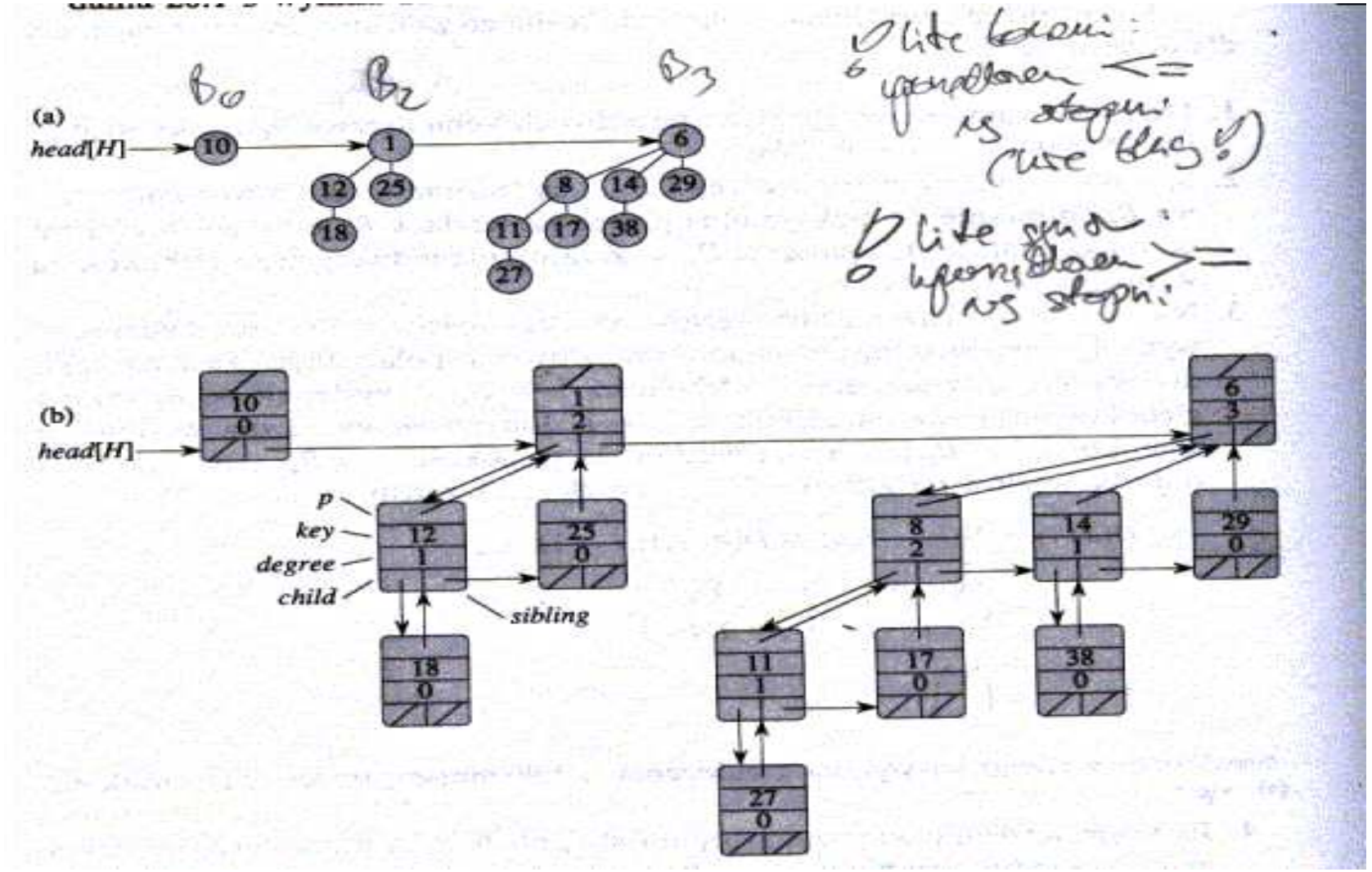
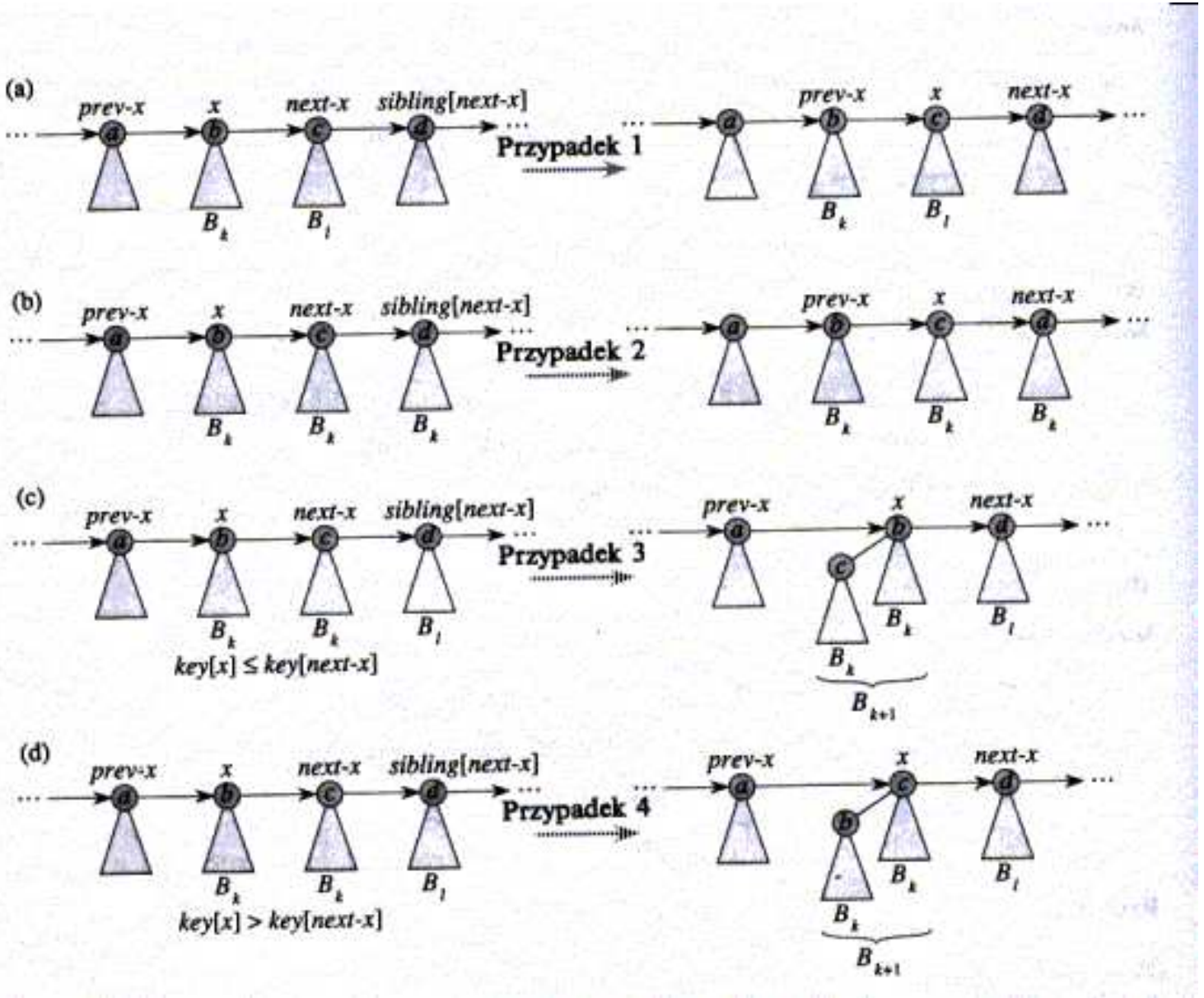
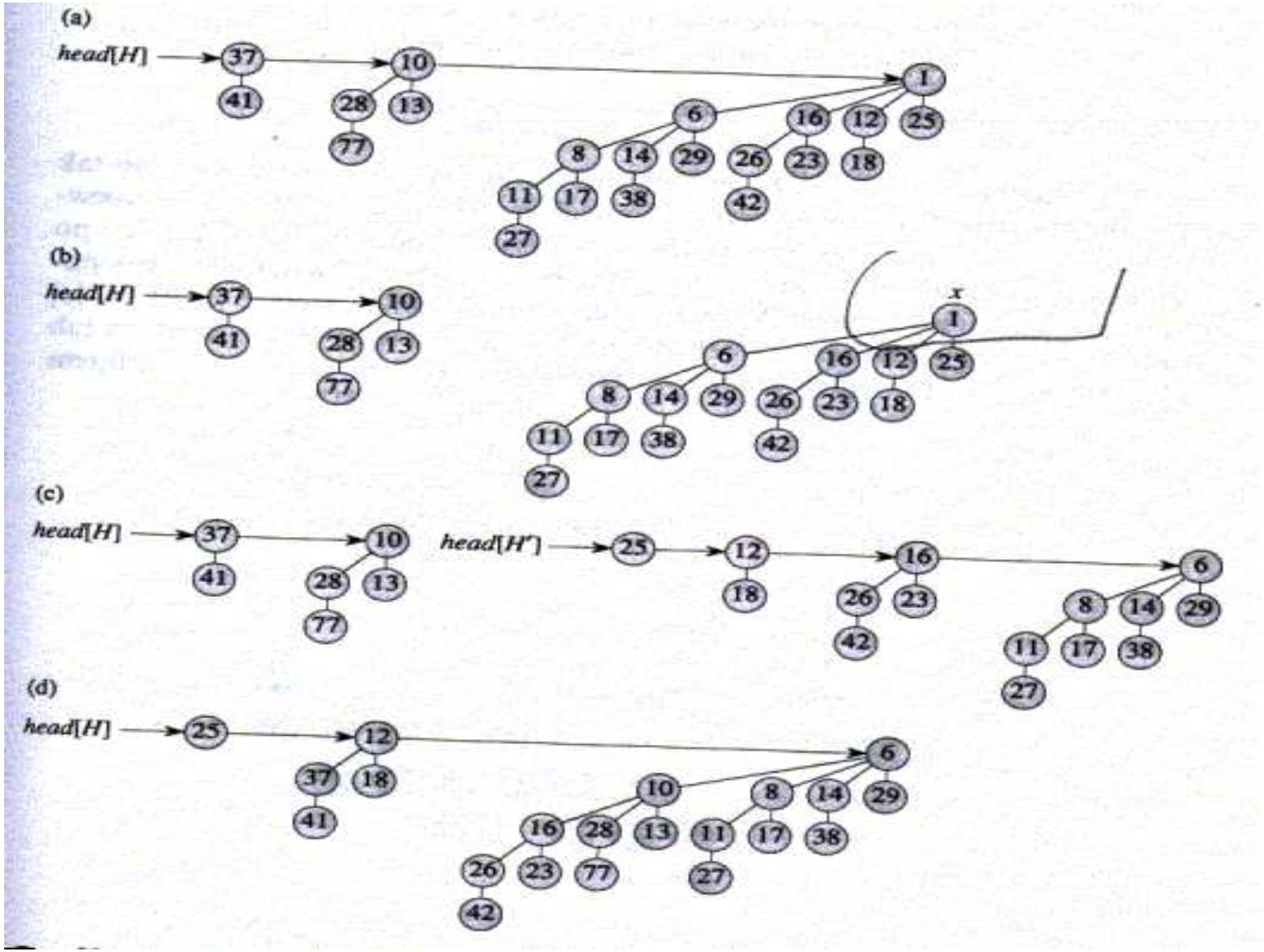
# Kopiec dwumianowy.



```
BINOMIAL-HEAP-UNION(H1, H2)
 1  H ← MAKE-BINOMIAL-HEAP()
 2  head[H] ← BINOMIAL-HEAP-MERGE(H1, H2)
 3  zwolnij obiekty H1 i H2, ale nie listy na które wskazują
 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]) lub
              (sibling[next-x] ≠ NIL
              i degree[sibling[next-x]] = degree[x])
11           then prev-x ← x                          ▷ Przypadki 1 i 2
12                x ← next-x                          ▷ Przypadki 1 i 2
13           else if key[x] ⩽ key[next-x]
14                then sibling[x] ← sibling[next-x]   ▷ Przypadek 3
15                     BINOMIAL-LINK(next-x, x)       ▷ Przypadek 3
16                else if prev-x = NIL                ▷ Przypadek 4
17                     then head[H] ← next-x          ▷ Przypadek 4
18                     else sibling[prev-x] ← next-x  ▷ Przypadek 4
19                     BINOMIAL-LINK(x, next-x)       ▷ Przypadek 4
20                     x ← next-x                     ▷ Przypadek 4
21           next-x ← sibling[x]
22  return H
```

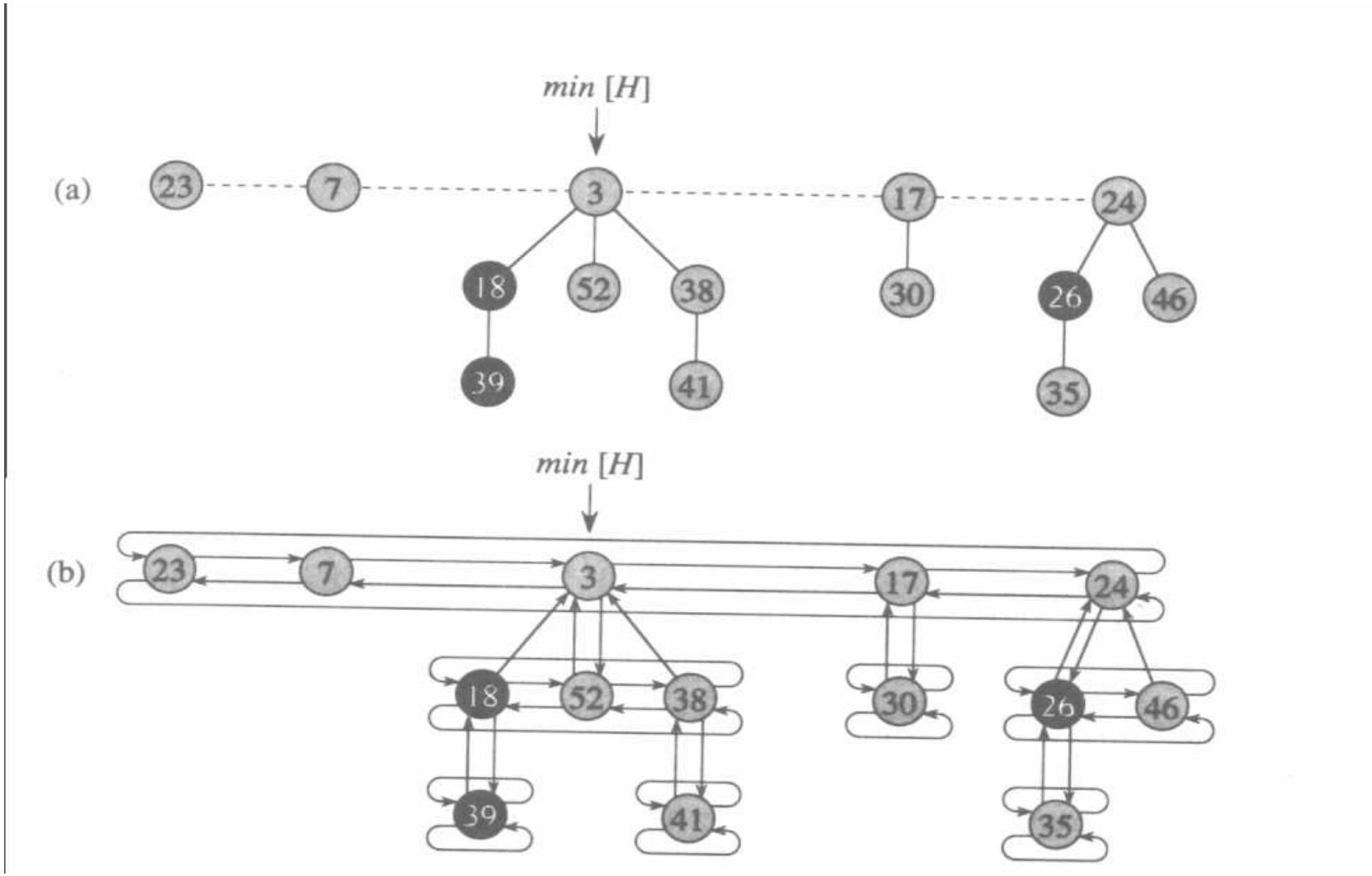

(a)
prev-x
x
next-x
sibling[next-x]
Przypadek 1
(b)
Przypadek 2
(c)
Przypadek 3
key[x] ≤ key[next-x]
(d)
Przypadek 4
key[x] > key[next-x]

# ExtractMin

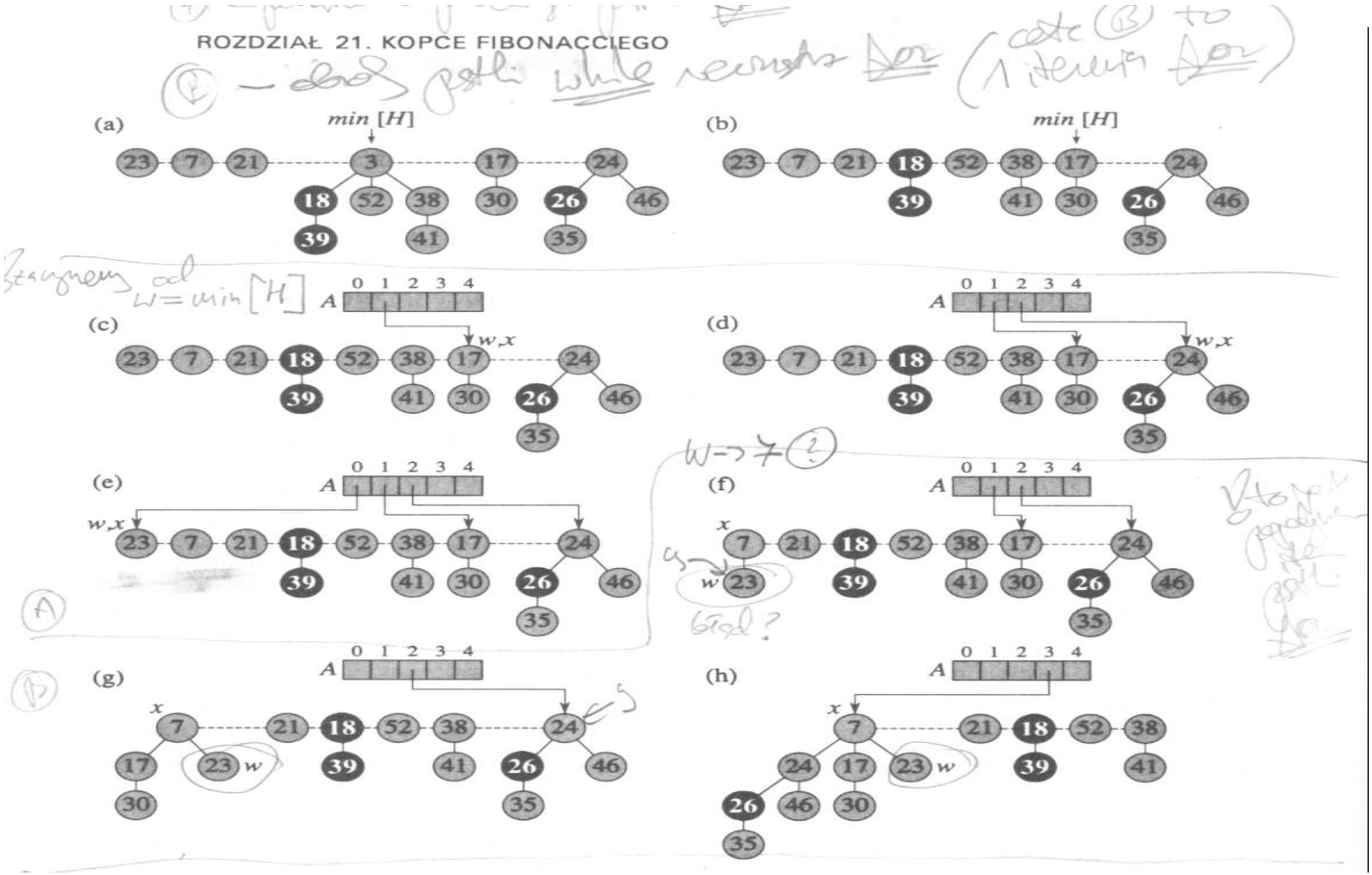

(a)
head[H]
37
41
10
28
13
77
1
6
16
12
25
8
14
29
26
23
18
11
17
38
42
27
(b)
head[H]
37
41
10
28
13
77
x
1
6
16
12
25
8
14
29
26
23
18
11
17
38
42
27
(c)
head[H]
37
41
10
28
13
77
head[H']
25
12
18
16
26
23
42
6
8
14
29
11
17
38
27
(d)
head[H]
25
12
37
18
41
6
10
8
14
29
16
28
13
11
17
38
26
23
77
27
42

# Kopiec Fibonacciego.

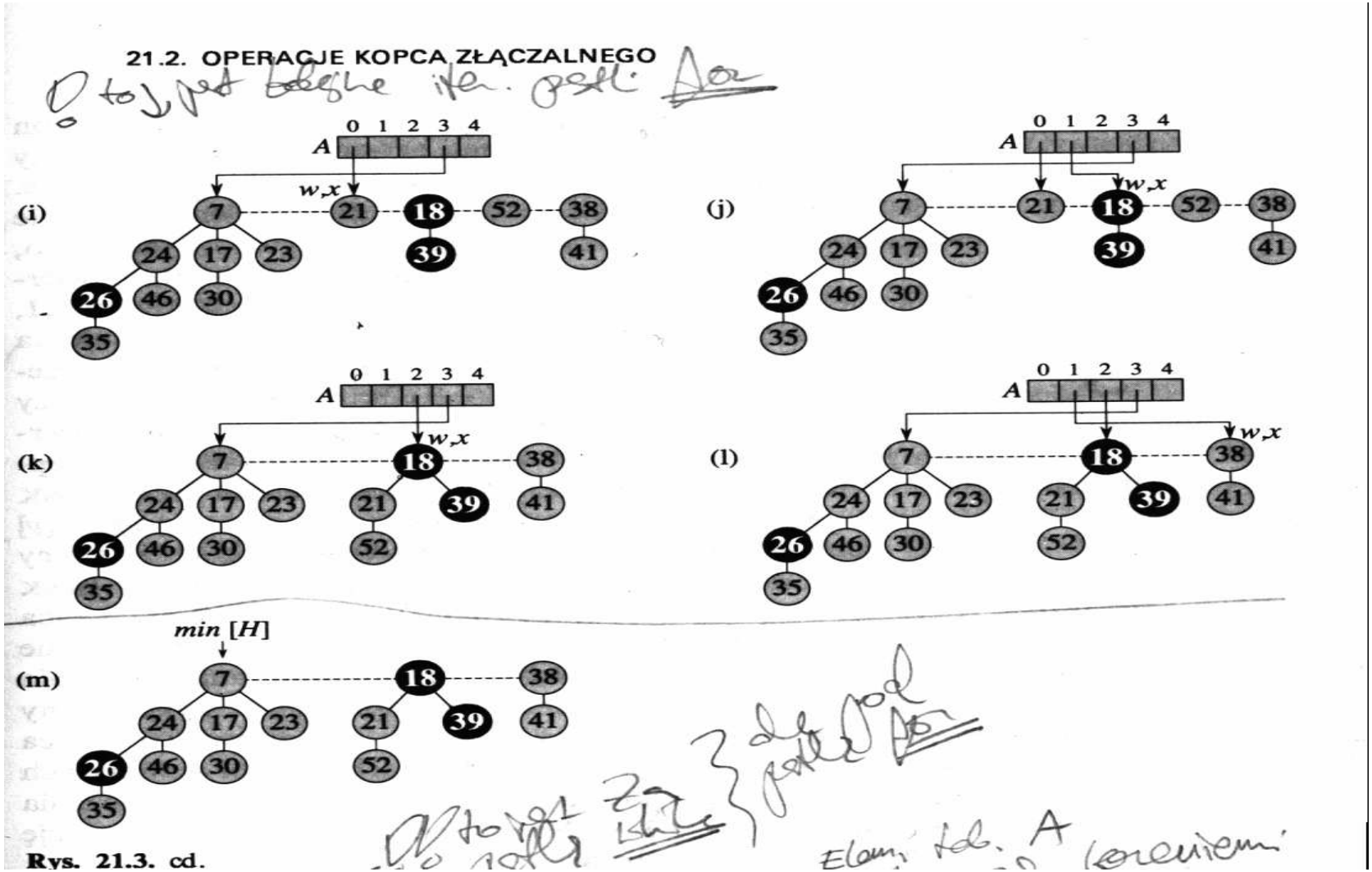
Reprezentacja kopca Fibonacciego:



Operacja ExtractMin:

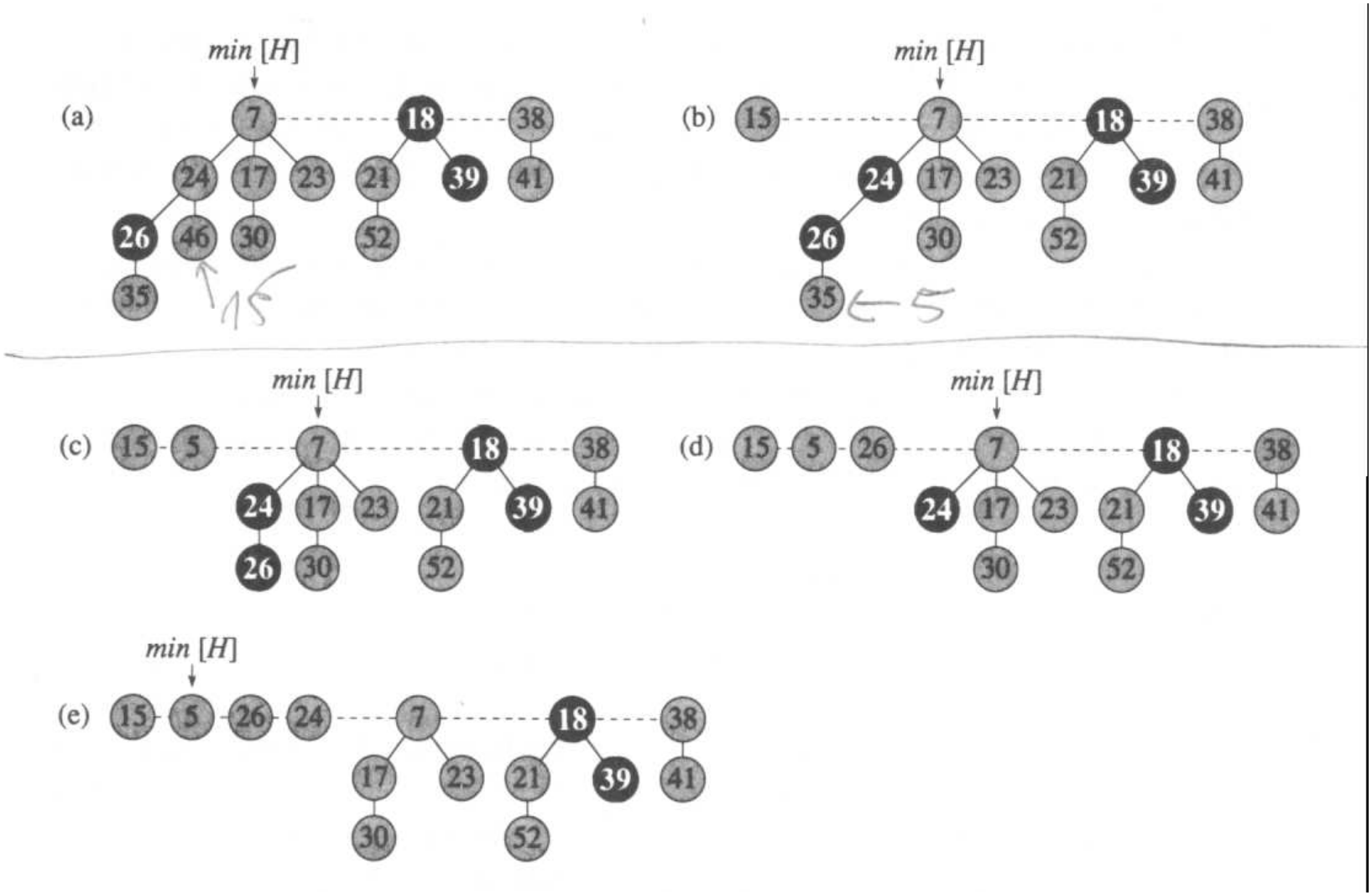


Operacja ExtractMin c.d.:

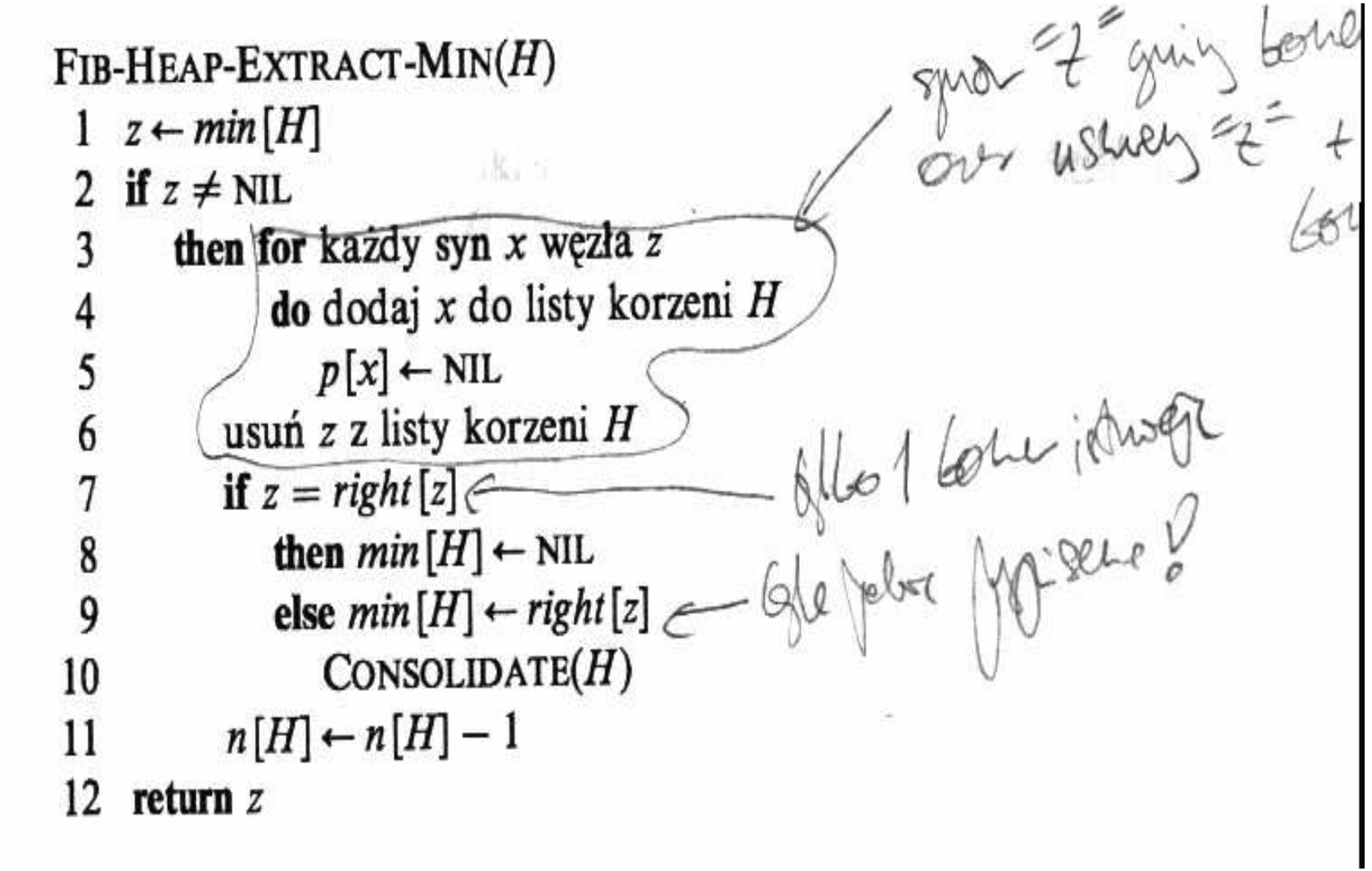



Rys. 21.3. cd.

Operacja DecreaseKey:

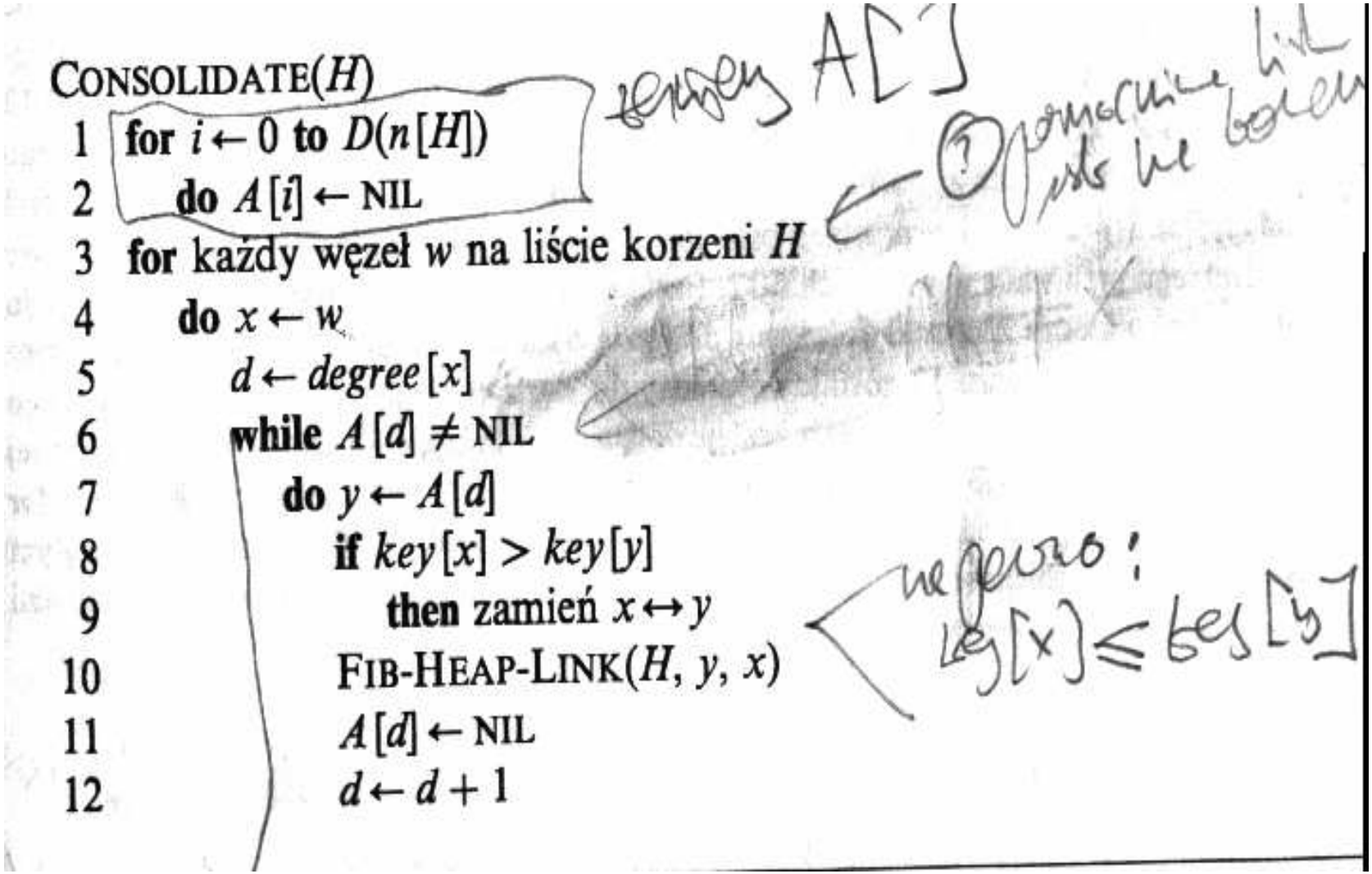


Kod operacji ExtractMin:



```
FIB-HEAP-EXTRACT-MIN(H)
 1  z ← min[H]
 2  if z ≠ NIL
 3     then for każdy syn x węzła z
 4             do dodaj x do listy korzeni H
 5                p[x] ← NIL
 6          usuń z z listy korzeni H
 7          if z = right[z]
 8             then min[H] ← NIL
 9             else min[H] ← right[z]
10                  CONSOLIDATE(H)
11          n[H] ← n[H] − 1
12  return z
```

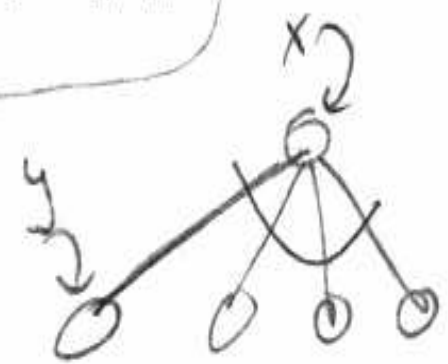
Kod operacji ExtractMin c.d.:



```
CONSOLIDATE(H)
 1  for i ← 0 to D(n[H])
 2      do A[i] ← NIL
 3  for każdy węzeł w na liście korzeni H
 4      do x ← w
 5         d ← degree[x]
 6         while A[d] ≠ NIL
 7             do y ← A[d]
 8                if key[x] > key[y]
 9                   then zamień x ↔ y
10                FIB-HEAP-LINK(H, y, x)
11                A[d] ← NIL
12                d ← d + 1
```

Kod operacji ExtractMin c.d.:

```
13        A[d] ← x
14    min[H] ← NIL
15    for i ← 0 to D(n[H])
16        do if A[i] ≠ NIL
17           then dodaj A[i] do listy korzeni H
18                if min[H] = NIL lub key[A[i]] < key[min[H]]
19                   then min[H] ← A[i]
```

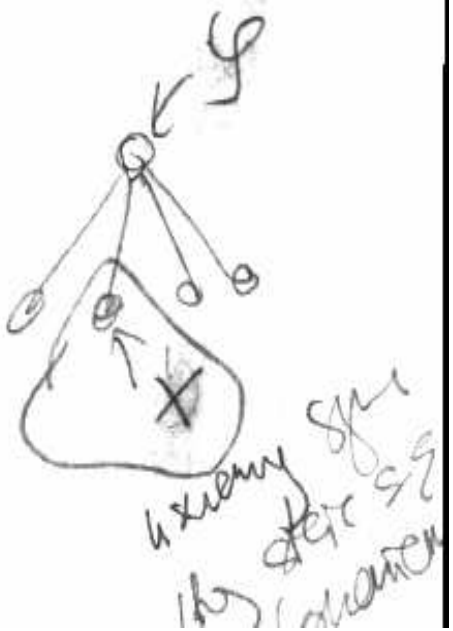



```
FIB-HEAP-LINK(H, y, x)
1  usuń y z listy korzeni H
2  uczyń y synem x i zwiększ degree[x] o 1
3  mark[y] ← FALSE
```

Kod operacji DecreaseKey:

```
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 H
```



Kod operacji DecreaseKey c.d.:

```
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)
```