20.2-3. Wykaż, że jeśli wykonujemy tylko operacje kopca złączalnego, to maksymalny stopień węzła $D(n)$ w n-węzłowym kopcu Fibonacciego nie przekracza $\lfloor \lg n \rfloor$.

20.2-4. Profesor McGee obmyślił nową strukturę danych opartą na kopcach Fibonacciego. Kopiec profesora ma taką samą strukturę jak kopiec Fibonacciego i umożliwia wykonywanie wszystkich operacji kopca złączalnego. Implementacja tych operacji jest taka sama jak dla kopców Fibonacciego z tą różnicą, że przy wstawianiu elementu i łączeniu dwóch kopców w ostatnim kroku skracamy listę korzeni, wywołując procedurę CONSOLIDATE. Jakie są w najgorszym przypadku czasy wykonywania poszczególnych operacji w takiej strukturze? Na ile nowatorski jest pomysł profesora?

20.2-5. Uzasadnij, że jeśli jedynymi operacjami na kluczach są porównania dwóch kluczy (jak ma to miejsce we wszystkich przedstawionych w tym rozdziale implementacjach), to nie wszystkie operacje kopca złączalnego mogą działać w czasie zamortyzowanym $O(1)$.

20.3. Zmniejszanie wartości klucza i usuwanie węzła

W tym podrozdziale pokazujemy, jak zmniejszać wartość klucza w węźle kopca Fibonacciego w czasie zamortyzowanym $O(1)$ oraz jak usuwać dowolny węzeł z n-węzłowego kopca Fibonacciego w czasie zamortyzowanym $O(D(n))$. Jeśli stosujemy te operacje, to nie mamy gwarancji, że wszystkie drzewa w kopcu Fibonacciego będą nieuporządkowanymi drzewami dwumianowymi. Będą jednak na tyle do nich zbliżone, że maksymalny stopień węzła $D(n)$ da się oszacować jako $O(\lg n)$. Z oszacowania tego będzie wynikać to, że procedury FIB-HEAP-EXTRACT-MIN i FIB-HEAP-DELETE działają w zamortyzowanym czasie $O(\lg n)$.

Zmniejszanie wartości klucza

W przedstawionym poniżej pseudokodzie operacji FIB-HEAP-DECREASE-KEY zakładamy jak poprzednio, że usunięcie węzła z listy nie zmienia zawartości żadnych pól w strukturze reprezentującej usuwany węzeł.

```
FIB-HEAP-DECREASE-KEY(H, x, k)
1  if k > key[x]
2      then error „nowa wartość klucza jest większa niż bieżąca"
3  key[x] ← k
4  y ← p[x]
5  if y ≠ NIL i key[x] < key[y]
```

```
6       then CUT(H, x, y)
7            CASCADING-CUT(H, y)
8   if key[x] < key[min[H]]
9     then min[H] ← x
```

```
CUT(H, x, y)
1  usuń x z listy synów y i zmniejsz degree[y] o 1
2  dodaj x do listy korzeni kopca H
3  p[x] ← NIL
4  mark[x] ← FALSE
```

```
CASCADING-CUT(H, y)
1  z ← p[y]
2  if z ≠ NIL
3    then if mark[y] = FALSE
4           then mark[y] ← TRUE
5           else CUT(H, y, z)
6                CASCADING-CUT(H, z)
```

Procedura FIB-HEAP-DECREASE-KEY działa następująco. W wierszach 1–3 jest sprawdzane, czy nowa wartość klucza nie jest większa niż bieżący klucz w węźle x, a następnie jest zapisywany w x nowy klucz. Jeśli x jest korzeniem lub $key[x] \geq key[y]$, gdzie y jest ojcem x, to nie ma potrzeby zmian w strukturze kopca, ponieważ porządek kopcowy nie został naruszony. Warunek ten jest sprawdzany w wierszach 4–5.

Jeśli porządek kopcowy został naruszony, to w kopcu może zajść wiele zmian. Najpierw jest **odcinany** węzeł x w wierszu 6. Procedura CUT „odcina" połączenie między x a jego ojcem y, czyniąc x korzeniem.

W celu otrzymania zapowiadanego oszacowania czasu wykonywania operacji są wykorzystywane pola *mark*. W każdym węźle przechowują one mały fragment jego historii. Przypuśćmy, że z węzłem x działy się następujące rzeczy:

1. W pewnej chwili x był korzeniem.
2. Następnie x został dołączony do innego węzła.
3. Potem od x zostało odciętych dwóch jego synów.

Zaraz po utracie drugiego syna węzeł x zostaje odcięty od swojego ojca i zostaje nowym korzeniem. Pole $mark[x]$ ma wartość TRUE, jeśli zaszły zdarzenia 1 i 2 oraz został odcięty jeden z synów x. W wierszu 4 procedury CUT pole $mark[x]$ przyjmuje zatem wartość FALSE, ponieważ następuje właśnie zdarzenie 1. (Widzimy teraz, dlaczego w wierszu 3 procedury FIB-HEAP-LINK pole $mark[y]$ przyjmuje wartość FALSE: węzeł y zostaje dołączony do innego węzła, zachodzi

więc zdarzenie 2. Następnym razem, kiedy zostanie odcięty syn węzła y, pole $mark[y]$ przyjmie wartość TRUE).

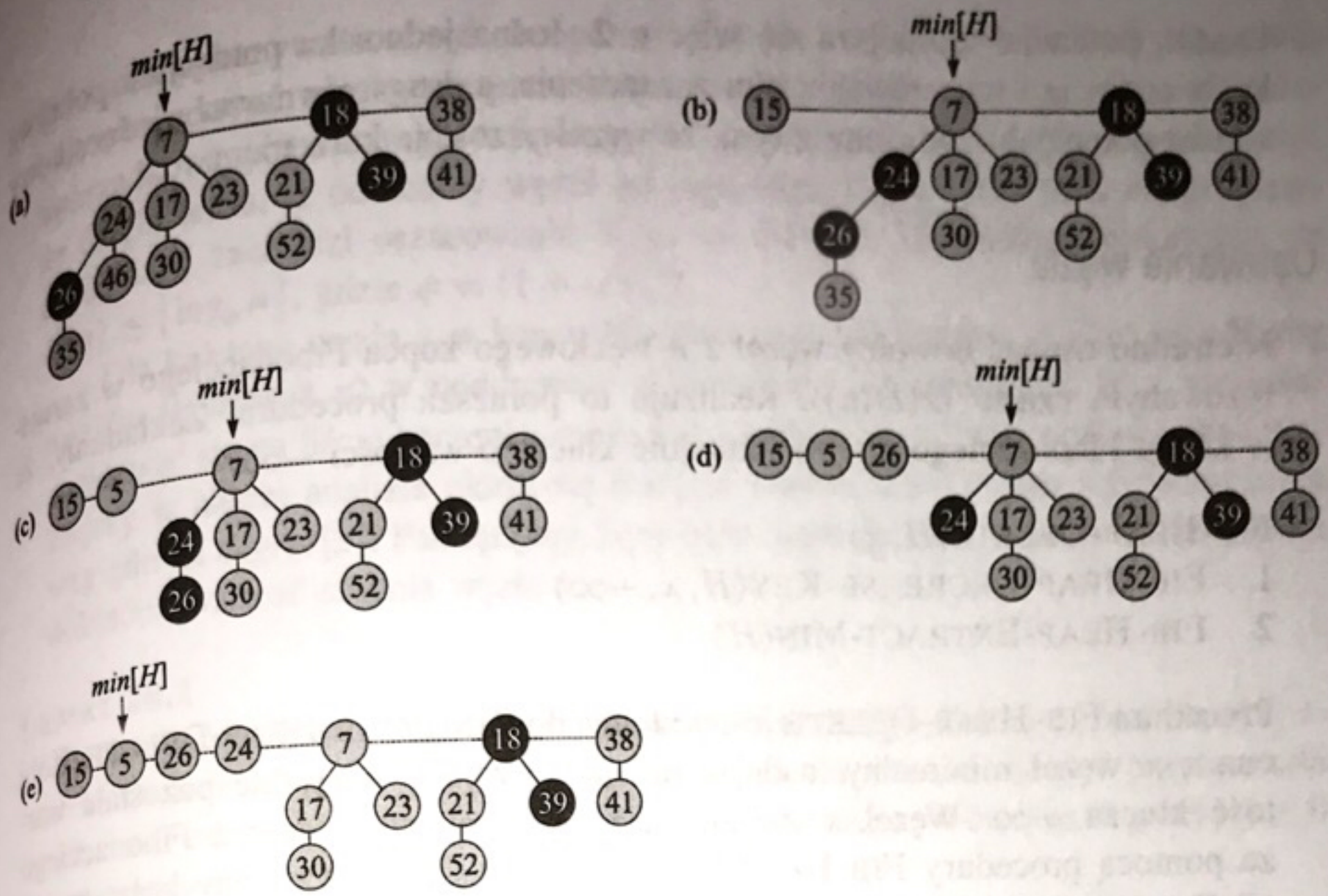
To jeszcze nie koniec, ponieważ x mógł być drugim synem odciętym od swojego ojca y, licząc od chwili, kiedy y został dołączony do innego węzła. W wierszu 7 procedury FIB-HEAP-DECREASE-KEY jest zatem na węźle y wykonywana operacja **odcięcia kaskadowego**. Jeśli y jest korzeniem, to sprawdzenie warunku w wierszu 2 procedury CASCADING-CUT powoduje jedynie powrót z wywołania. Jeśli y nie jest zaznaczony, to w wierszu 4 zostaje zaznaczony, ponieważ jego pierwszy syn został właśnie odcięty, po czym następuje powrót z wywołania. Jeśli natomiast y jest zaznaczony, to oznacza, że utracił on właśnie drugiego syna; w wierszu 5 y zostaje odcięty, a w wierszu 6 następuje rekurencyjne wywołanie procedury CASCADING-CUT dla węzła z będącego ojcem y. Rekurencyjne wywołania procedury CASCADING-CUT „wędrują" w ten sposób w górę drzewa, aż do napotkania korzenia lub niezaznaczonego węzła.

Po wykonaniu wszystkich kaskadowych odcięć procedura FIB-HEAP-DECREASE-KEY kończy działanie, uaktualniając (jeśli trzeba) wartość $min[H]$ w wierszach 8–9. Jedynym węzłem ze zmienioną wartością klucza jest x, którego klucz został zmniejszony. Nowym węzłem minimalnym jest zatem poprzedni węzeł minimalny albo x.

Na rysunku 20.4 widać przebieg dwóch wywołań procedury FIB-HEAP-DECREASE-KEY, zaczynając od kopca Fibonacciego z rys. 20.4(a). W pierwszym wywołaniu, przedstawionym na rys. 20.4(b), nie ma kaskadowych odcięć. W drugim, widocznym na rys. 20.4(c)–(e), występują dwa kaskadowe odcięcia.

Wykażemy teraz, że zamortyzowany koszt operacji FIB-HEAP-DECREASE-KEY wynosi tylko $O(1)$. Zaczniemy od wyznaczenia kosztu faktycznego. Czas działania procedury FIB-HEAP-DECREASE-KEY jest stały, jeśli nie liczyć czasu spędzonego na wykonywaniu kaskadowych odcięć. Przypuśćmy, że w trakcie wykonywania procedury FIB-HEAP-DECREASE-KEY nastąpiło c rekurencyjnych wywołań procedury CASCADING-CUT. Każde wywołanie procedury CASCADING-CUT zajmuje czas stały, jeśli pominąć wywołania rekurencyjne. Zatem faktyczny koszt procedury FIB-HEAP-DECREASE-KEY, z uwzględnieniem wszystkich wywołań rekurencyjnych, wynosi $O(c)$.

Obliczymy teraz zmianę potencjału. Oznaczmy przez H kopiec Fibonacciego tuż przed operacją FIB-HEAP-DECREASE-KEY. Przy każdym rekurencyjnym wywołaniu procedury CASCADING-CUT, z wyjątkiem ostatniego, jest odcinany zaznaczony węzeł i polu *mark* jest nadawana wartość FALSE. Na końcu mamy $t(H) + c$ drzew (początkowych $t(H)$ drzew, $c - 1$ drzew utworzonych przez kaskadowe odcięcia oraz drzewo o korzeniu x) i co najwyżej $m(H) - c + 2$ zaznaczonych węzłów ($c - 1$ węzłów przestało być węzłami zaznaczonymi podczas kaskadowych odcięć, a ostatnie wywołanie procedury CASCADING-CUT mogło



Rys. 20.4. Dwa wywołania procedury FIB-HEAP-DECREASE-KEY. **(a)** Początkowy kopiec Fibonacciego. **(b)** Klucz 46 zostaje zmniejszony do wartości 15. Zawierający go węzeł staje się korzeniem, a jego ojciec (z kluczem 24), który poprzednio był niezaznaczony, zostaje zaznaczony. **(c)–(e)** Klucz 35 zostaje zmniejszony do wartości 5. W części (c) węzeł zawierający teraz klucz 5 staje się korzeniem. Jego ojciec, z kluczem 26, jest zaznaczony, więc następuje kaskadowe odcięcie. Węzeł z kluczem 26 zostaje odcięty od swojego ojca i staje się niezaznaczonym korzeniem w części (d). Ponieważ węzeł z kluczem 24 jest również zaznaczony, następuje kolejne kaskadowe odcięcie. W części (e) rysunku węzeł ten zostaje odcięty od swojego ojca i staje się niezaznaczonym korzeniem. Kaskadowe odcinanie zatrzymuje się w tym punkcie, ponieważ węzeł z kluczem 7 jest korzeniem. (Nawet gdyby nie był, seria odcięć skończyłaby się, ponieważ węzeł ten jest niezaznaczony). Część (e) przedstawia otrzymany w wyniku operacji FIB-HEAP-DECREASE-KEY kopiec Fibonacciego, w którym wskaźnik $min[H]$ wskazuje na nowy węzeł minimalny.

spowodować zaznaczenie węzła). Zmiana potencjału nie przekracza zatem

$$((t(H) + c) + 2(m(H) - c + 2)) - (t(H) + 2m(H)) = 4 - c.$$

Zamortyzowany koszt operacji FIB-HEAP-DECREASE-KEY wynosi więc co najwyżej

$$O(c) + 4 - c = O(1),$$

ponieważ możemy tak przeskalować jednostki potencjału, by zdominowały stałą ukrytą w wyrażeniu $O(c)$.

Widać teraz, dlaczego w definicji funkcji potencjału wystąpił składnik równy podwojonej liczbie zaznaczonych węzłów. Kiedy zaznaczony węzeł y zostaje odcięty w trakcie kaskadowego odcinania, jego polu *mark* zostaje nadana wartość

FALSE, potencjał zmniejsza się więc o 2. Jedna jednostka potencjału pokrywa koszt odcięcia i wyzerowania bitu zaznaczenia, a druga równoważy jednostkowy wzrost potencjału związany z tym, że węzeł y zostaje korzeniem.

Usuwanie węzła

Nietrudno usunąć dowolny węzeł z n-węzłowego kopca Fibonacciego w zamortyzowanym czasie $O(D(n))$. Realizuje to poniższa procedura. Zakładamy, że w kopcu Fibonacciego nie ma aktualnie klucza o wartości $-\infty$.

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

Procedura FIB-HEAP-DELETE jest podobna do BINOMIAL-HEAP-DELETE. Robi ona z x węzeł minimalny, nadając mu mniejszą niż wszystkie pozostałe wartość klucza $-\infty$. Węzeł x zostaje następnie usunięty z kopca Fibonacciego za pomocą procedury FIB-HEAP-EXTRACT-MIN. Zamortyzowany koszt operacji FIB-HEAP-DELETE jest sumą wynoszącego $O(1)$ zamortyzowanego kosztu FIB-HEAP-DECREASE-KEY i wynoszącego $O(D(n))$ zamortyzowanego kosztu FIB-HEAP-EXTRACT-MIN. Ponieważ w podrozdz. 20.4 przekonamy się, że $D(n) = O(\lg n)$, koszt zamortyzowany operacji FIB-HEAP-DELETE wynosi $O(\lg n)$.

ZADANIA

20.3-1. Przypuśćmy, że korzeń x w kopcu Fibonacciego jest zaznaczony. Wyjaśnij, jak mogło do tego dojść. Uzasadnij, że z punktu widzenia analizy fakt, iż x jest zaznaczony, nie ma znaczenia, chociaż nie jest on węzłem, który był wcześniej dołączony do innego, a później stracił jednego syna.

20.3-2. Uzasadnij oszacowanie $O(1)$ zamortyzowanego kosztu operacji FIB-HEAP-DECREASE-KEY jako średni koszt przypadający na jedną operację, stosując metodę kosztu sumarycznego.

20.4. Oszacowanie maksymalnego stopnia

Aby udowodnić, że zamortyzowany koszt operacji FIB-HEAP-EXTRACT-MIN i FIB-HEAP-DELETE wynosi $O(\lg n)$, musimy wykazać, że górne ograniczenie $D(n)$ na stopień dowolnego węzła w n-węzłowym kopcu Fibonacciego jest równe $O(\lg n)$. W myśl zadania 20.2-3, jeśli wszystkie drzewa w kopcu Fibonacciego są nieuporządkowanymi drzewami dwumianowymi, to $D(n) = \lfloor \lg n \rfloor$.