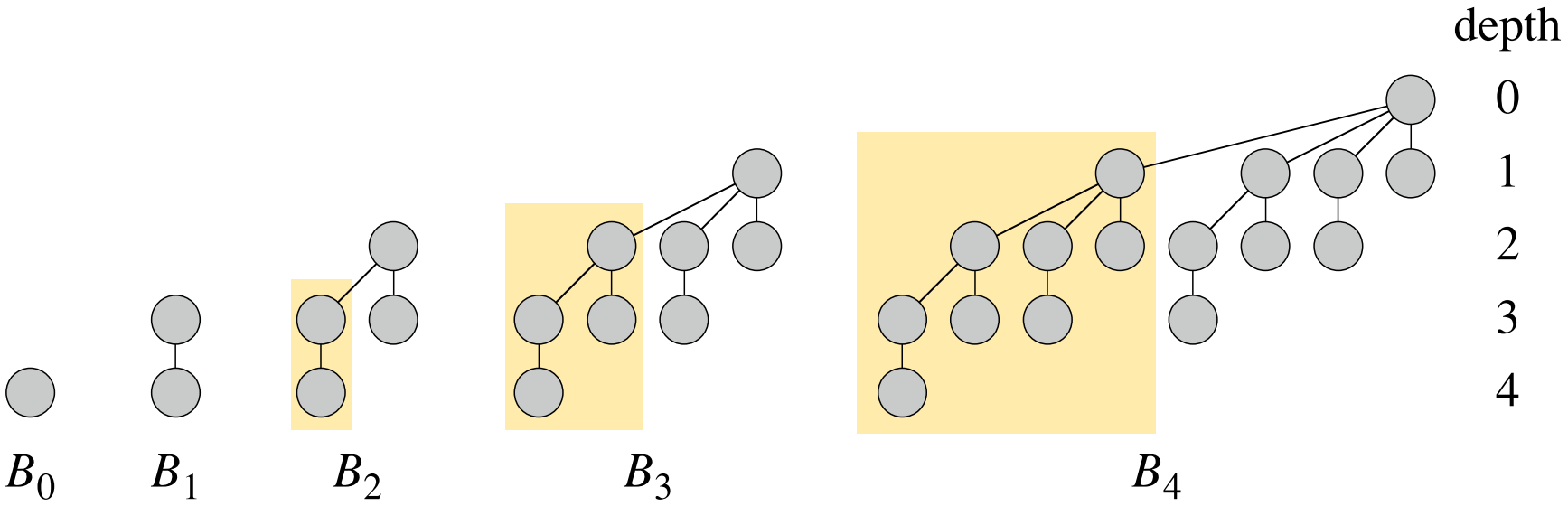
«یک درخت B_{k-1} را به عنوان فرزند سمت چپ به ریشه‌ی درخت B_{k-1} اضافه می‌کنیم.»



درخت دوجمله‌ای

مثال

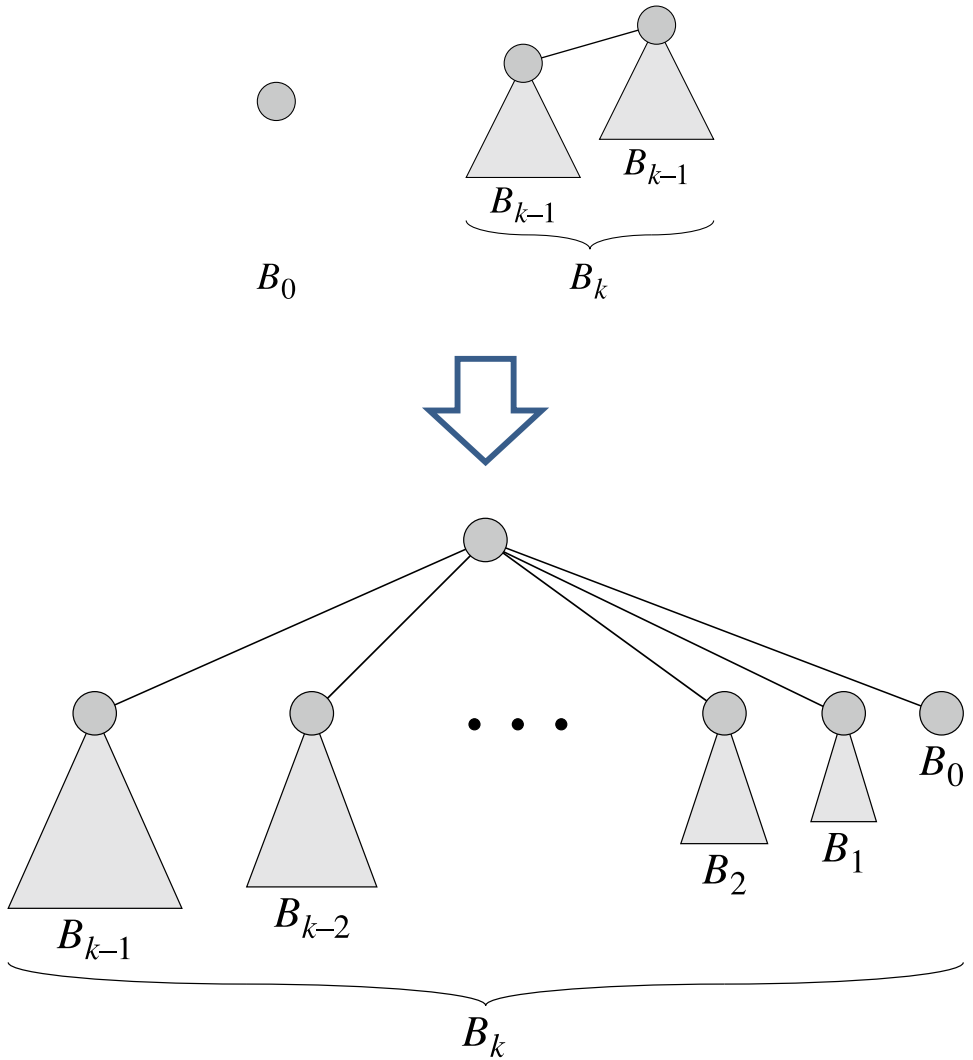
BINOMIAL TREE



درخت دوجمله‌ای

تعریف‌های ساختاری معادل

BINOMIAL TREE

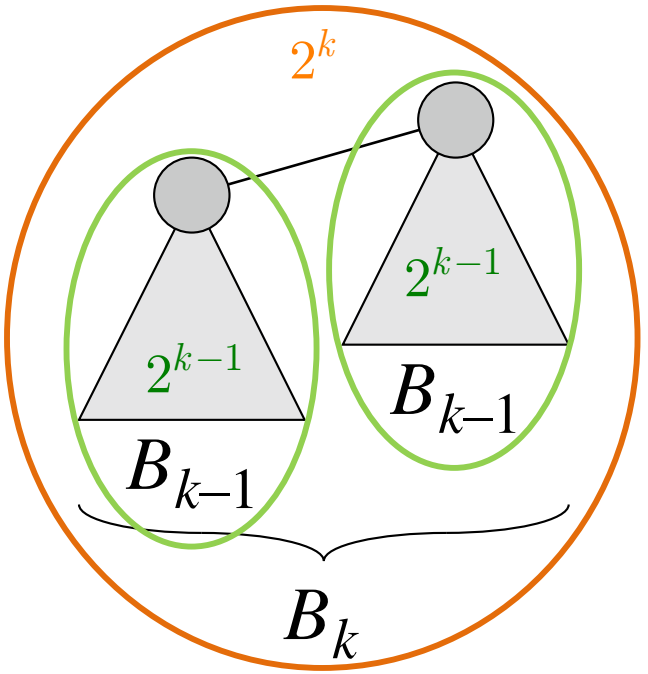


درخت دوجمله‌ای

ویژگی‌ها (۱)

BINOMIAL TREE

تعداد گره‌های درخت B_k برابر با 2^k است.



$$N(B_k) = N(B_{k-1}) + N(B_{k-1}) = 2N(B_{k-1})$$
$$N(B_0) = 1$$

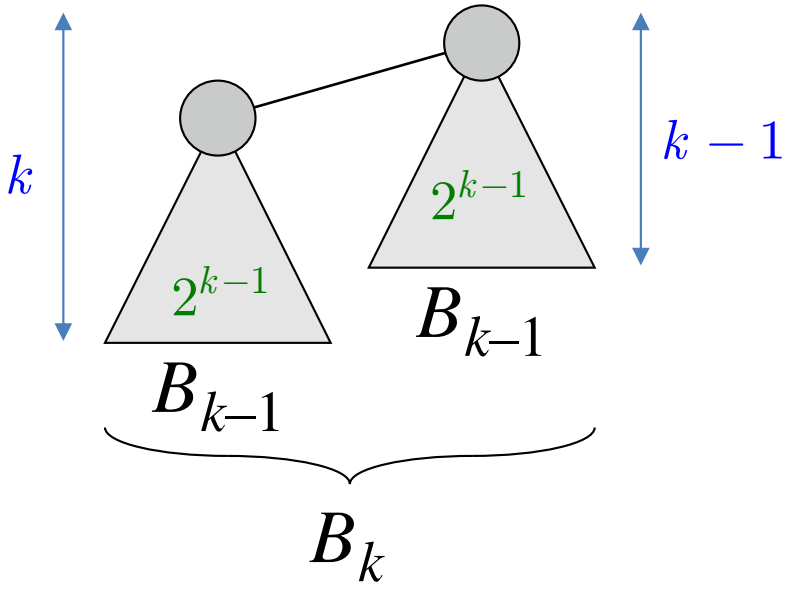
درخت دوجمله‌ای

ویژگی‌ها (۲)

BINOMIAL TREE

ارتفاع درخت B_k برابر با k است.

حداکثر عمق در درخت B_k یک واحد بیشتر از حداکثر عمق در B_{k-1} است.



$$h(B_k) = h(B_{k-1}) + 1$$
$$h(B_0) = 0$$

درخت دوجمله‌ای

ویژگی‌ها (۳)

BINOMIAL TREE

تعداد گره‌های سطح iام درخت B_k برابر با $\binom{k}{i}$ است.

اگر $D(k,i)$ تعداد گره‌ها در عمق i از درخت B_k باشد،
چون B_k از دو کپی B_{k-1} ساخته شده است،
هر گره در سطح i در B_{k-1} در B_k یک بار در سطح i و یک بار در سطح $i+1$ ظاهر می‌شود

$\Downarrow$

تعداد گره‌ها در سطح i در B_k = تعداد گره‌ها در سطح i در B_{k-1} + تعداد گره‌ها در سطح $i-1$ در B_{k-1}

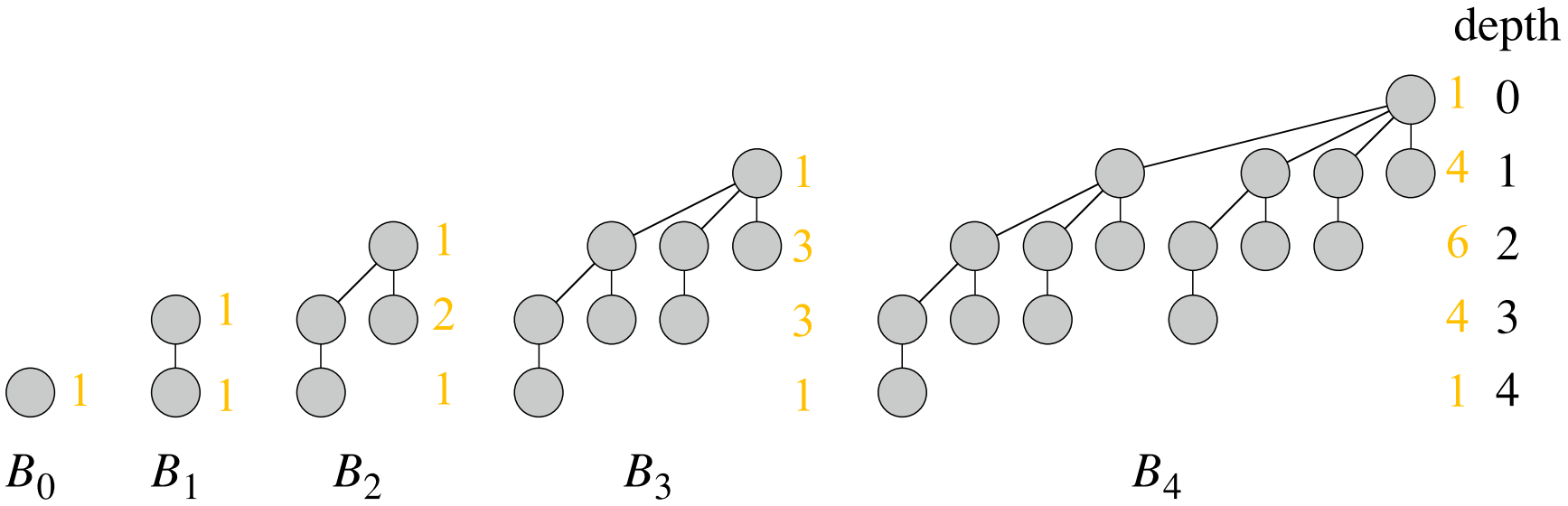
$$\begin{aligned} D(k,i) &= D(k-1,i) + D(k-1,i-1) \\ &= \binom{k-1}{i} + \binom{k-1}{i-1} \\ &= \binom{k}{i}. \end{aligned}$$

درخت دوجمله‌ای

ویژگی‌ها (۳): مثال

BINOMIAL TREE

تعداد گره‌های سطح iام درخت B_k برابر با $\binom{k}{i}$ است.

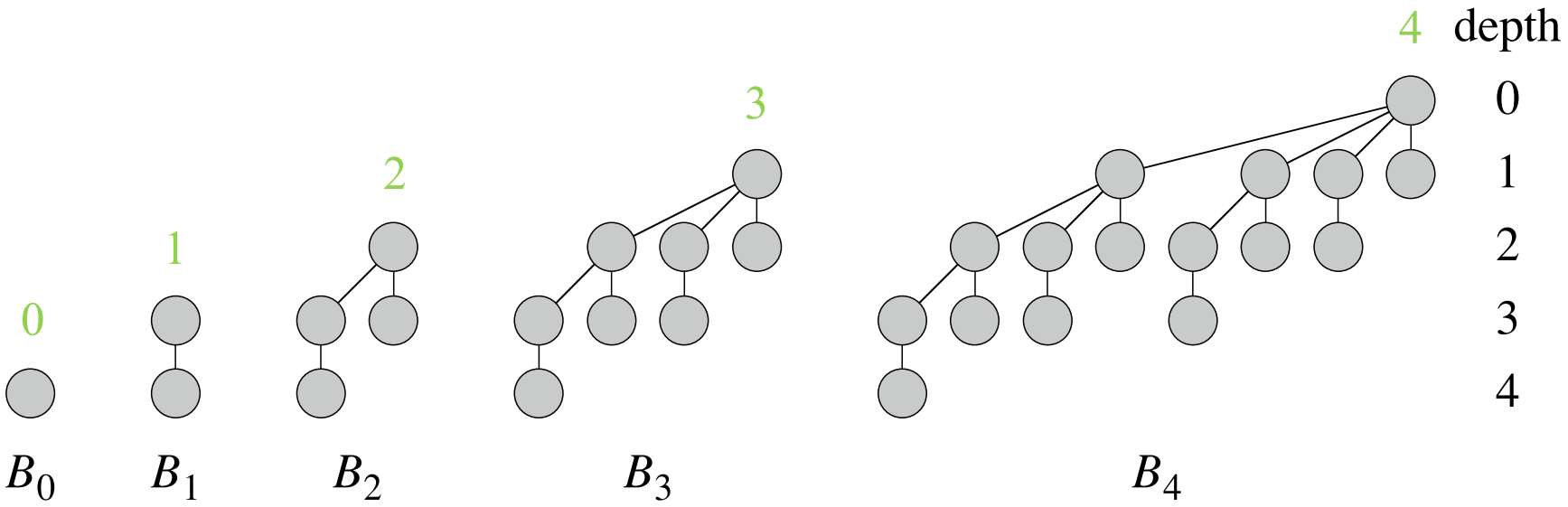


درخت دوجمله‌ای

ویژگی‌ها (۴)

BINOMIAL TREE

درجه‌ی ریشه‌ی درخت B_k برابر با k است.

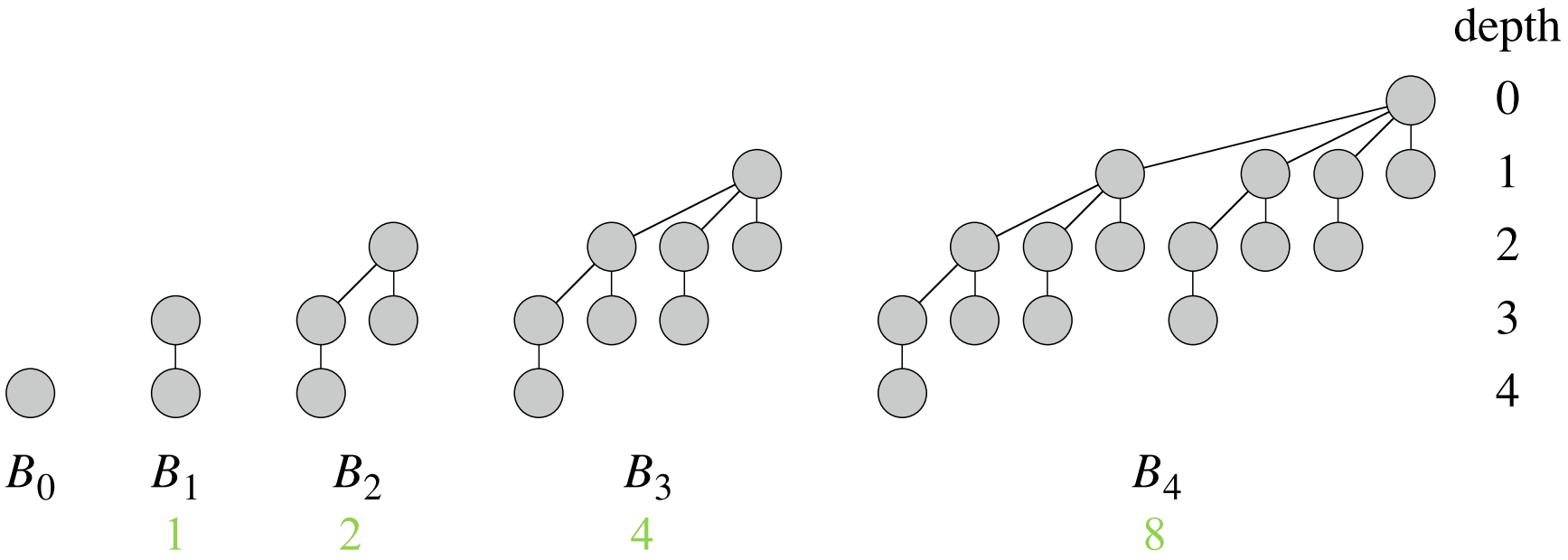


درخت دوجمله‌ای

ویژگی‌ها (۵)

BINOMIAL TREE

تعداد برگ‌های درخت B_k برابر با 2^{k-1} است (اگر $k > 0$).

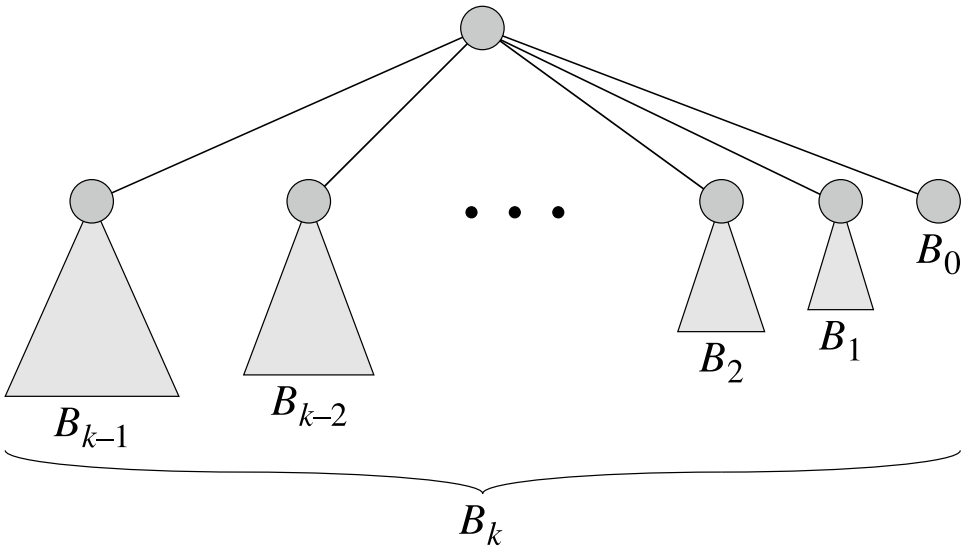


درخت دوجمله‌ای

ویژگی‌ها (۶)

BINOMIAL TREE

درخت B_k یک درخت مرتب است که درجه‌ی خروجی ریشه‌ی آن بین تمام درجات خروجی گره‌های آن حداکثر است و به‌علاوه
درخت B_k از راست به چپ، زیردرخت‌های B_0، B_1، ...، B_{k-1} را دارد.



درخت دوجمله‌ای

ویژگی‌ها (۷)

BINOMIAL TREE

ماکزیمم درجه‌ی خروجی هر گره در درخت دوجمله‌ای با n گره برابر با $\log_2 n$ است.

هیپ دوجمله‌ای

BINOMIAL HEAP

هیپ دوجمله‌ای
Binomial Heap

H یک مجموعه از درخت‌های دوجمله‌ای که دارای ویژگی‌های هیپ دوجمله‌ای باشد:

- هر درخت دوجمله‌ای در H در خاصیت min-heap صدق کند.
  (کلید هر گره بزرگ‌تر یا مساوی با کلید پدرش است ⇐ ریشه دارای کوچک‌ترین کلید است.)
- برای هر عدد صحیح نامنفی k **حداکثر یک** درخت دوجمله‌ای B_k در H وجود دارد.
  (که این درخت ریشه‌ای با درجه‌ی k دارد)

هیپ دوجمله‌ای

ویژگی

BINOMIAL HEAP

هیپ دوجمله‌ای با n گره شامل حداکثر $\lfloor \log_2 n \rfloor + 1$ درخت دوجمله‌ای است.

نمایش دودویی عدد n دارای $\lfloor \log_2 n \rfloor + 1$ بیت است، به طوری که $n = \sum_{i=0}^{\lfloor \log_2 n \rfloor} b_i 2^i$.

درخت دوجمله‌ای B_i در H ظاهر می‌شود اگر و فقط اگر بیت $b_i = 1$ باشد.

$\Downarrow$

هیپ دوجمله‌ای با n گره شامل حداکثر $\lfloor \log_2 n \rfloor + 1$ درخت دوجمله‌ای است.

پس در هیپ دوجمله‌ای با n گره، حداکثر تعداد ریشه‌ها $\lfloor \log_2 n \rfloor + 1$ است.

هیپ دوجمله‌ای

ویژگی

BINOMIAL HEAP

برای هر عدد طبیعی می‌توان یک هیپ دوجمله‌ای با همان تعداد گره ساخت.

$$n = 13$$

$$(1101)_2 = 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1$$

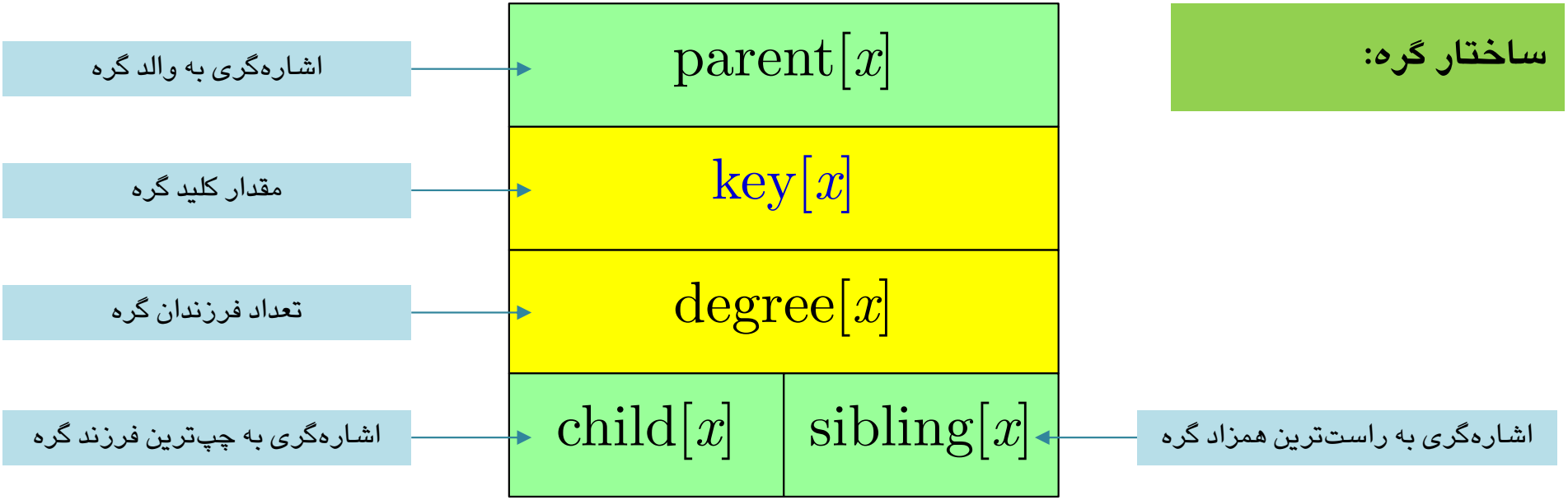
$$B = (b_3, b_2, b_0)$$

هیپ دوجمله‌ای

بازنمایی

BINOMIAL HEAP

ساختار گره:



ساختار ساختمان داده:

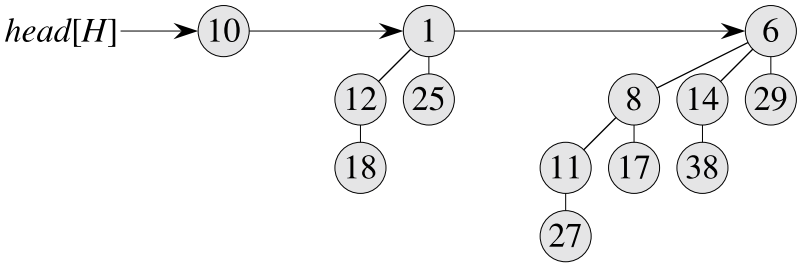
ریشه‌های درخت‌های هیپ دوجمله‌ای به ترتیب افزایش درجه در یک لیست پیوندی قرار می‌گیرند (لیست ریشه)

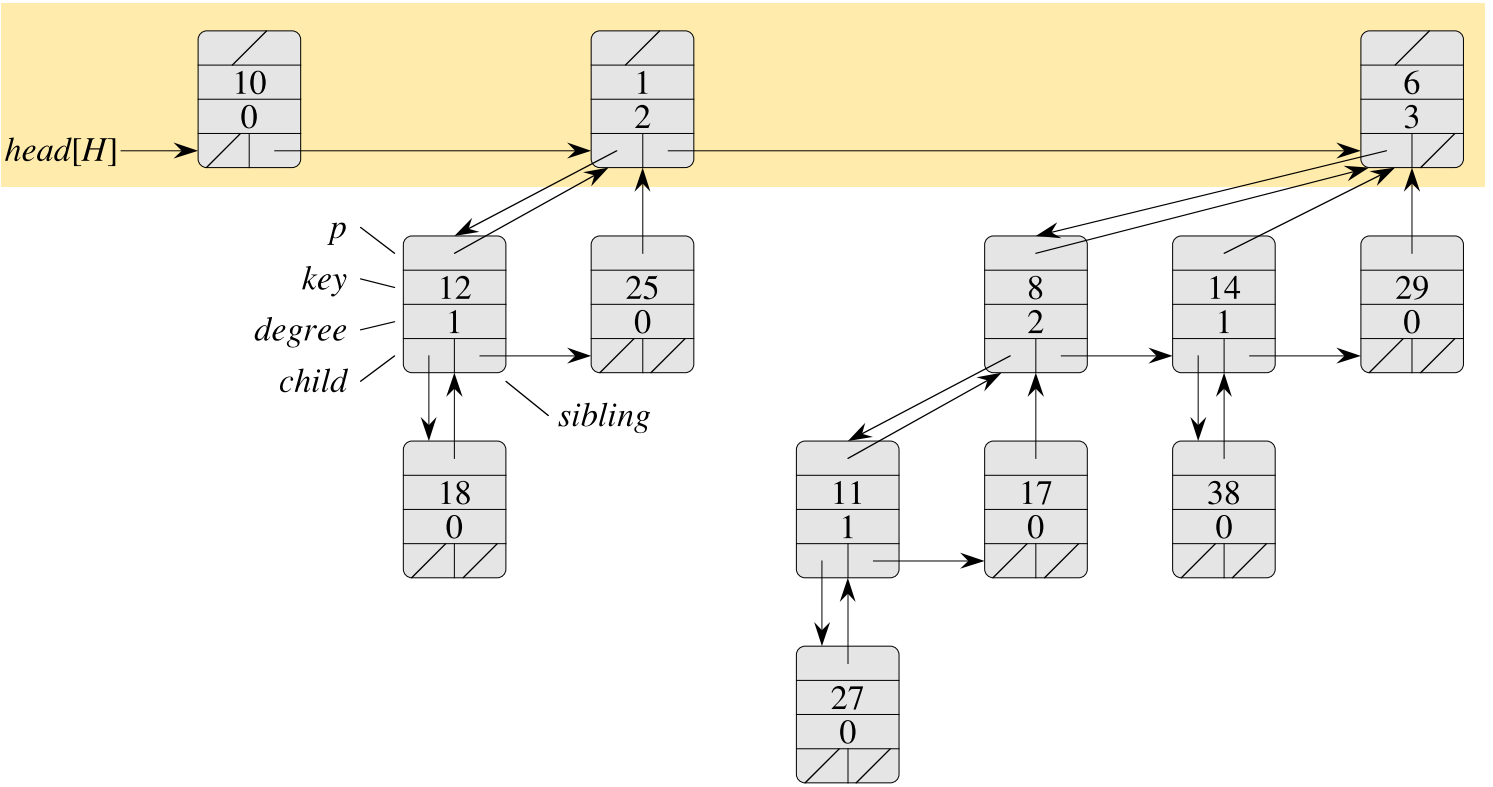
اشاره‌گر $\text{head}[H]$ به اولین ریشه اشاره می‌کند.

هیپ دوجمله‌ای

بازنمایی: مثال

BINOMIAL HEAP





عملیات بر روی هیپ دوجمله‌ای

ایجاد یک هیپ دوجمله‌ای جدید

MAKE-BINOMIAL-HEAP()

برای شیئ H حافظه می‌گیرد و اشاره‌گر به آن را برمی‌گرداند که $\text{head}[H] = \text{NIL}$

زمان اجرا: $\Theta(1)$

عملیات بر روی هیپ دوجمله‌ای

یافتن کوچک‌ترین کلید

BINOMIAL-HEAP-MINIMUM(H)

فقط ریشه‌ها مقایسه می‌شوند؛ اشاره‌گر به آن را برمی‌گرداند.

```
BINOMIAL-HEAP-MINIMUM(H)
1   y ← NIL
2   x ← head[H]
3   min ← ∞
4   while x ≠ NIL
5         do if key[x] < min
6               then min ← key[x]
7                    y ← x
8            x ← sibling[x]
9   return y
```

چون تعداد ریشه‌ها حداکثر $\lfloor \log_2 n \rfloor + 1$ است:

زمان اجرا: $\Theta(\log n)$

عملیات بر روی هیپ دوجمله‌ای

اجتماع دو هیپ دوجمله‌ای

BINOMIAL-HEAP-UNION(H_1, H_2)

به طور پی‌درپی درخت‌های دوجمله‌ای دارای ریشه‌های هم‌درجه را به هم متصل می‌کند.

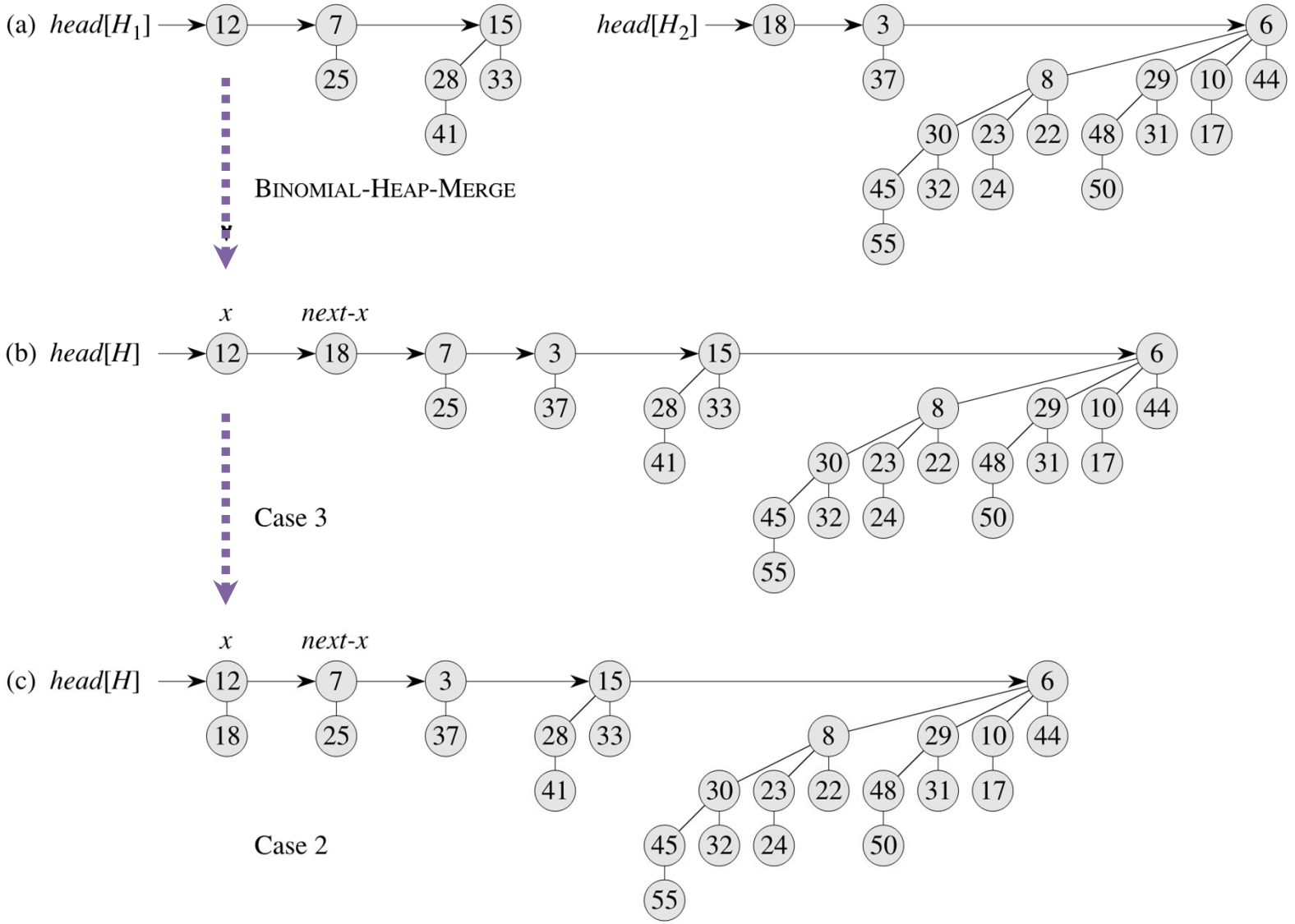
```
BINOMIAL-HEAP-UNION(H1, H2)
 1  H ← MAKE-BINOMIAL-HEAP()
 2  head[H] ← BINOMIAL-HEAP-MERGE(H1, H2)
 3  free the objects H1 and H2 but not the lists they point to
 4  if head[H] = NIL
 5     then return H
 6  prev-x ← NIL
 7  x ← head[H]
 8  next-x ← sibling[x]
 9  while next-x ≠ NIL
10        do if (degree[x] ≠ degree[next-x]) or
                 (sibling[next-x] ≠ NIL and degree[sibling[next-x]] = degree[x])
11              then prev-x ← x                                  ▷ Cases 1 and 2
12                   x ← next-x                                  ▷ Cases 1 and 2
13              else if key[x] ≤ key[next-x]
14                      then sibling[x] ← sibling[next-x]        ▷ Case 3
15                           BINOMIAL-LINK(next-x, x)            ▷ Case 3
16                      else if prev-x = NIL                     ▷ Case 4
17                              then head[H] ← next-x            ▷ Case 4
18                              else sibling[prev-x] ← next-x    ▷ Case 4
19                           BINOMIAL-LINK(x, next-x)            ▷ Case 4
20                           x ← next-x                          ▷ Case 4
21           next-x ← sibling[x]
22  return H
```

```
BINOMIAL-LINK(y, z)
1  p[y] ← z
2  sibling[y] ← child[z]
3  child[z] ← y
4  degree[z] ← degree[z] + 1
```

زمان اجرا: $\Theta(\log n)$

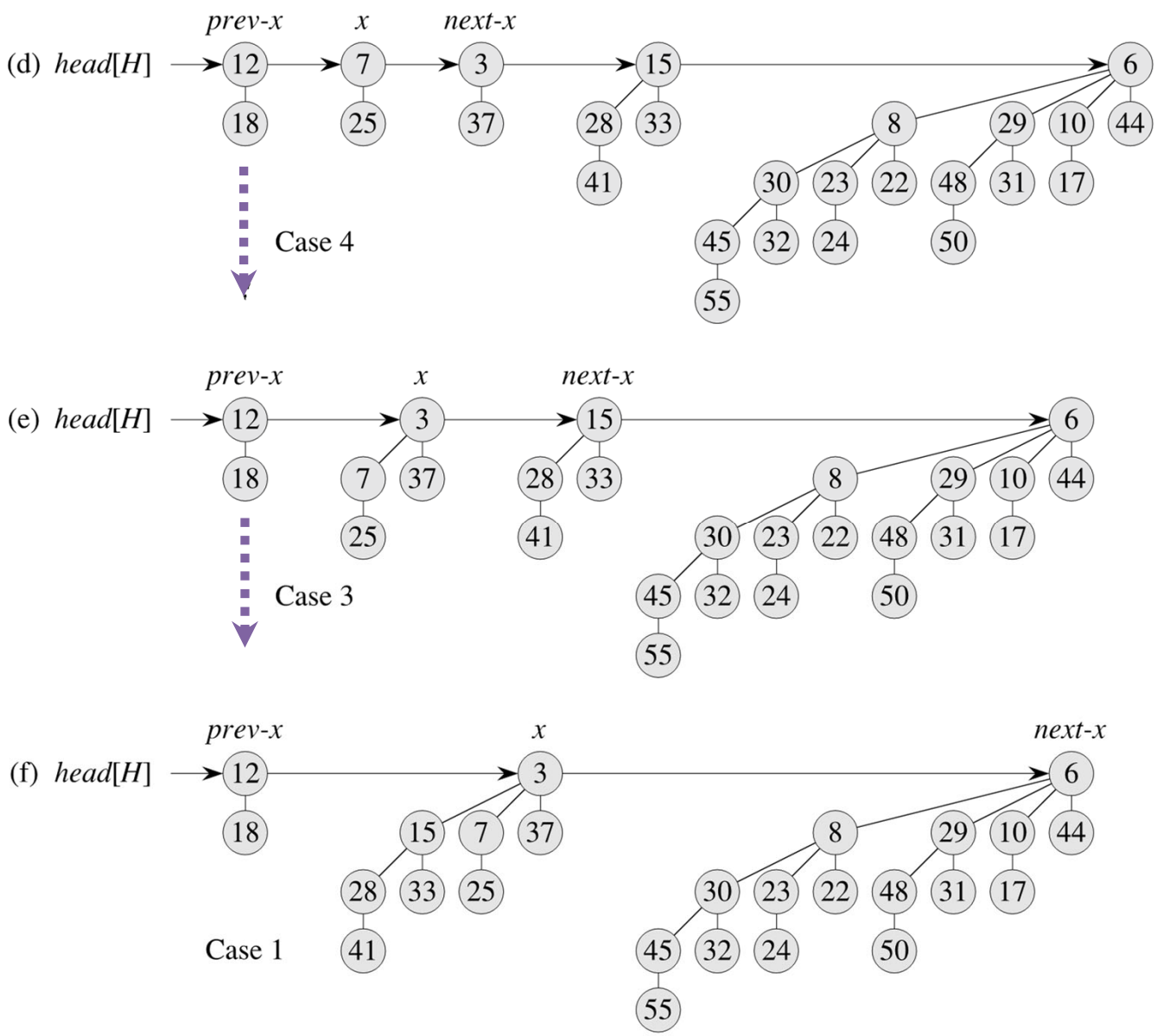
عملیات بر روی هیپ دوجمله‌ای

اجتماع دو هیپ دوجمله‌ای: مثال (۱ از ۲)



عملیات بر روی هیپ دوجمله‌ای

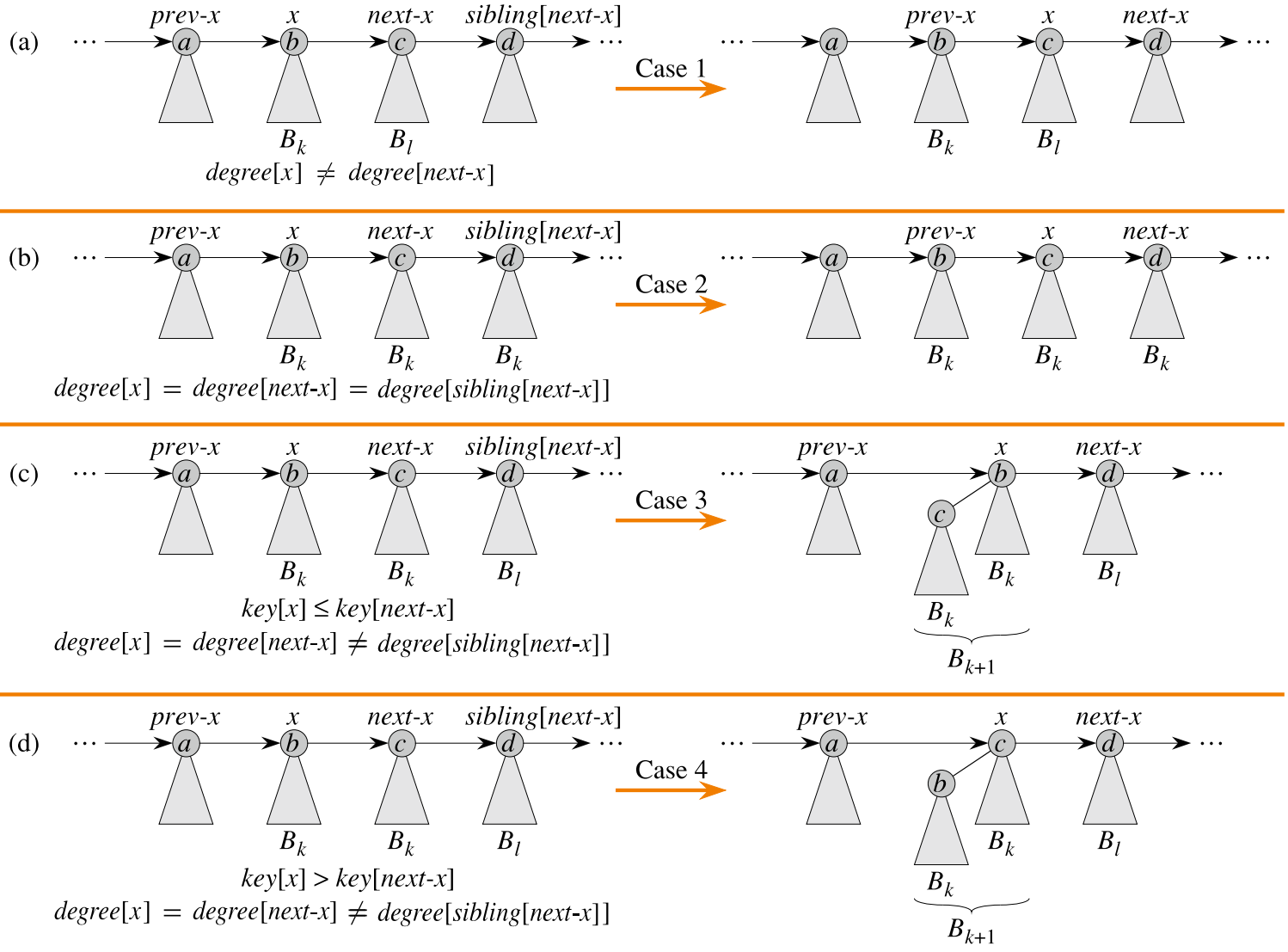
اجتماع دو هیپ دوجمله‌ای: مثال (۲ از ۲)



عملیات بر روی هیپ دوجمله‌ای

اجتماع دو هیپ دوجمله‌ای: حالت‌های چهارگانه

x is the root of a B_k-tree and $l > k$



عملیات بر روی هیپ دوجمله‌ای

درج یک گره در هیپ دوجمله‌ای

BINOMIAL-HEAP-INSERT(H,x)

یک هیپ دوجمله‌ای جدید با یک گره می‌سازد و سپس آن را با هیپ قبلی اجتماع می‌گیرد.

```
BINOMIAL-HEAP-INSERT(H, x)
1  H' ← MAKE-BINOMIAL-HEAP()
2  p[x] ← NIL
3  child[x] ← NIL
4  sibling[x] ← NIL
5  degree[x] ← 0
6  head[H'] ← x
7  H ← BINOMIAL-HEAP-UNION(H, H')
```

زمان اجرا: $\Theta(\log n)$

عملیات بر روی هیپ دوجمله‌ای

استخراج گره با کلید می‌نیمم در هیپ دوجمله‌ای

BINOMIAL-HEAP-EXTRACT-MIN(H)

ریشه‌ی دارای کلید می‌نیمم را حذف می‌کند و مقدار آن را برمی‌گرداند.

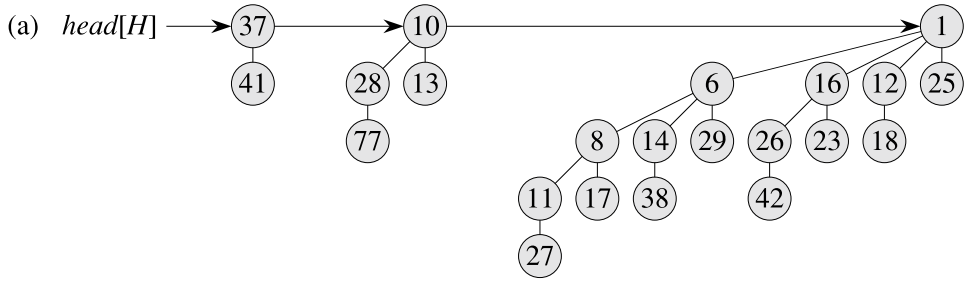
```
BINOMIAL-HEAP-EXTRACT-MIN(H)
1  find the root x with the minimum key in the root list of H,
       and remove x from the root list of H
2  H' ← MAKE-BINOMIAL-HEAP()
3  reverse the order of the linked list of x's children,
       and set head[H'] to point to the head of the resulting list
4  H ← BINOMIAL-HEAP-UNION(H,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

زمان اجرا: $\Theta(\log n)$

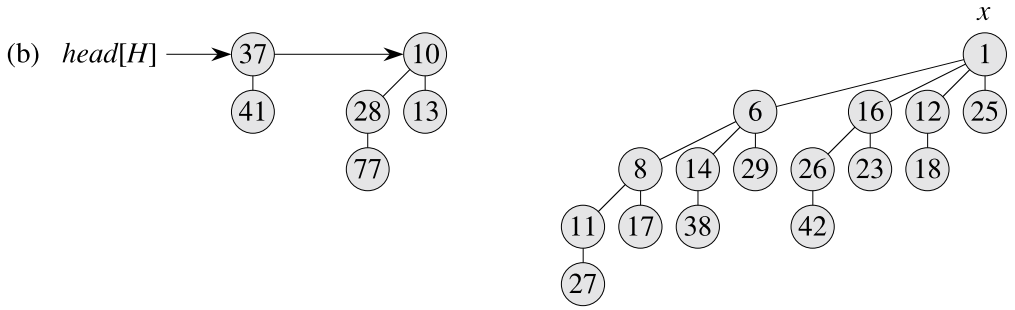
عملیات بر روی هیپ دوجمله‌ای

استخراج گره با کلید می‌نیمم در هیپ دوجمله‌ای: مثال

(a)

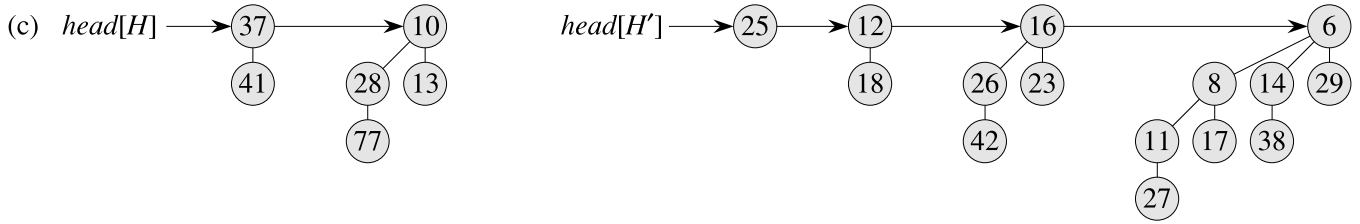


(b)



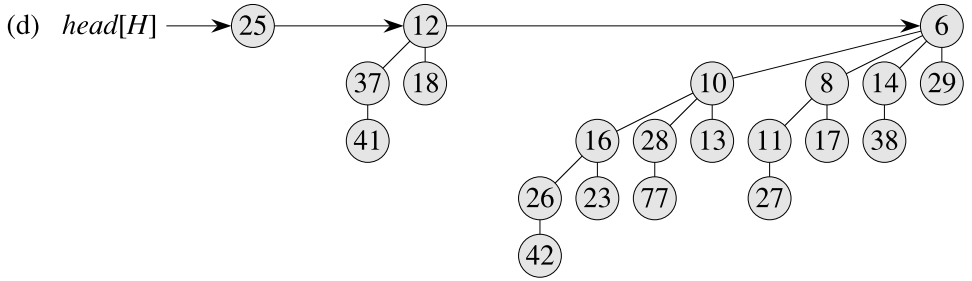
لیست زیردرخت‌های ریشه به ترتیب معکوس در هیپ H' درج می‌شود:

(c)



اجتماع دو هیپ حاصل را به‌دست می‌آوریم:

(d)



عملیات بر روی هیپ دوجمله‌ای

کاهش یک کلید در هیپ دوجمله‌ای

BINOMIAL-HEAP-DECREASE-KEY(H,x,k)

مقدار کلید x را به k کاهش می‌دهد.

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

مقدار کلید کاهش‌یافته را با والدش مقایسه می‌کند و در صورت بزرگ‌تر بودن والد، با آن تعویض می‌شود.

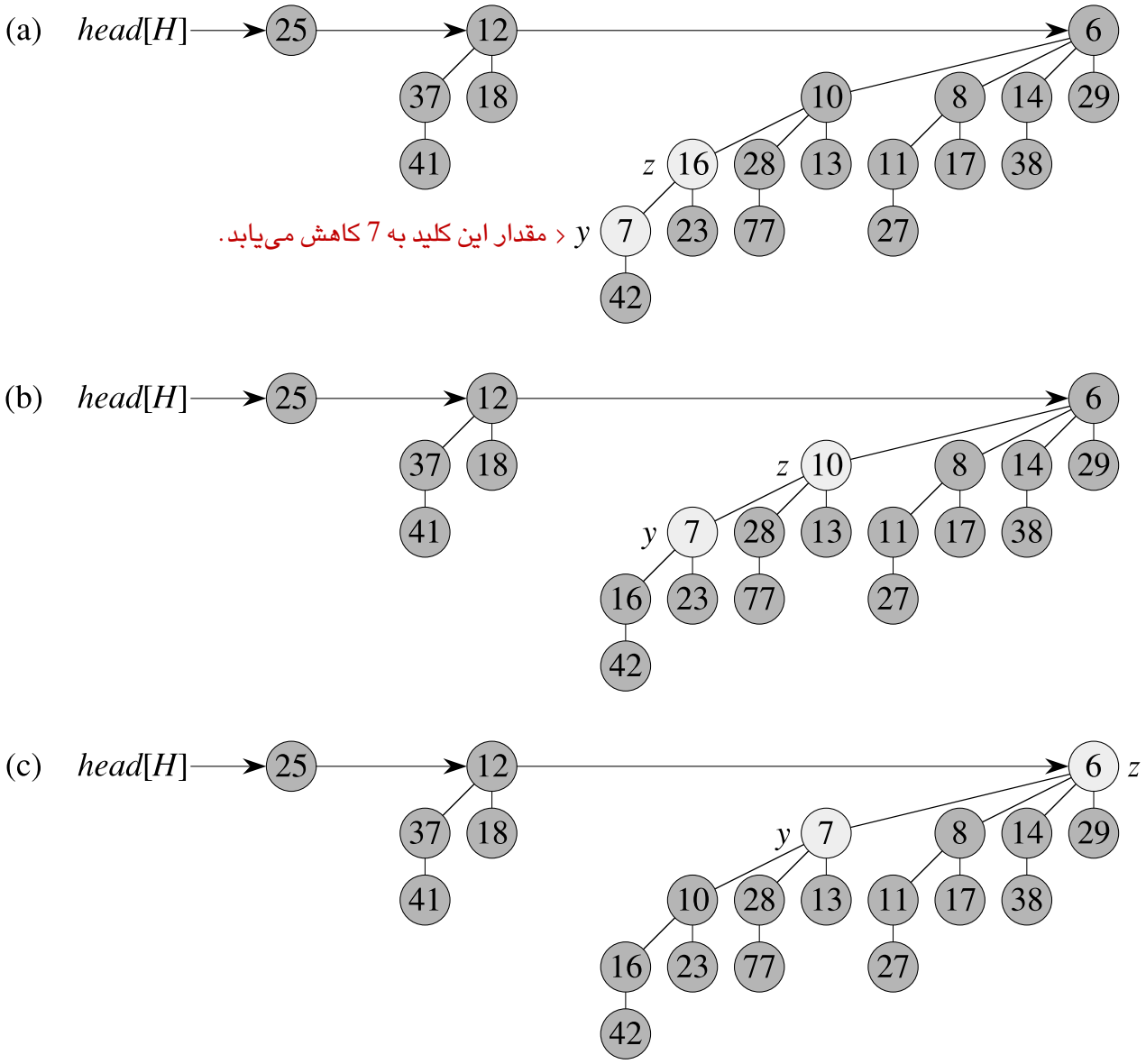
این عمل آن‌قدر تکرار می‌شود تا به والدی برسد که از آن کوچک‌تر باشد.

زمان اجرا متناسب با ارتفاع درخت است، پس:

زمان اجرا: $\Theta(\log n)$

عملیات بر روی هیپ دوجمله‌ای

کاهش یک کلید در هیپ دوجمله‌ای: مثال



عملیات بر روی هیپ دوجمله‌ای

حذف یک کلید در هیپ دوجمله‌ای

BINOMIAL-HEAP-DELETE(H,x)

کلید x را از هیپ دوجمله‌ای حذف می‌کند.

```
BINOMIAL-HEAP-DELETE(H, x)
1  BINOMIAL-HEAP-DECREASE-KEY(H, x, −∞)
2  BINOMIAL-HEAP-EXTRACT-MIN(H)
```

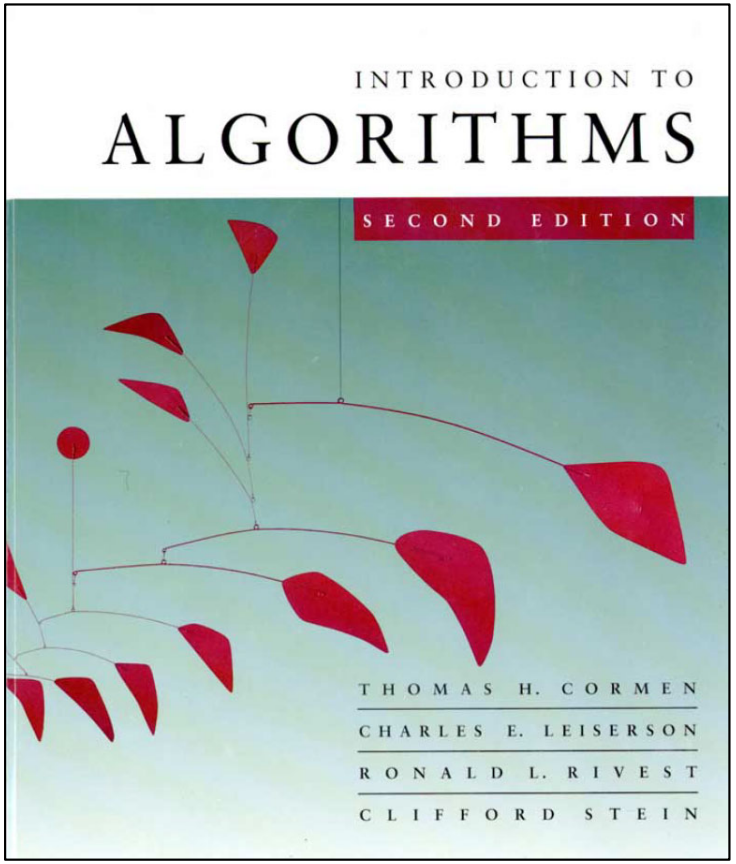
زمان اجرا: $\Theta(\log n)$

عملیات بر روی هیپ دوجمله‌ای

خلاصه‌ی زمان‌های اجرا

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

مرجع



T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein,
Introduction to Algorithms,
2nd Edition, MIT Press, 2001.

Chapter 19

19 Binomial Heaps

This chapter and Chapter 20 present data structures known as ***mergeable heaps***, which support the following five operations.

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.

In addition, the data structures in these chapters also support the following two operations.

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.[1]

DELETE(H, x) deletes node x from heap H.

As the table in Figure 19.1 shows, if we don't need the UNION operation, ordinary binary heaps, as used in heapsort (Chapter 6), work well. Operations other than UNION run in worst-case time $O(\lg n)$ (or better) on a binary heap. If the UNION operation must be supported, however, binary heaps perform poorly. By concatenating the two arrays that hold the binary heaps to be merged and then running MIN-HEAPIFY (see Exercise 6.2-2), the UNION operation takes $\Theta(n)$ time in the worst case.

[1] As mentioned in the introduction to Part V, our default mergeable heaps are mergeable min-heaps, and so the operations MINIMUM, EXTRACT-MIN, and DECREASE-KEY apply. Alternatively, we could define a ***mergeable max-heap*** with the operations MAXIMUM, EXTRACT-MAX, and INCREASE-KEY.