Stack dan Queue dengan Representasi List Linier

Tim Pengajar IF2030

Semester I/2009-2010

Stack

- Stack: list linier yang:
  - dikenali elemen puncaknya (TOP)
  - aturan penyisipan dan penghapusan elemennya tertentu:
    - Penyisipan selalu dilakukan "di atas" TOP
    - Penghapusan selalu dilakukan pada TOP
- TOP adalah satu-satunya alamat tempat terjadi operasi
- Elemen Stack tersusun secara LIFO *(Last In First Out)*

Definisi Fungsional Stack

- Jika diberikan **S** adalah Stack dengan elemen **ElmtS**:

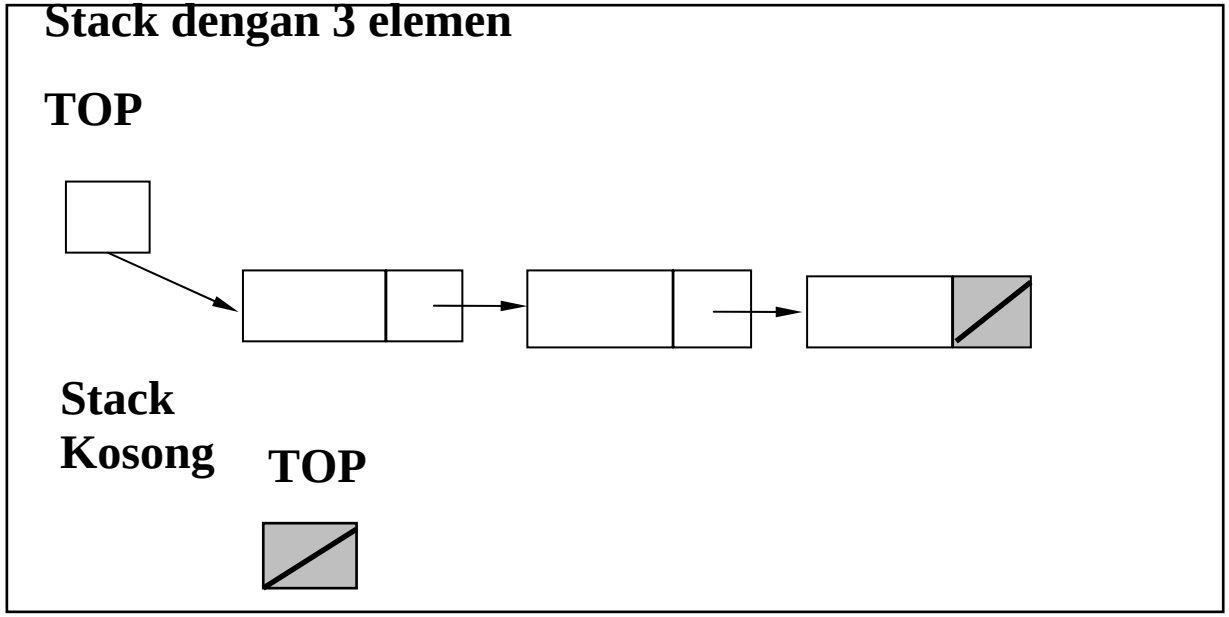
CreateEmpty	: $\rightarrow$ S	{ Membuat sebuah stack kosong}
IsEmpty	: S $\rightarrow$ boolean	{ Test stack kosong, true jika stack kosong, false jika S tidak kosong }
IsFull	: S $\rightarrow$ boolean	{ Test stack penuh, true jika stack penuh, false jika S tidak }
Push	: ElmtS x S $\rightarrow$ S	{ Menambahkan sebuah elemen ElmtS sebagai TOP, TOP berubah nilainya }
Pop	: S $\rightarrow$ S x ElmtS	{ Mengambil nilai elemen TOP, sehingga TOP yang baru adalah elemen yang datang sebelum elemen TOP, mungkin S menjadi kosong }

Definisi Fungsional

- Definisi Selektor:
  - Jika **S** adalah sebuah Stack, maka
    - **Top(S)** adalah alamat elemen TOP, di mana operasi penyisipan/penghapusan dilakukan
    - **InfoTop(S)** adalah informasi yang disimpan pada Top(S)
- Definisi **Stack kosong** adalah Stack dengan **Top(S)=Nil** (tidak terdefinisi)

Stack dengan Representasi List Linier

- List linier “biasa” ➔ stack



Operasi-Operasi Dasar pada Stack

- CreateEmpty → menjadi create list kosong
- Push → menjadi insert first dalam list linier
- Pop → menjadi delete first dalam list linier

ADT Stack Representasi List


```
/* File : linkstack.h */
#ifndef _LINKSTACK_H
#define _LINKSTACK_H
#include "boolean.h"
#include <stdlib.h>

#define Nil NULL
/* Deklarasi infotype */
typedef int infotype;
/* Stack dengan representasi berkait dengan pointer */
typedef struct tElmtStack * address;
typedef struct tElmtStack {
        infotype Info;
        address Next;
} ElmtStack;

/* Type stack dengan ciri TOP : */
typedef struct {
        address TOP;  /* alamat TOP: elemen puncak */
} Stack;
```

ADT Stack Representasi List


```
/* Selektor */
#define Top(S) (S).TOP
#define InfoTop(S) (S).TOP->Info
#define Next(P) (P)->Next
#define Info(P) (P)->Info

/* Prototype manajemen memori */
void Alokasi (address *P, infotype X);
/* I.S. Sembarang */
/* F.S. Alamat P dialokasi, jika berhasil maka Info(P)=X dan
        Next(P)=Nil */
/*      P=Nil jika alokasi gagal */
void Dealokasi (address  P);
/* I.S. P adalah hasil alokasi, P <> Nil */
/* F.S. Alamat P didealokasi, dikembalikan ke sistem */
```

ADT Stack Representasi List

```
/* ********* PROTOTYPE REPRESENTASI LOJIK STACK ***************/
boolean IsEmpty (Stack S);
/* Mengirim true jika Stack kosong: TOP(S) = Nil */
void CreateEmpty (Stack * S);
/* I.S. sembarang */
/* F.S. Membuat sebuah stack S yang kosong */
void Push (Stack * S, infotype X);
/* Menambahkan X sebagai elemen Stack S */
/* I.S. S mungkin kosong, X terdefinisi */
/* F.S. X menjadi TOP yang baru jika alokasi X berhasil, */
/*      jika tidak, S tetap */
/* Pada dasarnya adalah operasi Insert First pada list linier */
void Pop (Stack * S, infotype * X);
/* Menghapus X dari Stack S. */
/* I.S. S tidak mungkin kosong */
/* F.S. X adalah nilai elemen TOP yang lama, */
/*      elemen TOP yang lama didealokasi */
/* Pada dasarnya adalah operasi Delete First pada list linier */
#endif
```

Queue

- QUEUE adalah list linier yang:
  - dikenali elemen pertama (HEAD) dan elemen terakhirnya (TAIL)
  - aturan penyisipan dan penghapusan elemennya didefinisikan sebagai berikut:
    - Penyisipan selalu dilakukan setelah elemen terakhir
    - Penghapusan selalu dilakukan pada elemen pertama
  - satu elemen dengan yang lain dapat diakses melalui informasi NEXT
- Elemen Queue tersusun secara FIFO *(First In First Out)*

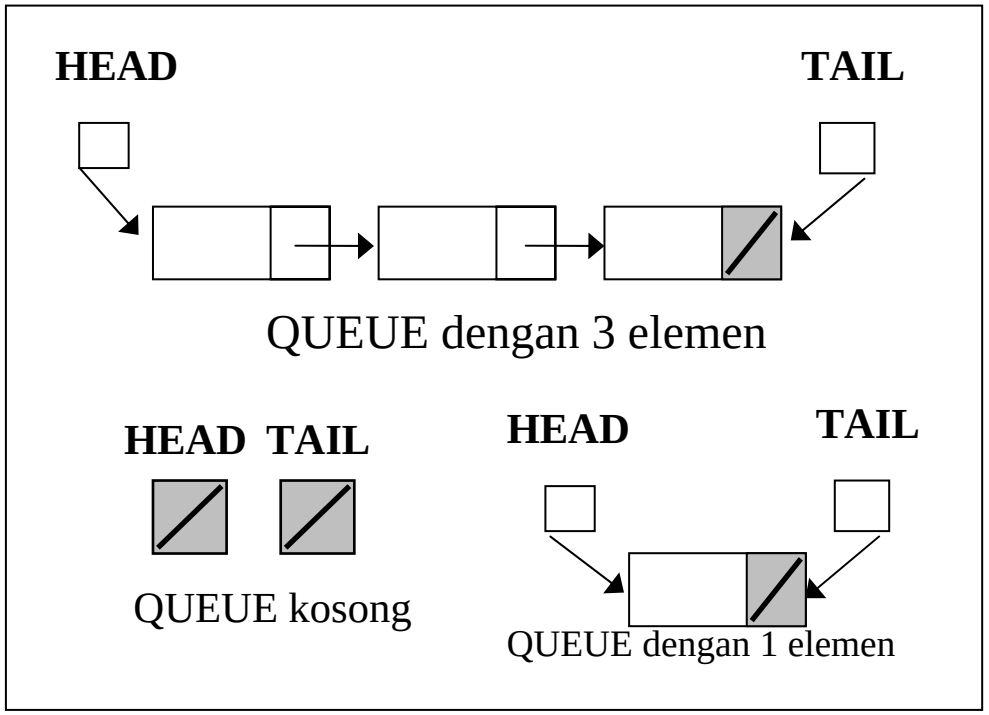
Definisi Fungsional

- Jika diberikan **Q** adalah QUEUE dengan elemen **ElmtQ**

```
IsEmpty        : Q → boolean   { Tes terhadap Q: true jika Q kosong,
                                 false jika Q tidak kosong }
IsFull         : Q → boolean   { Tes terhadap Q: true jika memori Q
                                 sudah  penuh,
                                 false jika memori Q tidak penuh }
NBElmt(Q)      : Q → integer   { Mengirimkan banyaknya elemen Q }
CreateEmpty    : → Q           { Membuat sebuah antrian kosong }
Add            : ElmtQ x Q → Q { Menambahkankan sebuah elemen setelah
                                 elemen ekor QUEUE }
Del            : Q → Q x ElmtQ { Menghapus kepala QUEUE,
                                 mungkin Q menjadi kosong }
```

Queue dengan Representasi List

- List Linier yang dicatat first dan last ➔ queue



Operasi-operasi dasar pada Queue

- CreateEmpty → menjadi create list kosong
- Add → menjadi insert last pada list
- Del → menjadi delete first pada list

ADT Queue dengan Rep. List


```
/* File : queuelist.h */
#ifndef _QUEUELIST_H
#define _QUEUELIST_H
#include "boolean.h"
#include <stdlib.h>

#define Nil NULL
/* Deklarasi infotype */
typedef int infotype;
/* Queue dengan representasi berkait dengan pointer */
typedef struct tElmtQueue * address;
typedef struct tElmtQueue {
        infotype Info;
        address Next;
} ElmtQueue;

/* Type queue dengan ciri HEAD dan TAIL : */
typedef struct {
        address HEAD;  /* alamat penghapusan */
        address TAIL;  /* alamat penambahan */
} Queue;
```

ADT Queue dengan Rep. List

```
/* Selektor */
#define Head(Q) (Q).HEAD
#define Tail(Q) (Q).TAIL
#define InfoHead(Q) (Q).HEAD->Info
#define InfoTail(Q) (Q).TAIL->Info
#define Next(P) (P)->Next
#define Info(P) (P)->Info

/* Prototype manajemen memori */
void Alokasi (address *P, infotype X);
/* I.S. Sembarang */
/* F.S. Alamat P dialokasi, jika berhasil maka Info(P)=X dan
        Next(P)=Nil */
/*      P=Nil jika alokasi gagal */
void Dealokasi (address  P);
/* I.S. P adalah hasil alokasi, P <> Nil */
/* F.S. Alamat P didealokasi, dikembalikan ke sistem */
```

ADT Queue dengan Rep. List


```
boolean IsEmpty (Queue Q);
/* Mengirim true jika Q kosong: HEAD(Q)=Nil and TAIL(Q)=Nil */
int NBElmt(Queue Q);
/* Mengirimkan banyaknya elemen queue. Mengirimkan 0 jika Q kosong */
/*** Kreator ***/
void CreateEmpty(Queue * Q);
/* I.S. sembarang */
/* F.S. Sebuah Q kosong terbentuk */
/*** Primitif Add/Delete ***/
void Add (Queue * Q, infotype X);
/* Proses: Mengalokasi X dan menambahkan X pada bagian TAIL dari Q
   jika alokasi berhasil; jika alokasi gagal Q tetap */
/* Pada dasarnya adalah proses insert last */
/* I.S. Q mungkin kosong */
/* F.S. X menjadi TAIL, TAIL "maju" */
void Del(Queue * Q, infotype * X);
/* Proses: Menghapus X pada bagian HEAD dari Q dan mendealokasi
   elemen HEAD */
/* Pada dasarnya operasi delete first */
/* I.S. Q tidak mungkin kosong */
/* F.S. X = nilai elemen HEAD pd I.S., HEAD "mundur" */

#endif
```

ADT Queue dengan Rep. List


```
void Add (Queue * Q, infotype X)
/* Proses: Mengalokasi X dan menambahkan X pada bagian TAIL dari Q
   jika alokasi berhasil; jika alokasi gagal Q tetap */
/* Pada dasarnya adalah proses insert last */
/* I.S. Q mungkin kosong */
/* F.S. X menjadi TAIL, TAIL "maju" */
{  /* Kamus Lokal */
   address P;
   /* Algoritma */
   Alokasi(&P,X);
   if (P!=Nil) {
      if (IsEmpty(*Q)) {
          Head(*Q) = P;
          Tail(*Q) = P;
      } else {
          Next(Tail(*Q)) = P;
           Tail(*Q) = P;
      }
   } /* else: alokasi gagal, Q tetap */
}
```

ADT Queue dengan Rep. List

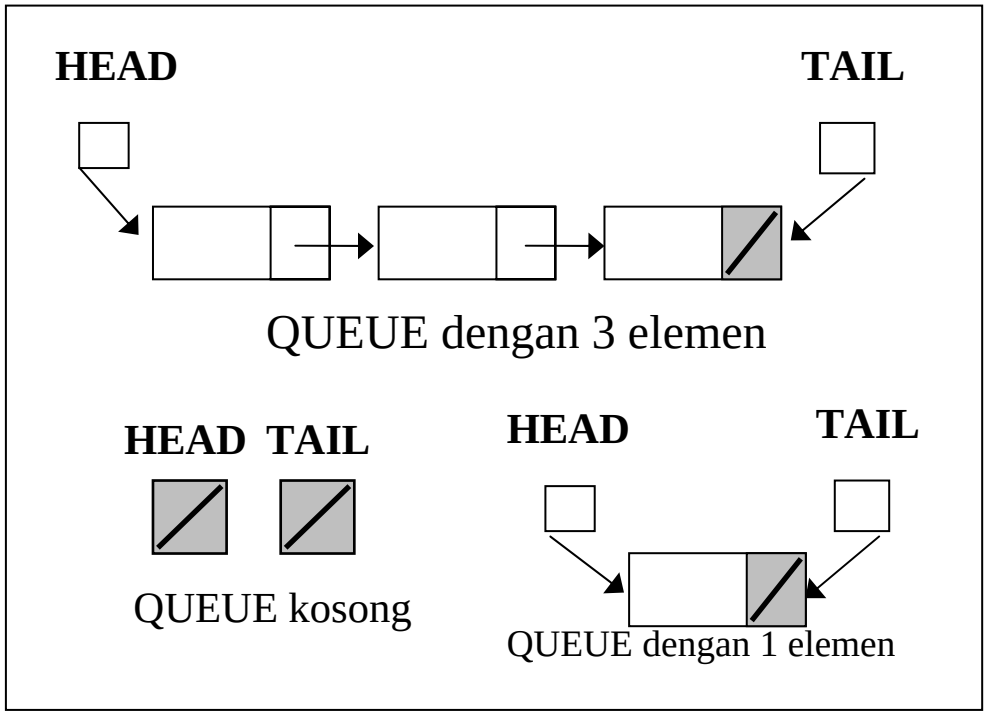
```
void Del(Queue * Q, infotype * X);
/* Proses: Menghapus X pada bagian HEAD dari Q dan mendealokasi
   elemen HEAD */
/* Pada dasarnya operasi delete first */
/* I.S. Q tidak mungkin kosong */
/* F.S. X = nilai elemen HEAD pd I.S., HEAD "mundur" */
{  /* Kamus Lokal */
   address P;
   /* Algoritma */
   *X = InfoHead(*Q);
   P = Head(*Q);
   Head(*Q) = Next(Head(*Q));
   if (Head(*Q)==Nil) {
      Tail(*Q) = Nil;
   }
   Dealokasi(P);
}
```

Priority Queue

- Priority queue:
  - Elemen queue terurut menurut suatu prioritas tertentu
  - Sering dianggap: *modified queue*
  - Menambahkan elemen berarti menambahkan elemen sesuai urutan prioritas
  - Menghapus elemen adalah menghapus elemen dengan prioritas tertinggi/terendah (pada bagian Head)

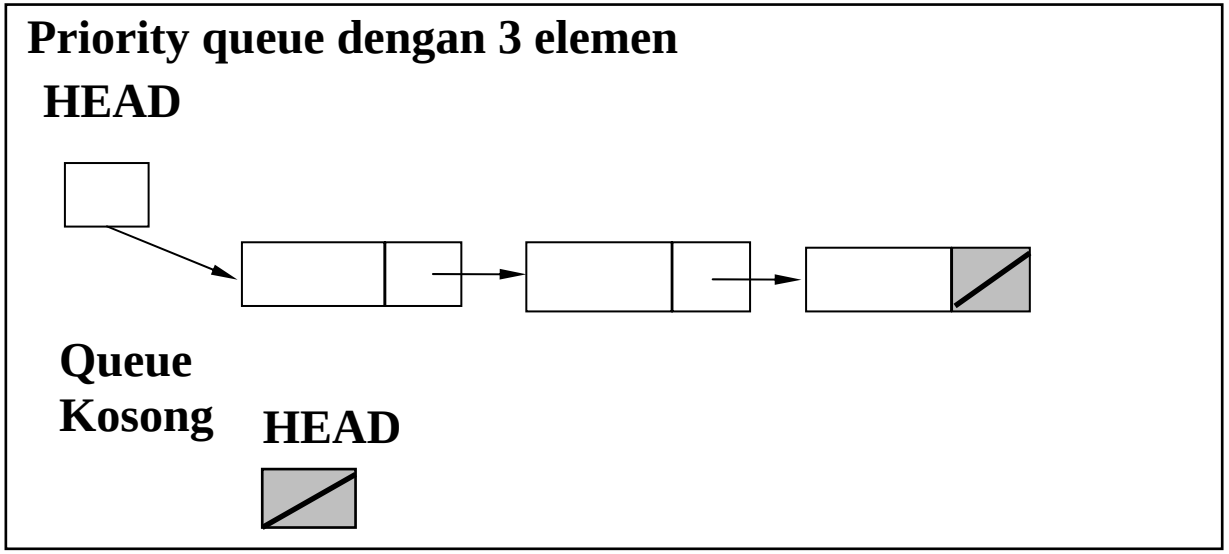
Priority Queue dengan Rep. List

- List Linier yang dicatat first dan last ➔ priority queue



Priority Queue dengan Rep. List

- List linier “biasa” ➔ priority queue



Operasi-operasi dasar Priority Queue

- Elemen queue:

```
type ElmtQueue : < Info : infotype,
                   Prio : integer,
                   Next : address >
```

- Add → menambahkan elemen secara terurut mengikuti prioritas tertentu
- Del → menghapus elemen dengan prioritas tertinggi/terendah (pada bagian Head)
  - pada dasarnya sama saja dengan Del pada queue/list biasa
- Perhatikan representasi lojik yang digunakan

PR

- Untuk praktikum 12:
  - Modul pra-praktikum: P12. ADT Stack - Representasi dengan List Linier
  - Modul pra-praktikum: P13. ADT Queue - Representasi dengan List Linier
  - Modul pra-praktikum: P14. ADT Priority Queue

Pengumuman

- Tugas Besar I
  - Pengumpulan hardcopy dan softcopy laporan dan program dimundurkan menjadi Rabu, 18 November 2009 pukul 17.00
  - Demo dimundurkan menjadi dari Kamis, 19 November 2009 s.d. Senin, 23 November 2009
  - Pengumpulan softcopy ke http://students.if.itb.ac.id/~if16046/tubes1/ karena server milestone sedang bermasalah.

Pengumuman

- Kuis-2
  - Waktu: Kamis, 19 November 2009 pukul 14.00 s.d. selesai
  - Tempat default: ruang kelas masing-masing
  - Materi:
    - stack+queue dengan representasi array
    - list linier (dengan seluruh representasi fisik)
    - variasi list linier
    - stack+queue dengan representasi list linier
  - Bahasa C
  - Bawa pensil dan penghapus