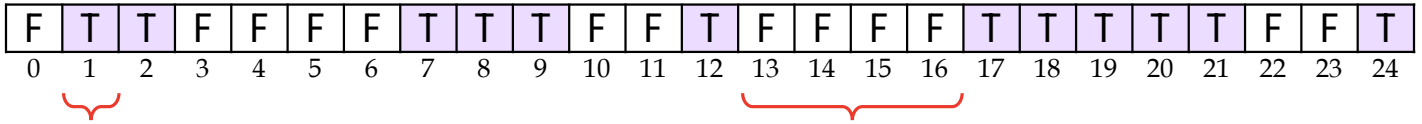
Studi Kasus: Pengelolaan Memori Representasi Kontigu

IF2110/IF2111 – Algoritma dan Struktur Data
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Memori

Jumlah blok: N_BLOCK = 25



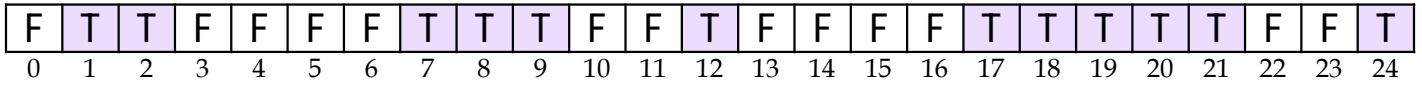
Satu blok
F: kosong
T: isi

Satu zone
⟨13,4⟩: zone bebas/kosong
⟨17,5⟩: zone isi
(zone ⟨i,n⟩: dimulai dari indeks i
sebanyak n blok)

Deskripsi Persoalan

Memori dinyatakan sebagai N_BLOCK buah blok kontigu

- F: KOSONG
- T: ISI



Zone Bebas: blok-blok berurutan yang berstatus F (KOSONG)

Dinyatakan oleh indeks blok KOSONG pertama dan ukurannya

- Zone bebas I: ⟨0,1⟩
- Zone bebas II: ⟨3,4⟩
- Zone bebas III: ⟨10,2⟩
- Zone bebas IV: ⟨13,4⟩
- Zone bebas V: ⟨22,2⟩

Deskripsi Persoalan

Alokasi dan dealokasi memori menyebabkan perubahan terhadap zone bebas

Realisasikan prosedur-prosedur sebagai berikut:

- Prosedur **initMem** mengeset semua blok menjadi blok KOSONG
- Prosedur **allocBlock** melakukan alokasi: membuat blok KOSONG sejumlah x menjadi ISI; menghasilkan indeks awal ($startIdx$) di mana alokasi dilakukan
- Prosedur **deallocBlock** "membebaskan" zone ISI: membuat blok ISI sejumlah x yang berawal di indeks $startIdx$ menjadi KOSONG
- Prosedur **compaction** *(memory compaction)* memampatkan memori sehingga semua blok KOSONG berada di bagian kiri memori dan semua blok ISI berada di kanan memori

Representasi secara kontigu dengan array

Representasi dengan Array: Struktur Data

```
KAMUS
  constant UNDEF: integer = -1
  constant N_BLOCK: integer = 100
  STATMEM: array [0..N_BLOCK-1] of boolean
  { tabel status memori: true jika ISI, false jika kosong }
```

Representasi dengan Array: Prosedur initMem

Secara umum: set seluruh elemen STATMEM menjadi bernilai false

Algoritma: Diktat hlm. 167

```
procedure initMem
{ I.S.: Sembarang }
{ F.S.: Semua blok memori dinyatakan KOSONG }
{ Proses: Semua status blok dengan indeks [0..N_BLOCK-1] dijadikan
  KOSONG dengan traversal blok [0..N_BLOCK-1].
  Proses sekuensial tanpa penanganan kasus kosong (N_BLOCK ≥ 0) }
```

Representasi dengan Array: Prosedur allocBlock (1)

```
procedure allocBlock (input x: integer, output startIdx: integer)
{ I.S.: Sembarang.
        x adalah banyaknya blok yang diminta untuk dialokasi,
        yaitu dijadikan ISI.
{ F.S.: Jika ada zone yang memenuhi syarat (ada zone dengan jumlah
        blok kontigu sebanyak x atau lebih yang berstatus kosong),
        maka startIdx akan berisi indeks blok bebas pertama pada zone
        yang dipilih untuk dialokasi;
        kemudian status pemakaian memori zone tsb. dimutakhirkan.
        Jika tidak ada lagi zone yang memenuhi syarat (tidak ada
        zone dengan jumlah blok sebanyak x) maka startIdx bernilai UNDEF
        dan status blok tetap seperti pada I.S. }
```

Representasi dengan Array: Prosedur allocBlock (2)

Strategi First Fit: Alokasi dilakukan pada zone yang memenuhi syarat yang **pertama kali ditemukan**.

Diktat hlm. 168

Sketsa umum algoritma:

inisialisasi
repeat
  *cari blok kosong pertama (skema **search**)*
  if *ketemu blok kosong* **then**
    catat indeks sebagai `NAwal`*, yaitu blok awal zone kosong*
    hitung `NKosong`*, yaitu banyaknya blok dlm zone kosong tersebut*
until *semua blok sudah diperiksa* `or NKosong ≥ x`
terminasi
if *ada zone memenuhi syarat* **then**
  ubah status semua blok pada zone tersebut [startIdx..startIdx+x−1] menjadi ISI

Representasi dengan Array: Prosedur allocBlock (3)

Strategi Best Fit: Alokasi dilakukan terhadap zone yang memenuhi syarat dan berukuran sama dengan x, atau jika tidak ada zone berukuran x, maka diambil yang ukurannya minimal. Keuntungan: blok tidak terpartisi kecil-kecil

Diktat Hlm. 169

```
inisialisasi
repeat
    cari zone kosong pertama (skema search)
    if ketemu blok kosong then
        hitung banyaknya blok dlm zone kosong tsb.
    if ada zone kosong dan memenuhi syarat then
        if zone kosong pertama then
            inisialisasi
            ukuran zone Minimum: NBMin dan Posisi Awal NAwal
        else cek apakah lebih baik, jika ya, update NBMin dan startIdx
until semua blok diperiksa or blok berukuran = x
terminasi
if ada zone memenuhi syarat then
    ubah status blok pada zone tersebut [startIdx..startIdx+x−1] menjadi ISI
```

Representasi dengan Array: Prosedur deallocBlock

Secara umum: Pemrosesan sekuensial (traversal) untuk membuat status blok [startIdx..startIdx+x-1] KOSONG

Diktat hlm. 170

```
procedure deallocBlock (input x, startIdx: integer)
{ I.S.: x adalah ukuran zone, bilangan positif dan startIdx adalah
        alamat blok awal zone tersebut, dengan startIdx ∈ [0..N_BLOCK-x],
        Blok dengan indeks startIdx s.d. startIdx+x-1 pasti berstatus ISI. }
{ F.S.: Tabel status memori dengan indeks blok startIdx..startIdx+x-1
        menjadi KOSONG }
{ Proses: Sebuah zone berukuran x dan berawal pada blok startIdx
          di-dealokasi (statusnya dijadikan kosong) }
```

Representasi dengan Array: Prosedur compaction (1)

```
procedure compaction
{ I.S.: Sembarang }
{ F.S.: Tabel status memori menjadi dua bagian:
        - zone KOSONG di belahan "kiri" (indeks kecil),
        - zone ISI di belahan "kanan".
        Jika k adalah integer 0..N_BLOCK yang merupakan indeks blok
        ISI pertama pada zone ISI, maka ada 3 kemungkinan
        F.S.:
        1. Jika k = 0  maka semua blok adalah ISI, tidak ada
           zone kosong. }
        2. Jika 0 < k < N_BLOCK, maka blok dengan indeks[0..k-1]
           adalah zone KOSONG, bagian dengan indeks [k..N_BLOCK-1]
           adalah zone ISI
        3. Jika k = N_BLOCK maka semua blok adalah
           KOSONG, memori terdiri dari sebuah zone KOSONG.
```

Representasi dengan Array: Prosedur compaction (2)

Proses: “menggeser” elemen tabel yang berstatus KOSONG ke kiri

Dua pass (diktat hlm. 170):

- Traversal untuk mencacah banyaknya blok kosong, misalnya NKosong
- Traversal untuk memberi status [0..NKosong−1] dengan KOSONG dan [NKosong..N_BLOCK−1] dengan ISI

Satu pass:

- Traversal untuk mengelompokkan KOSONG di kiri dan ISI di kanan dengan menukarkan dua elemen.