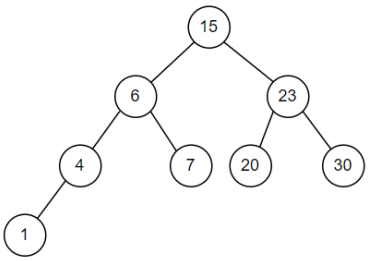
Bagian 1 (45 menit)

1. Di antara kebutuhan berikut ini, manakah yang cocok diimplementasikan menggunakan tree?
   a. Pengaturan dokumen (file system) dalam beberapa folder yang memungkinkan ada hirarki.
   b. Penyimpanan data mahasiswa yang memungkinkan pengelompokan berdasarkan angkatannya.
   c. Penyelesaian masalah Tower of Hanoi.
   d. Penyelesaian masalah Fibonacci.

2. Berapakah tinggi maksimum sebuah tree dengan N buah nodes?
   a. N-1
   b. N
   c. Log N
   d. Fibonacci(N+3) - 1

3. Perhatikan tree berikut ini.



   Tuliskan hasil traversal tree tersebut dengan menggunakan algoritma post order traversal (dengan urutan pemrosesan anak dari kiri ke kanan). Tuliskan node dipisahkan dengan karakter spasi.
   Contoh jawaban: 1 2 3 4

   Jawaban: 1 4 7 6 20 30 23 15

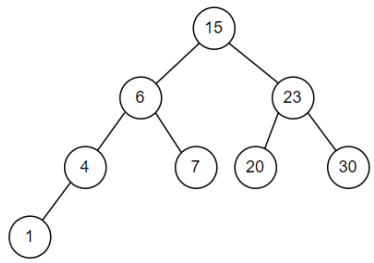
4. Dalam operasi remove data pada BST dengan menggunakan kaidah successor in-order, diketahui node yang di-remove memiliki 2 anak. Berikut pernyataan yang benar mengenai operasi remove pada BST menggunakan kaidah tersebut
   a. Node pengganti dapat merupakan leaf node, atau internal node yang hanya memiliki anak kanan
   b. Node pengganti nilainya lebih kecil daripada node yang dihapus
   c. Node pengganti dapat merupakan node yang memiliki 2 anak.
   d. Node pengganti merupakan internal node yang memiliki anak kiri

5. Perhatikan potongan program berikut.

```
static int result = 0;
static void mystery(Node root){
    if (root == null)
        return;
    if (root.left == null && root.right == null)
            result += root.data;

    mystery(root.left);
    mystery(root.right);
}
```

Apabila method mystery dipanggil dengan parameter root dari tree berikut, berapa nilai variable result setelah pemanggilan berakhir?



x result += leaf

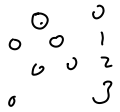
Jawaban: 58

6. Kompleksitas operasi insert, delete, dan search pada Binary Search Tree saat mencapai kasus terburuk adalah
    a. O(N) untuk semua operasi
    b. O(N) untuk insert dan delete, O(log N) untuk search
    c. O(log N) untuk semua operasi
    d. O(N log N) untuk insert dan delete, O(log N) untuk search

7. Manakah pernyataan yang benar mengenai AVL Tree?
    a. Selisih tinggi subtree kiri dan kanan setiap node tidak akan lebih dari 1
    b. Setelah operasi deletion sebuah data pasti memerlukan proses rebalancing tree
    c. AVL Tree merupakan BST yang setiap internal node-nya memiliki 2 children
    d. Setelah insertion, setiap eksternal node diperiksa apakah masih balanced atau tidak

8. Operasi pencarian data pada struktur data AVL Tree kompleksitas waktunya sama dengan pencarian secara optimal pada struktur data
    a. Array terurut
    b. Linked list
    c. Hash table
    d. Queue

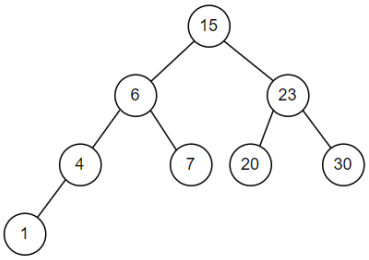
9. Maksimum tinggi AVL-Tree yang memiliki 6 buah node adalah

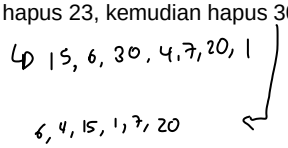


   a. 2
   b. 3
   c. 1
   d. 4

10. Semua operasi berikut ini pada AVL-Tree bersifat logaritmis (O(Log N), dengan N adalah banyaknya node) KECUALI: (Note: struktur AVL-tree yang standar)
   a. Mendapatkan elemen leaf dengan depth terbesar dari suatu AVL-Tree.
   b. Mendapatkan elemen leaf yang terbesar dalam AVL-Tree.
   c. Mendapatkan elemen predecessor in-order dari elemen pada root AVL-Tree.
   d. Mendapatkan node dengan nilai maksimum atau minimum dari suatu branch AVL-Tree.

11. Pada AVL-Tree berikut ini dilakukan operasi: hapus 23, kemudian hapus 30.

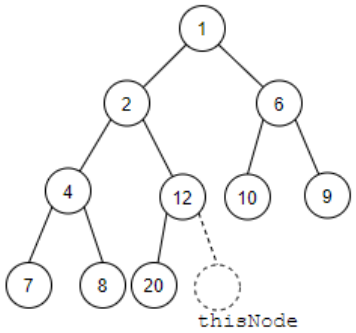




    Tuliskan level order dari AVL-Tree setelah operasi-operasi tersebut termasuk rebalancing yang terjadi. Penghapusan menerapkan kaidah *successor in-order*. Tuliskan jawaban dalam satu baris deretan bilangan pada node dipisahkan oleh tand spasi, contoh: 4 1 6 15 20 7

    (**Kunci jawaban**: 6 4 15 1 7 20)

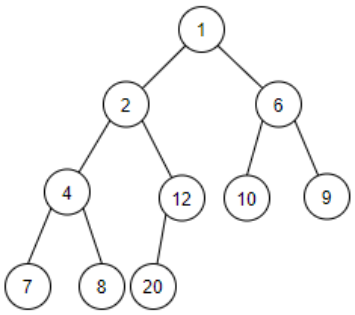
12. Perhatikan binary heap yang direpresentasikan dengan tree sebagai berikut:



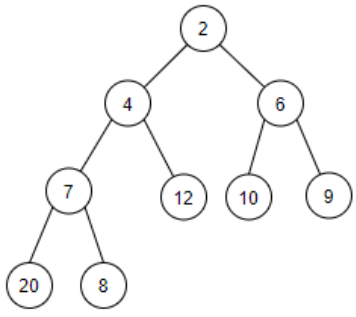
Jika dilakukan operasi insert data 11, maka setelah operasi tersebut selesai, node dengan label thisNode akan berisi data:

a. 12
b. 11
c. 20
d. 1

13. Perhatikan binary heap yang direpresentasikan dengan tree sebagai berikut:

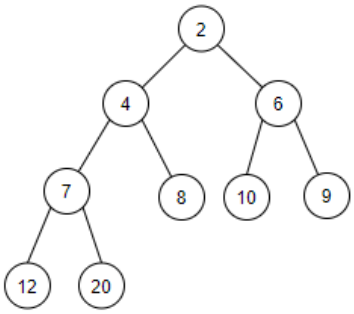


Jika dilakukan operasi delete minimum, maka binary heap akan menjadi:

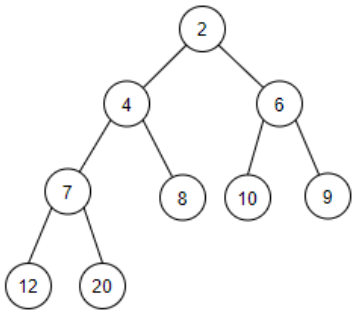


a

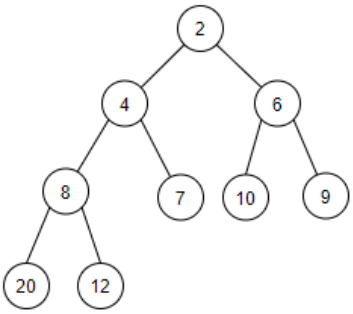
b.



c.



d.

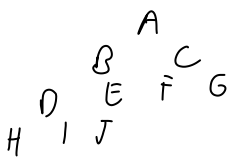


14. Perhatikan binary heap yang direpresentasikan dengan array sebagai berikut:

Indeks:	0	1	2	3	4	5	6	7	8	9
Isi Array:	A	B	C	D	E	F	G	H	I	J

Siapakah parent dari data J?

a. E
b. D

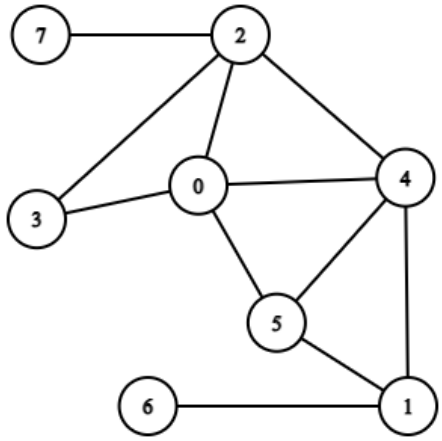


c. F
d. G

15. Operasi manakah yang TIDAK UMUM dilakukan pada sebuah Binary Heap?
    a. Menghapus elemen yang terakhir ditambahkan
    b. Mendapatkan elemen minimum
    c. Menghapus elemen minimum
    d. Menambahkan elemen baru

16. Struktur data manakah yang tepat diimplementasikan dengan Binary Heap?
    a. Minimum Priority Queue, Maximum Priority Queue
    b. Minimum Priority Queue, Stack
    c. Maximum Priority Queue, Stack
    d. Queue, Stack

17. Perhatikan graf berikut.



Tuliskan urutan traversal graf tersebut dimulai dari verteks 0 dengan menggunakan algoritma:

a. Breadth First Search (BFS) 0, 2, 3, 4, 5, 7, 1, 6
b. Depth First Search (DFS) 0, 2, 7, 4, 1, 5, 6, 7

catatan:

- *Jika terdapat lebih dari satu tetangga yang dapat dikunjungi, verteks dengan angka lebih kecil akan dikunjungi terlebih dahulu.*
- *Tulis nama verteksnya dengan dipisahkan tanda spasi, contoh 0 5 6.*

Jawaban:

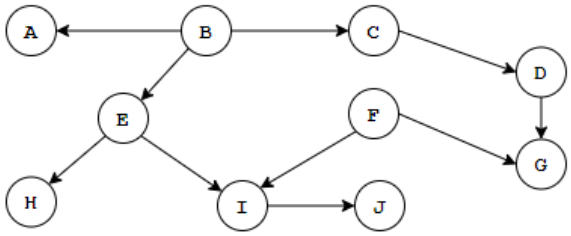
a. 0 2 3 4 5 7 1 6
b. 0 2 3 4 1 5 6 7

18. Pernyataan berikut yang **SALAH** mengenai struktur data graf adalah:
    a. Dengan Algoritma Prim's, kita dapat mengetahui jarak terpendek graf berbobot
    b. Algoritma DFS dapat diimplementasikan dengan pendekatan rekursif
    c. Topological sorting tidak dapat dilakukan pada graf yang memiliki cycle
    d. Algoritma BFS dapat digunakan untuk mencari jarak terpendek pada graf tidak berbobot

19. Sebuah graf tidak berarah dengan N buah verteks dan M buah edges direpresentasikan dengan matriks. Operasi printAllAdjacent(Verteks v) akan mencetak semua verteks yang adjacent dengan verteks v. Berapakah kompleksitas dari operasi tersebut?
    a. O (N)
    b. O (M)
    c. O (NM)
    d. O (N + M)

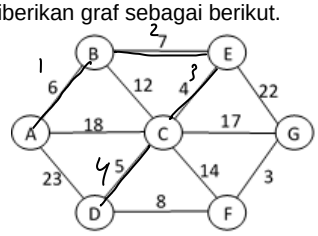
20. Hal-hal yang pasti benar pada penggunaan Algoritma Dijkstra dalam pencarian path terpendek adalah:
    a. Hanya digunakan untuk non-negative weighted graph.
    b. Hanya digunakan untuk weighted graph.
    c. Bukan untuk menemukan shortest-path antara pasangan verteks.
    d. Hanya digunakan untuk undirected graph.

21. Diberikan graf berarah sebagai berikut:



Node mana saja yang mungkin menjadi node ke-1 pada topological sort-nya?
    a. B, F
    b. F, J
    c. G, H, J
    d. B, G

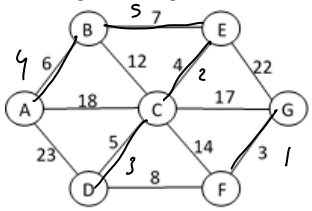
22. Diberikan graf sebagai berikut.



Jika algoritma Prim digunakan untuk mendapatkan MST pada graph tersebut. Jika algoritma dimulai dari A, urutan berapakah sisi CD diambil?

a. 4
b. 3
c. 2
d. Tidak diambil karena membentuk siklik.

23. Diberikan graf sebagai berikut.



Jika algoritma Kruskal digunakan untuk mendapatkan MST pada graph tersebut. Urutan berapakah sisi CD diambil?

a. 3
b. 2
c. 4
d. Tidak diambil karena membentuk siklik

24. Query manakah yang dapat dijalankan lebih efisien (kompleksitas waktu paling baik) pada representasi **adjacency matrix** dibandingkan representasi lain (adjacency list dan edge-list) jika M adalah banyaknya sisi dan N banyaknya verteks, serta M proporsional dan mendekati N^2.
    a. Memeriksa apakah verteks ke i dan verteks ke j bersifat adjacent secara simetris (dari ke i ke j dan sebaliknya).
    b. Memeriksa apakah verteks i reachable dari verteks j.
    c. Mengetahui sisi mana dalam graph yang memiliki bobot terbesar.
    d. Mengetahui out-degree dari suatu verteks.

25. Masalah secondary clustering dapat terjadi pada Hash Table yang mengimplementasikan collision resolution berikut (Catatan: jawaban bisa lebih dari satu. Opsi jawaban yang salah bernilai minus):
    a. **Linear probing**
    b. **Quadratic probing**
    c. Double hashing
    d. Open Hashing

26. Diberikan sebuah Hash Table yang menggunakan Linear Probing. Pada awalnya, berikut ini adalah isi dari Hash Table:

Indeks	0	1	2	3	4	5	6	7	8	9
Isi Cell			A	B	D	C				

Lalu, elemen D akan ditambahkan ke dalam Hash Table. Nilai hash dari elemen tersebut adalah 2. Maka elemen D akan ditempatkan pada cell di indeks ...

a. 4
b. 6
c. 7
d. 8

27. Diberikan sebuah Hash Table yang menggunakan Quadratic Probing. Pada awalnya, berikut ini adalah isi dari Hash Table:

Indeks	0	1	2	3	4	5	6	7	8	9
Isi Cell			A	B		C	D			

Lalu, elemen D akan ditambahkan ke dalam Hash Table. Nilai hash dari elemen tersebut adalah 2. Maka elemen D akan ditempatkan pada cell di indeks ...

a. 6
b. 4
c. 7
d. 9

28. Diberikan sebuah Hash Table yang menggunakan Double Hashing. Pada awalnya, berikut ini adalah isi dari Hash Table:

Indeks	0	1	2	3	4	5	6	7	8	9	10
Isi Cell			A	B		C		D			

Lalu, elemen E akan ditambahkan ke dalam Hash Table. Nilai Hash1 dari elemen tersebut adalah 2, dan nilai Hash2-nya adalah 5. Maka elemen E akan ditempatkan pada cell di indeks ...

a. 1
b. 0
c. 8
d. 9

29. Sebuah Hash Table diimplementasikan dengan Open Hashing, setiap cell memiliki hanya satu reference ke linked list masing-masing (singly linked list), yaitu header yang menunjuk ke node pertama pada linked list. Manakah implementasi yang tepat supaya operasi insert selalu memiliki kompleksitas O(1)?

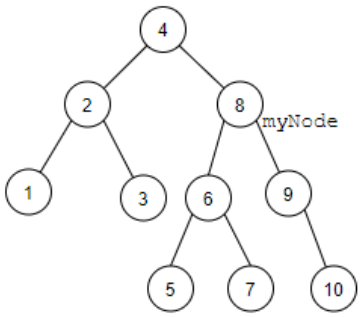
a. Menambahkan elemen baru di posisi header
b. Menambahkan elemen baru di posisi sebelum node terakhir
c. Menambahkan elemen baru di tengah linked list
d. Menambahkan elemen baru di posisi setelah node terakhir

Bagian 2 (menit)

1. Class Node memiliki 3 buah variabel: int data, Node left, dan Node right. Perhatikan method berikut ini:

```
1  public static int mystery(Node nodeX){
2      int count = 0;
3      if(nodeX != null){
4          if(nodeX.data % 2 == 0)
5              count += 1;
6          count += mystery(nodeX.left);
7          count += mystery(nodeX.right);
8      }
9      return count;
10 }
```

   a. Perhatikan myNode yaitu node yang ditunjuk pada gambar, tentukan berapakah return value dari pemanggilan mystery(myNode)? Jelaskan juga tracing-nya.



   b. Apa yang dilakukan oleh method mystery(Node nodeX) secara umum?

Kunci Jawaban:

a. 3.

Tracing:

→ mystery(node 8):
8 genap maka count = 1
count +=
→ left: mystery(node 6):
6 genap maka count = 1
count+=
→ left: mystery(node 5)
5 ganjil maka count = 0
count +=
→ left: mystery(5.left): base case
← return 0
count = 0 + 0 = 0
→ right: mystery(5.right): base case

← return 0
count = 0 + 0 = 0
← return 0
count = 1 + 0 = 1
count +=
→ right: mystery(node 7)
7 ganjil maka count = 0
count +=
→ left: mystery(null): base case
← return 0
count = 0
count +=
→ right: mystery(null): base case
← return 0
count = 0 + 0 = 0
← return 0
count = 1 + 0 = 1
← return 1
count = 1 + 1 = 2
count +=
→ right: mystery(node 9)
9 ganjil maka count = 0
count +=
→ left: mystery(null): base case return 0
count = 0
count +=
→ right: mystery(node 10)
10 genap maka count = 1
count +=
→ left: mystery(null): base case
← return 0
count = 1 + 0 = 1
count +=
→ right: mystery(null): base case
← return 0
count = 1 + 0 = 1
← return 1
count = 0 + 1 = 1
← return 1
count = 2 + 1 = 3
← return 3

b. Method tersebut akan menghitung berapa banyak node yang berisi data genap pada subtree dengan root nodeX.

2. Sebuah AVL Tree memiliki tinggi 8.
   a. Berapakah minimum banyaknya node dalam AVL Tree tersebut? Jelaskan.
   b. Berapakah maksimum banyaknya node dalam AVL Tree tersebut? Jelaskan.

Kunci Jawaban:

a. Banyaknya minimum node:

Bil. fibonacci:

i:	0	1	2	3	4	5	6	7	8	9	10	11
Fib(i):	0	1	1	2	3	5	8	13	21	34	55	89

Minimum nodes:

Tinggi:	0	1	2	3	4	5	6	7	8
Minimum nodes:	1	2	4	7	13-1	21-1	34-1	55-1=54	89-1=88

Jadi, minimum jumlah nodes-nya adalah 88 nodes.

Catatan: pola terlihat pada angka-angka yang di-highlight, yaitu
minimum nodes dari AVL tree dengan tinggi x = fib(x+3) - 1
Kemungkinan mahasiswa yang paham akan langsung pakai formula tersebut.

b. Banyaknya maksimum node adalah ketika tree-nya full (terisi sampai level terakhir, yaitu level ke-8, dengan root adalah di level 0), yaitu:

$2^{banyakLevel} - 1$
$= 2^9 - 1$
$= 2^6 . 2^3 - 1$
$= 64 . 8 - 1$
$= 512 - 1$
$= 511$ nodes

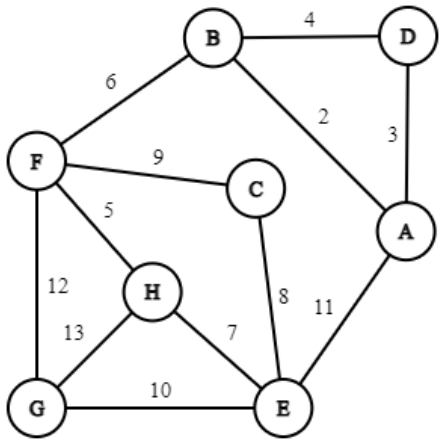
3. Lakukan fix-heap (heapify) dari data dalam array sebagai berikut (ada 12 data): 32, 4, 90, 78, 10, 70, 6, 33, 43, 48, 3, 19. Pada setiap step percolate-down, tuliskan isi array yang merepresentasikan bentuk heap tree.

(Catatan: dalam iterasi porcelate-down yang dilakukan sebanyak jumlah internal node, cetak tebal elemen array yang berubah isinya).

Jawaban:

32, 4, 90, 78, 10, **19**, 6, 33, 43, 48, 3, **70**
32, 4, 90, 78, **3**, 19, 6, 33, 43, 48, **10**, 70
32, 4, 90, **33**, 3, 19, 6, **78**, 43, 48, 10, 70
32, 4, **6**, 33, 3, 19, **90**, 78, 43, 48, 10, 70
32, **3**, 6, 33, **4**, 19, 90, 78, 43, 48, 10, 70
3, **4**, 6, 33, **10**, 19, 90, 78, 43, 48, **32**, 70

4. Perhatikan graf berikut.

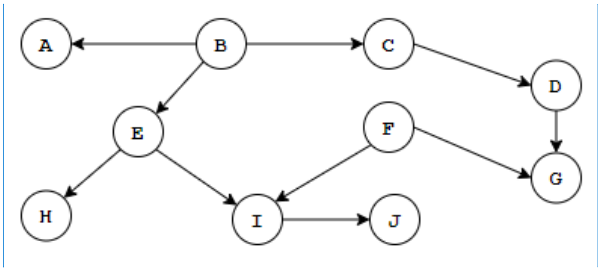


   a. Terapkan algoritma Kruskal untuk membuat Minimum Spanning Tree. Jawab dengan menuliskan edge apa saja yang diperiksa, dengan format sebagai berikut: Edge - Bobot - Status
   Contoh jawaban:
   AZ - 10 - X
   PQ - 11 - X
   Status X artinya …
   (Jelaskan status apa saja yang Anda pakai)

   b. Kapan algoritma Kruskal berhenti pada kasus graf tersebut? Jelaskan.

   Jawaban:

5. Diberikan graf berarah sebagai berikut:



   Topological sort untuk graf tersebut tidak unik. Dalam soal ini, verteks B dipilih menjadi verteks pertama, dan verteks J yang dipilih menjadi verteks terakhir pada topological sort-nya. Lalu ditulis dalam format sekuensial seperti pada tabel berikut:

Urutan ke:	1	2	3	4	5	6	7	8	9	10
Verteks:	B									J

a. Verteks mana sajakah yang mungkin menempati urutan ke-2 (sesuai yang tertulis pada tabel)? Jelaskan.
b. Verteks mana sajakah yang mungkin menempati urutan ke-9 (sesuai yang tertulis pada tabel)? Jelaskan.

Jawaban:

a. A, C, E, F. Penjelasan:

Setelah B diambil dan outgoing edges-nya dihapus, maka verteks A, C, E memiliki in-degree nol, sehingga bisa menjadi verteks ke-2. Verteks F juga bisa saja menjadi verteks ke-2 karena sejak semula memang in-degree-nya nol.

b. A, G, H, I. Penjelasan:

Jika J adalah verteks terakhir yang diambil pada topological sort, verteks sebelum J pada topological sort pasti tidak memiliki outgoing edge ke selain J, yaitu verteks A, G, H, I.

6. Diberikan sebuah hashtable yang bisa menampung 11 elemen. Hashtable ini menggunakan prinsip double hashing, di mana jika k adalah key, i menunjukkan sudah berapa collision yang terjadi, m adalah ukuran hashtable, dan fungsi lengkap untuk mencari posisi sebuah key di dalam hashtable tersebut adalah sebagai berikut.

   h(k,i) = (h1(k) + i*h2(k)) mod m

   Fungsi h1 dan h2 untuk setiap data dapat dilihat pada tabel berikut:

k	17	10	7	26	107	23	97	16
h1(k)	7	0	7	6	7	3	7	6
h2(k)	3	5	3	4	3	2	3	4

   Saat ini tabel sudah berisi 17, 10, dan 7. Lakukanlah operasi-operasi berikut ini secara berurutan dan tunjukkanlah isi dari hashtable tersebut.

   (1) sisip berturut-turut 26, 107, 23
   (2) hapus berturut-turut 26, 7 (menggunakan konsep lazy deletion)
   (3) sisip berturut-turut 97, 16

Kondisi awal:

indeks	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
berisi	10		107	7	23	97	26	17		

Tuliskan indeks untuk data berikutnya (jika tidak mengalami collision, collision ke 0):

Data 26 ditempatkan di . . . . 6 . . ., setelah mengalami collision ke . . . 0 . . .
Data 107 ditempatkan di . . . 2 . . ., setelah mengalami collision ke . . . 2 . . . .
Data 23 ditempatkan di . . . . 4 . . ., setelah mengalami collision ke . . . 6 . . . .
Data 97 ditempatkan di . . . . 5 . . ., setelah mengalami collision ke . . . 3 . . . .
Data 16 ditempatkan di . . . . 9 . . ., setelah mengalami collision ke . . . 4 . . . .