BAB 5

LINKED LIST

LINKED LIST ATAU ONE-WAY LIST

Adalah koleksi linier dari elemen data yang disebut Simpul atau Node.

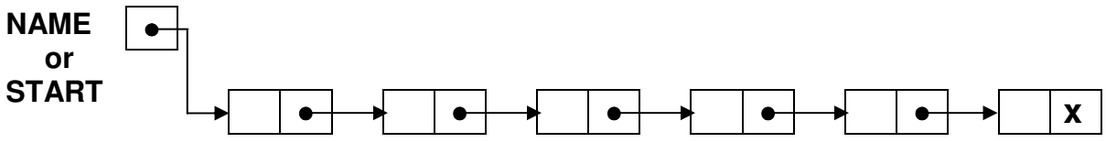
Cara melinierkan urutan adalah dengan menggunakan Penuding atau Pointer.

Setiap simpul terdiri atas dua bagian yaitu :
1. Berisi informasi data
2. Merupakan field link atau nextpointer.

Link menghubungkan satu elemen data ke elemen data lainnya, sehingga urutan elemen data tersebut membentuk suatu linier list.

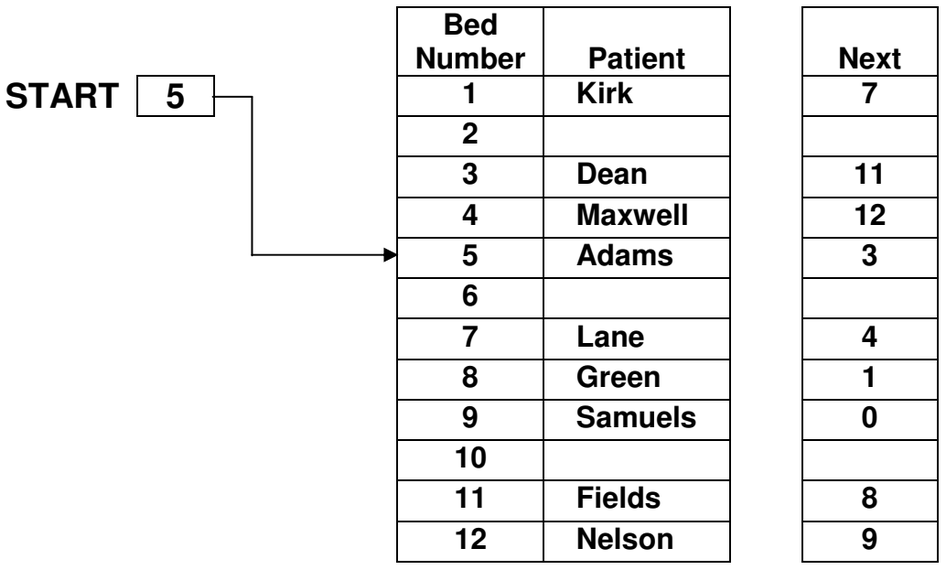
Link akan bernilai = 0 bila tidak menuding ke data (simpul) lainnya. Penuding ini disebut Penuding Nol.

Gambar list dengan 6 simpul



Gambar 1

Contoh :
Pada bangsal sebuah rumah sakit terdapat 12 tempat tidur. Sembilan di antaranya telah ditempati Pasien. Kita hendak membuat list nama para pasien tersebut secara alfabetik.



Bed Number	Patient	Next
1	Kirk	7
2		
3	Dean	11
4	Maxwell	12
5	Adams	3
6		
7	Lane	4
8	Green	1
9	Samuels	0
10		
11	Fields	8
12	Nelson	9

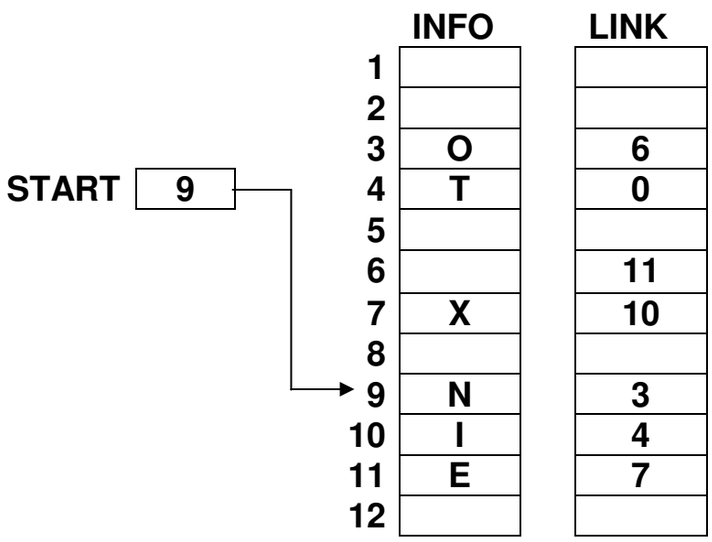
Gambar 2

PENYAJIAN LINKED LIST DALAM MEMORI

Disajikan dalam bentuk sebagai berikut :
1. Array INFO(K) : menyajikan bagian informasi
2. Array LINK(K) : field nextpointer
3. Variabel START : untuk menyimpan alamat dari elemen LIST
   Pada bagian akhir dari LIST, nextpointer bernilai NULL.

Contoh 1:

Menggambarkan suatu linked list dalam memori.
Data dalam Array INFO(K) adalah sebuah karakter tunggal.



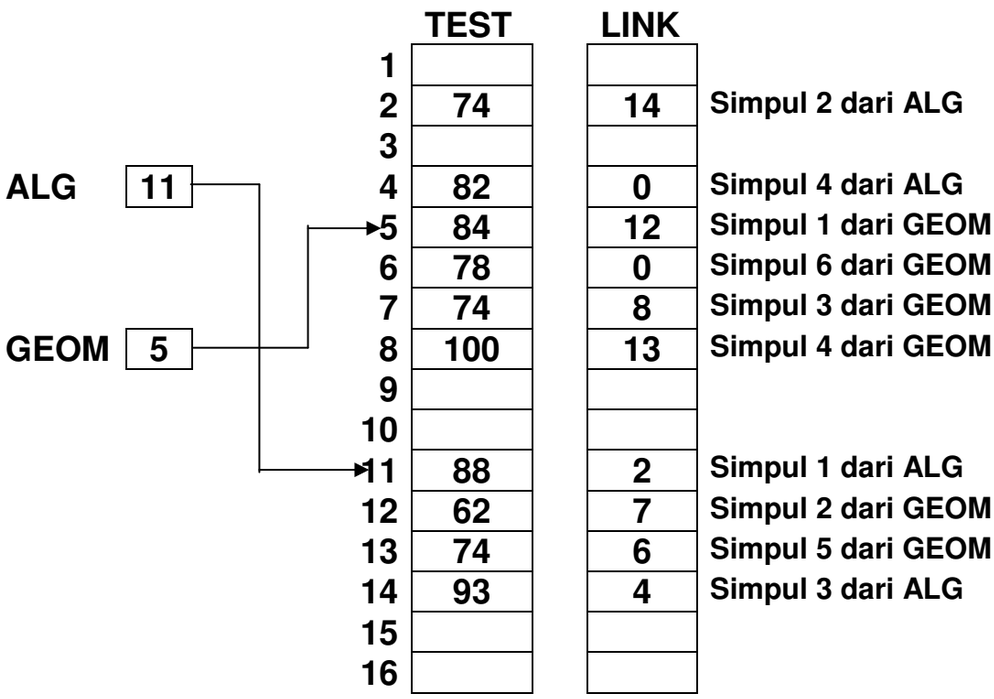
Gambar 3

String yang dinyatakan dari contoh tersebut adalah NO EXIT.

Di sini :

START	=	9	INFO[9]	=	N, elemen pertama adalah N
LINK(9)	=	3	INFO[3]	=	O, elemen kedua adalah O
LINK(3)	=	6	INFO[6]	=	Blank, elemen ketiga adalah blank
LINK(6)	=	11	INFO[11]	=	E, elemen keempat adalah E
LINK(11)	=	7	INFO[7]	=	X, elemen kelima adalah X
LINK(7)	=	10	INFO(10]	=	I, elemen keenam adalah I
LINK(10)	=	4	INFO[4]	=	T, elemen ketujuh adalah T
LINK(4)	=	0	Null, list terakhir		

Contoh 2 :
Memperlihatkan dua buah linked list, ALG dan GEOM, yang berturut-turut berisi nilai matakuliah Algoritma dan Geometri, dapat tersimpan bersama dalam array TEST dan LINK yang sama.
Pointer ALG berisi nilai 11, yakni lokasi dari simpul pertama list ALG,
Pointer GEOM berisi nilai 5, yakni lokasi simpul pertama dari list GEOM.

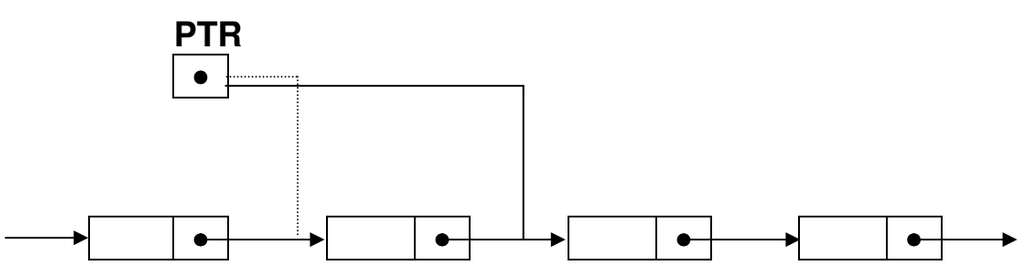


Gambar 4

Mengikuti pointer tersebut, maka :
Nilai dari ALG berturut-turut adalah : 88, 74, 93, 82
Nilai dari GEOM berturut-turut adalah : 84, 62, 74, 100, 74, 78

KUNJUNGAN LINKED LIST

Traversal atau kunjungan simpul list sesuai urutan untuk memproses setiap simpul tepat satu kali.
Algoritma traversal, menggunakan variabel penuding PTR untuk menuding simpul yang sedang di proses saat ini. Karena itu LINK(PTR) akan menuding simpul berikut dalam list.
Statement PTR := LINK(PTR) akan menggerakkan penuding ke simpul berikutnya.



Gambar 5

Algoritma Traversal secara rinci :
Mula-mula, kita awali dengan memberi nilai kepada PTR, sama dengan START. Kita proses INFO(PTR), yakni informasi pada simpul pertama dalam List. Selanjutnya PTR diperbaharui melalui statement PTR := LINK(PTR). Sekarang proses INFO(PTR), yakni informasi pada simpul kedua. Demikian seterusnya sampai nilai PTR = NULL, akhir dari traversal.

ALGORITMA
1. PTR := START.
2. Kerjakan Langkah 3 dan 4 dalam hal PTR <> NULL :
3. Proses INFO(PTR).
4. PTR := LINK(PTR).
5. EXIT.

CARI (SEARCHING) DALAM LINKED LIST
1. Cari dalam list tidak terurut
2. Cari dalam list terurut

CARI DALAM LIST TIDAK TERURUT

Melakukan Traversal Simpul list, sambil setiap kali memeriksa apakah informasi dalam simpul yang tengah dikunjungi tersebut sama dengan ITEM.

Dalam algoritma ini diperlukan 2 buah pemeriksaan pada setiap putaran yaitu :
1. Memeriksa apakah telah sampai pada akhir dari list, yakni dengan memeriksa apakah PTR = NULL.
2. Memeriksa apakah ITEM telah ditemukan lokasinya, yakni dengan memeriksa apakah INFO(PTR) = ITEM

ALGORITMA

SEARCH(INFO, LINK, START, ITEM, LOC)
1. PTR := START.
2. Kerjakan langkah 3 dalam hal PTR <> NULL :
3. Jika INFO(PTR) = ITEM, maka :
   LOC := PTR, exit.
   Bila tidak
   PTR := LINK(PTR).
4. LOC := NULL. (Pencarian gagal)
5. Exit.

CARI DALAM LIST TERURUT

Melakukan traversal simpul list, sambil setiap kali memeriksa apakah informasi dalam simpul yang tengah dikunjungi tersebut sama dengan ITEM. Karena terurutnya list, tidak perlu melakukan traversal sampai akhir dari list, walau ITEM tidak terdapat dalam list.
Begitu INFO(PTR) > ITEM, hentikan proses pencarian.
Dalam algoritma ini diperlukan 2 buah pemeriksaan pada setiap putaran yaitu :

1. Memeriksa apakah INFO(PTR) sudah lebih besar dari ITEM, berarti ITEM tidak terdapat dalam list, hentikan proses cari.
2. Memeriksa apakah ITEM telah ditemukan lokasinya, yakni dengan memeriksa apakah INFO(PTR) = ITEM.

ALGORITMA

SRCHSL(INFO, LINK, START, ITEM, LOC)
1. PTR := START.
2. Kerjakan langkah 3 dalam hal PTR <> NULL :
3. Jika INFO(PTR) < ITEM, maka :
   PTR := LINK(PTR).
   Bila tidak jika ITEM = INFO(PTR), maka :
   LOC := PTR, dan exit. (pencarian sukses)
   Bila tidak :
   LOC := NULL, and exit.
4. LOC := NULL.
5. Exit.

ALOKASI MEMORI : KOLEKSI SAMPAH

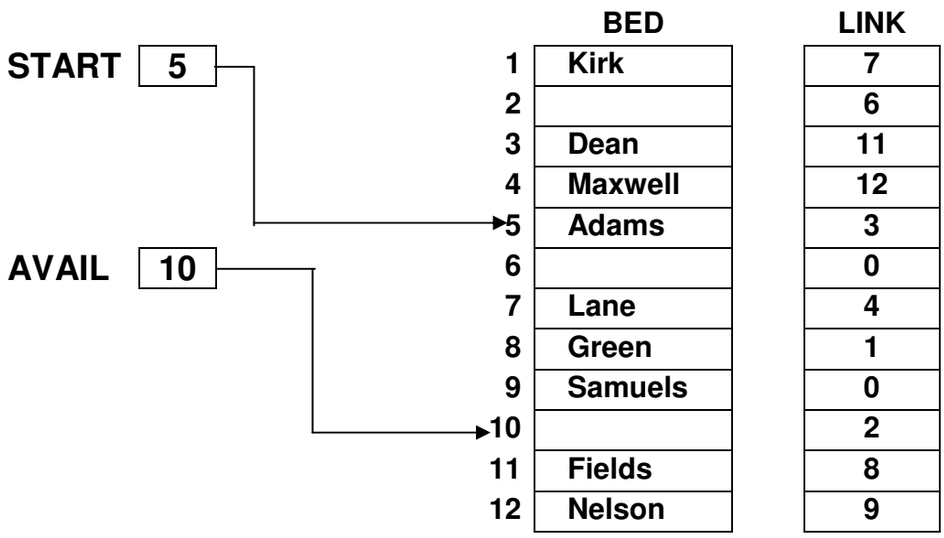
Ketika menyimpan linked list dalam memori, diasumsikan bahwa selalu dapat dilakukan penyisipan simpul baru ke dalam list, serta penghapusan simpul dari list.

Untuk itu diperlukan suatu mekanisme guna menyediakan memori bagi simpul baru, atau untuk mengelola memori yang sementara ini tidak berguna karena adanya penghapusan simpul, untuk sewaktu-waktu dapat dipakai lagi.

Sel memori dalam array yang tak digunakan, dihimpun menjadi sebuah linked list lain yang menggunakan variabel penuding list berupa array AVAIL, dituliskan sebagai :

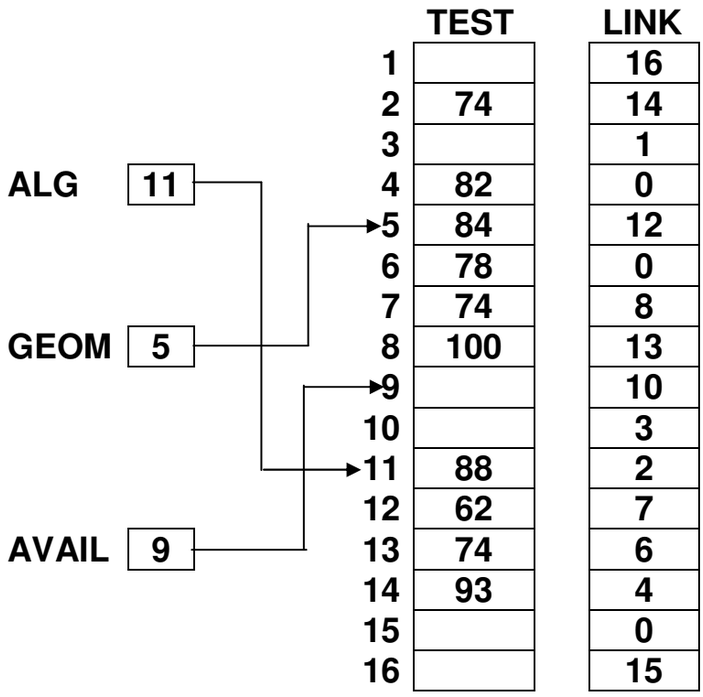
LIST(INFO, LINK, START, AVAIL)

Contoh :
Pandang Daftar Pasien pada contoh yang lalu. Daftar kita simpan dalam dua array BED dan LINK.
Pasien tempat tidur K dinyatakan sebagai BED(K). Maka ruang yang tersedia dalam array BED tersebut dapat dikaitkan seperti gambar di bawah ini.



Gambar 6

Contoh :
Perhatikan bahwa masing-masing list ALG dan GEOM boleh menggunakan list AVAIL. Dapat dicatat bahwa AVAIL = 9, maka TEST(9) adalah simpul bebas pertama dalam list AVAIL tersebut. Karena LINK(AVAIL) = LINK(9) = 10, maka TEST(10) adalah simpul bebas kedua dalam AVAIL, demikian seterusnya.



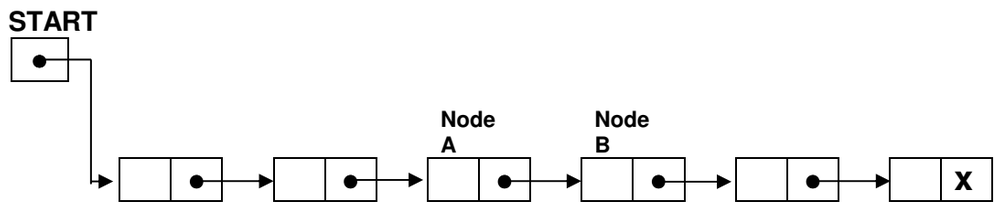
Gambar 7

KOLEKSI SAMPAH

Koleksi Sampah atau Garbage Collection adalah suatu metode dengan sistem operasi dapat secara periodik mengumpulkan semua ruang akibat penghapusan simpul list tersebut ke dalam list ruang bebas.

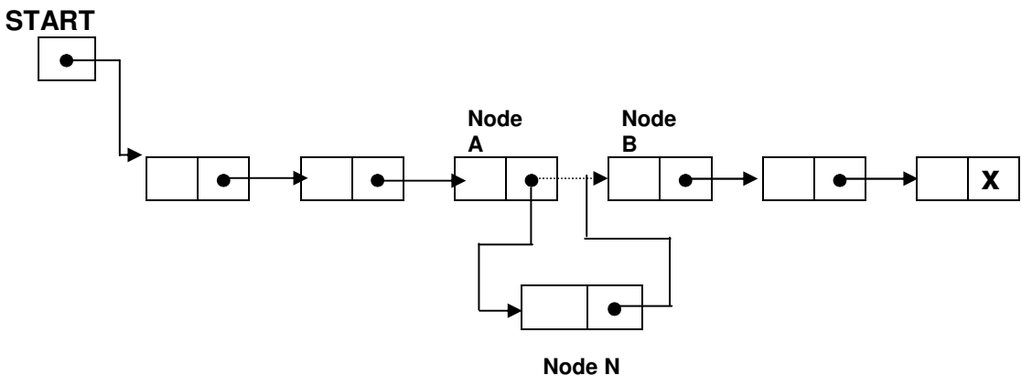
PENYISIPAN SIMPUL KE DALAM LINKED LIST

Misalkan LIST adalah linked list dengan A dan B adalah dua simpul yang berurutan.



Gambar 8a

Sisipkan simpul baru N ke dalam LIST, antara simpul A dan B.



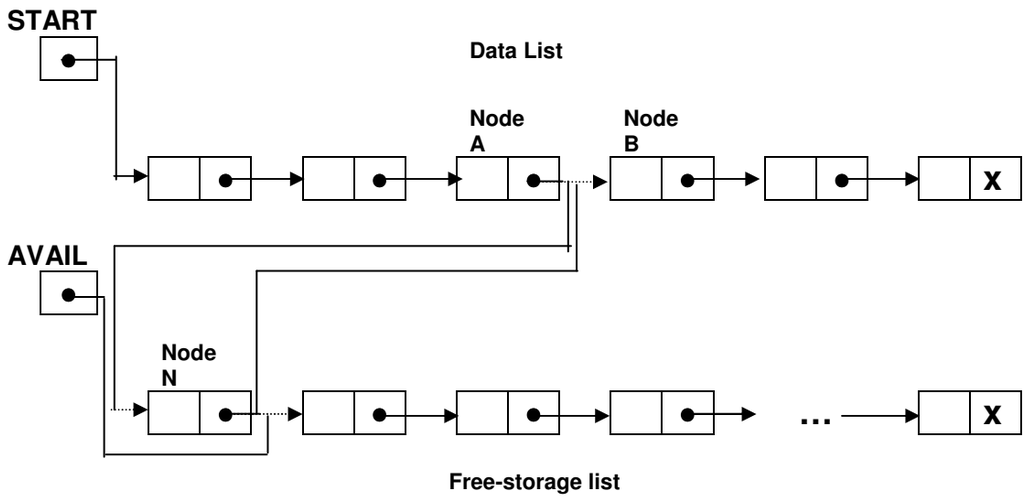
Gambar 8b

Di sini simpul A sekarang menuding ke simpul baru N, dan simpul N menuding ke simpul B, yang tadinya dituding oleh A.

Pandang bahwa linked list tersimpan di dalam memori dalam bentuk

LIST(INFO, LINK, START, AVAIL)

Gambar 8b, tidak dapat memperlihatkan bahwa simpul baru N memanfaatkan ruang memori dari list AVAIL. Untuk kemudahan proses, simpul pertama AVAIL dipakai untuk menyimpan simpul baru N tersebut.



Gambar 9

Perhatikan bahwa field Penuding berubah sebagai berikut :

1. Field nextpointer dari simpul A sekarang menuding ke simpul baru N, terhadap mana sebelumnya AVAIL menuding.
2. AVAIL sekarang menuding ke simpul kedua pada ruang bebas, terhadap mana sebelumnya simpul N menuding.
3. Field nextpointer dari simpul N, sekarang menuding ke simpul B, yang tadinya dituding oleh simpul A.

Di sini terdapat 2 kasus khusus yaitu :

1. Jika simpul baru N adalah simpul pertama dalam list, maka START akan menuding N.
2. Jika simpul baru N adalah simpul terakhir dalam list, maka N akan berisi penuding nol.

Contoh :

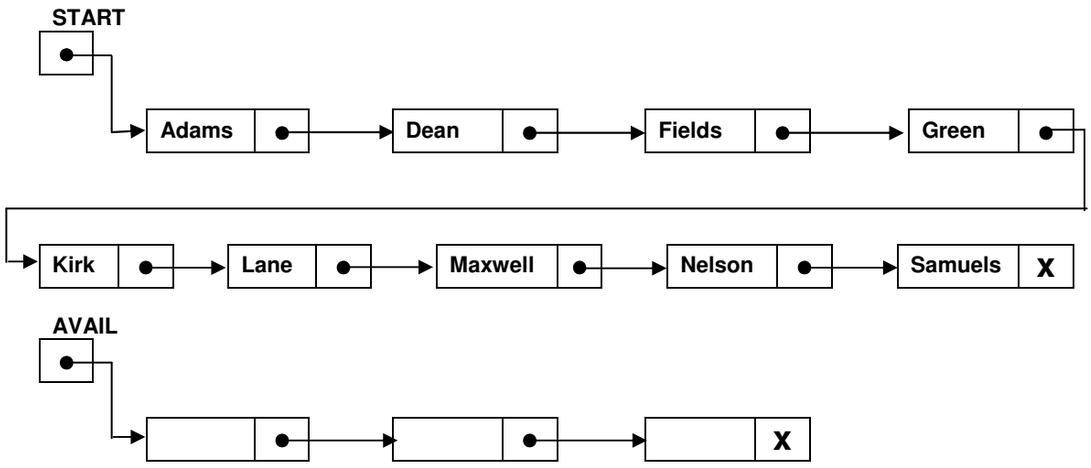
Pandang Daftar Alfabetik Pasien. Misalkan seorang pasien baru Hughes masuk.

Perhatikan bahwa :

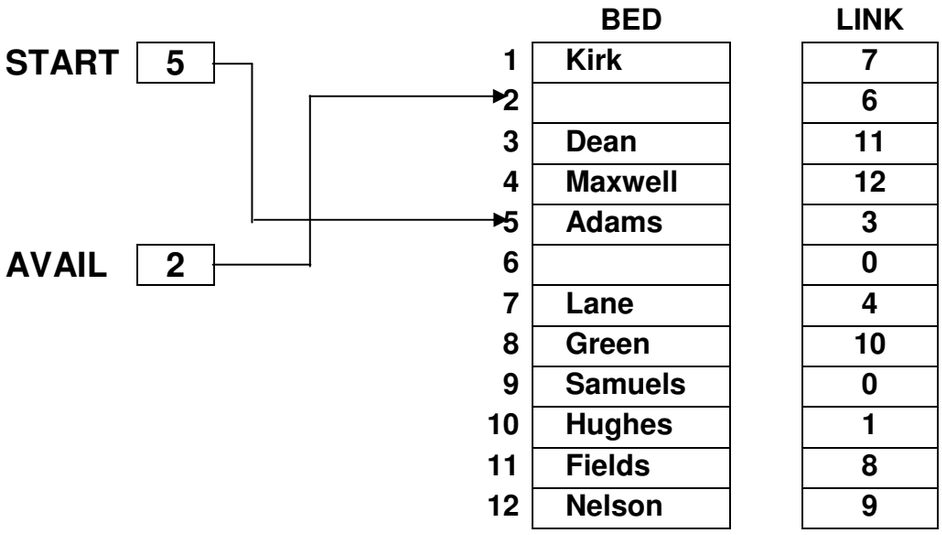
1. Hughes ditempatkan di ranjang 10, yakni ranjang pertama yang kosong (tersedia).
2. Hughes akan disisipkan dalam list, antara Green dan Kirk.

Terjadi 3 perubahan dalam Field Penuding sebagi berikut :

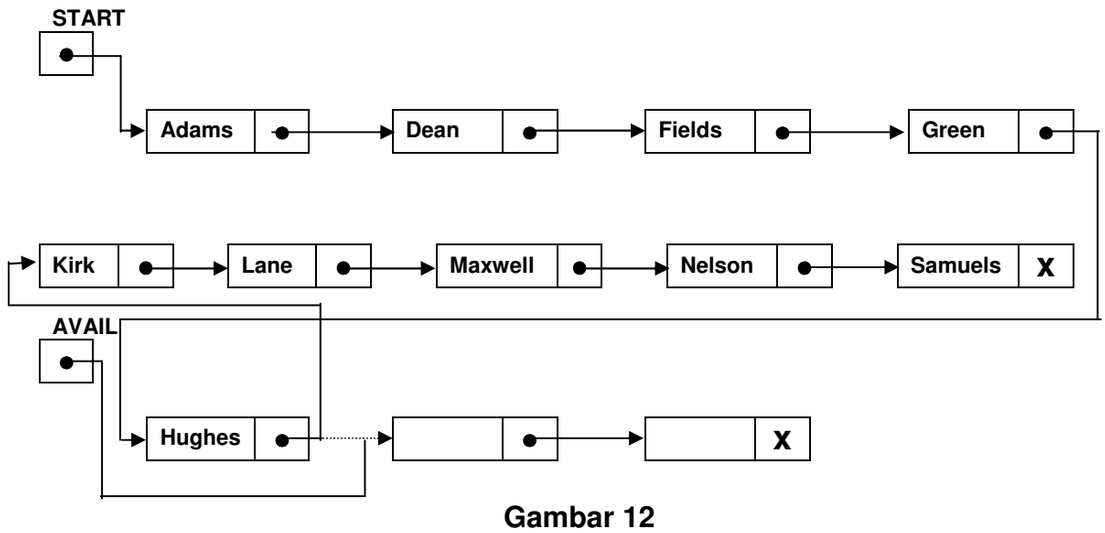
LINK(8) = 10 (sekarang Green menuding Hughes)
LINK(10) = 1 (sekarang Hughes menuding Kirk)
AVAIL = 2 (sekarang AVAIL menuding ranjang kosong kedua)



Gambar 10



Gambar 11

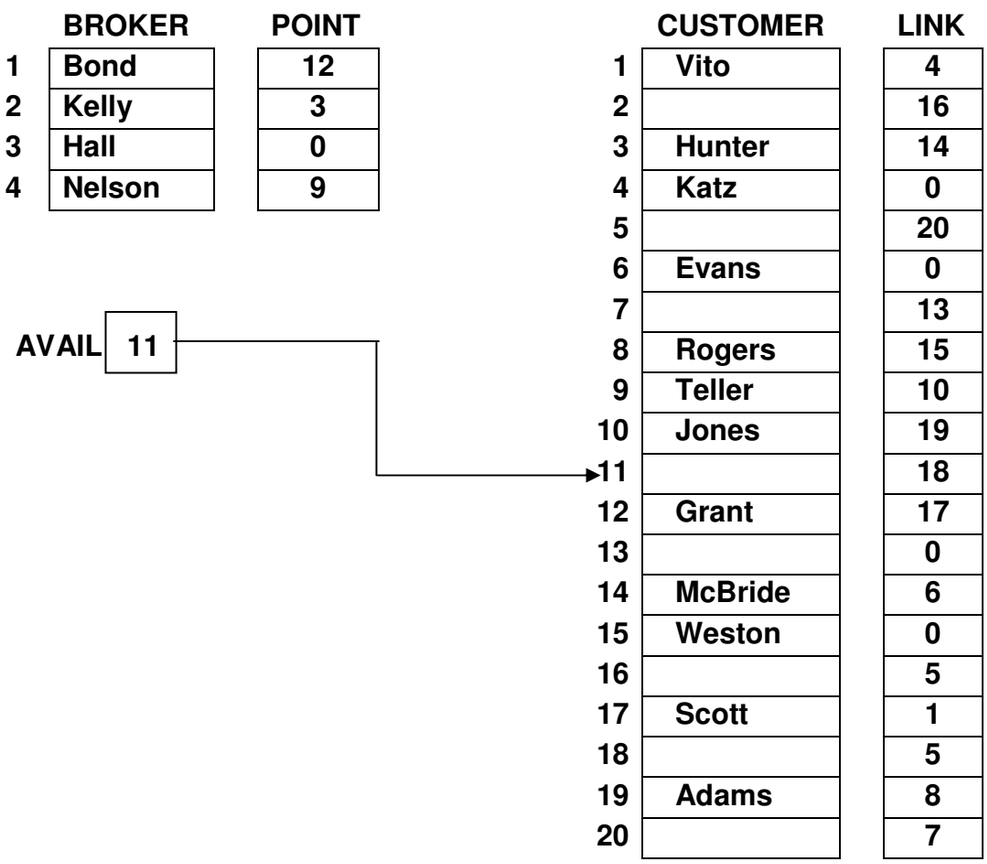


Gambar 12

Contoh :
Pandang suatu agen penjualan yang mempunyai 4 orang broker (perantara). Setiap broker mempunyai list customer (pelanggan) masing-masing.
Di sini keempat list pelanggan digabung dalam satu linked list dengan array CUSTOMER berisi nama pelanggan dan array LINK merupakan nextpointer.
Nama broker ditempatkan dalam array BROKER, dan digunakan pula variabel penuding dalam bentuk array POINT, sedemikian sehingga POINT(K) menuding ke lokasi dari simpul pertama BROKER(K).

Dapat kita lihat bahwa :

BROKER	PELANGGAN
Bond	Grant, Scott, Vito, Katz
Kelly	Hunter, McBride, Evans
Hall	-
Nelson	Teller, Jones, Adams, Rogers, Weston



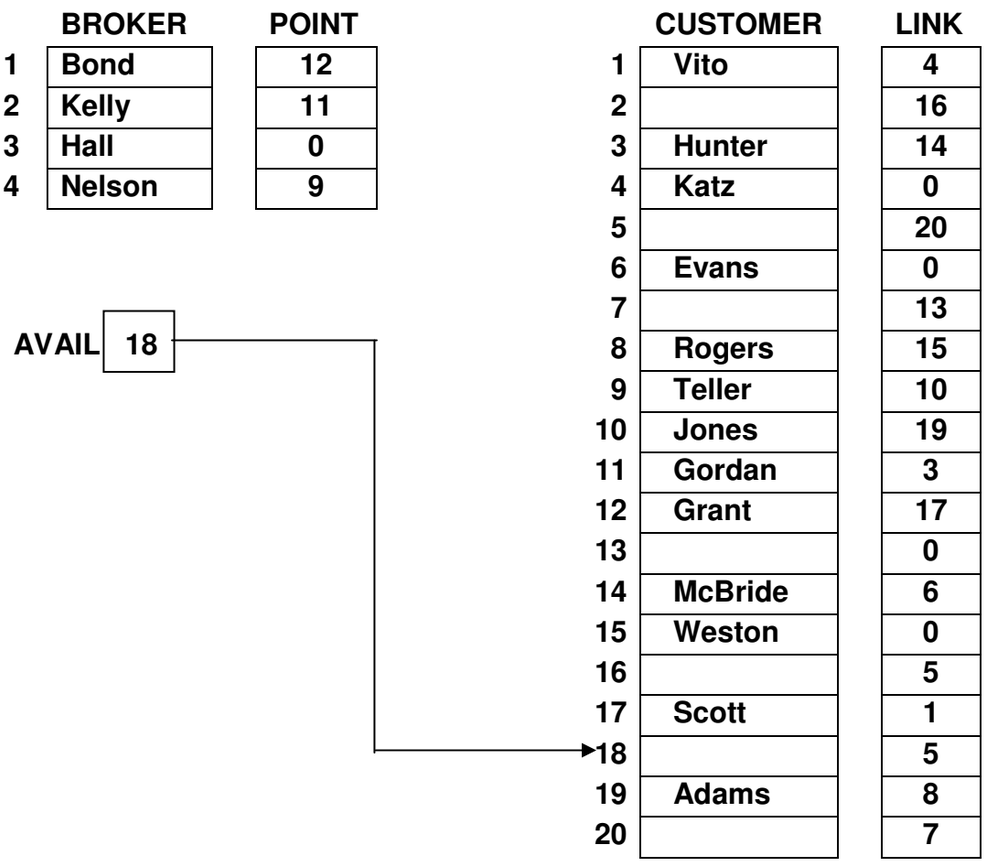
Gambar 13

Pandang daftar broker dan pelanggannya.
Karena list pelanggan tidak terurut, asumsikan bahwa setiap pelanggan baru ditambahkan pada awal list.

Misalkan Gordan adalah pelanggan baru dari Kelly.
Perhatikan bahwa :

1. Gordan dimasukkan sebagai CUSTOMER(11), yakni simpul tersedia pertama.
2. Gordan disisipkan di muka Hunter, yang tadinya adalah pelanggan pertama dari Kelly.

Di sini terjadi 3 perubahan dalam Field Penuding sebagai berikut :
POINT(2) = 11 (sekarang list mulai dengan Gordan)
POINT(11) = 3 (Gordan menuding Hunter)
AVAIL = 18 (sekarang AVAIL menuding ke simpul tersedia berikutnya)



Gambar 14

BROKER	PELANGGAN
Bond	Grant, Scott, Vito, Katz
Kelly	Gordan, Hunter, McBride, Evans
Hall	-
Nelson	Teller, Jones, Adams, Rogers, Weston