

Implementasi ADT List dengan Array dinamis

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

List dan array dinamis

Implementasi-implementasi ADT List yang telah dibahas memiliki keterbatasan kapasitas sesuai alokasi *array*.

Pada definisi ADT List secara logik tidak mengharuskan ada batasan jumlah elemen.

Bagaimana mengimplementasikan List yang jumlah elemennya tak terbatas menggunakan *array*?

List dan array dinamis

Menggunakan *array* dinamis:

- Ketika *array* sudah penuh dan hendak dilakukan penambahan elemen, alokasikan *array* baru yang lebih besar (misal $2 \times$ ukuran yang lama).
- Ketika *array* sudah “sepi” (misal hanya $\leq 25\%$ terisi), alokasikan *array* baru yang lebih kecil (misal $0.5 \times$ ukuran yang lama).

Contoh: alt-2a (rata kiri), dinamis

constant INITIAL_CAP: integer = 100

type ElType: integer { *elemen list* }

type List: < contents: array of ElType, { *penyimpanan elemen List.* }
capacity: integer, { *dibutuhkan pencatat ukuran alokasi saat ini.* }
nEff: integer > { *pencatat jumlah elemen efektif.* }

{ *Konstruktor* }

procedure CreateList(output l: List)

{ *Membentuk List kosong dengan kapasitas awal initialSize.* }

KAMUS LOKAL

-

ALGORITMA

l.capacity ← INITIAL_CAP

alokasi(l.contents, l.capacity)

l.nEff ← 0

Contoh: insertAt (alt-2a dinamis)

```
procedure insertAt(input/output l: List, input x: ElType, input idx: integer)  
{ ... }
```

KAMUS LOKAL

tmpArray: array of ElType

i: integer

ALGORITMA

```
if length(l) < l.capacity then { masih muat }
```

```
  i traversal [length(l)..idx+1]
```

```
    l.contents[i] = l.contents[i-1]
```

```
  l.contents[idx] = x
```

```
  l.nEff ← l.nEff+1
```

```
{ else: bersambung... }
```

Contoh: insertAt (alt-2a dinamis)

```
else { length(l)=l.capacity (tidak muat) }  
  alokasi(tmpArray, l.capacity)  
  i traversal [0..l.capacity-1]  
    tmpArray[i] ← l.contents[i]  
  dealokasi(l.contents);  
  alokasi(l.contents, l.capacity*2)  
  i traversal [0..idx-1]  
    l.contents[i] ← tmpArray[i]  
  l.contents[idx] ← x  
  i traversal [idx..l.capacity-1]  
    l.contents[i+1] ← tmpArray[i]  
  dealokasi(tmpArray)  
  l.capacity ← l.capacity*2  
  l.nEff ← l.nEff+1
```

*{ Catatan 1: dalam Bahasa C kita dapat menggunakan fungsi realloc() sehingga tidak perlu menyalin isi array secara manual. Namun ingat, realloc() bisa gagal.
Catatan 2: temukan room for improvement algoritma di atas }*

Contoh: deleteAt (alt-2a dinamis)

```
procedure deleteAt(input/output l: List, input idx: integer, output e: ElType)  
{ ... }
```

KAMUS LOKAL

tmpArray: array of ElType

newCap, i: integer

ALGORITMA

e ← getElmt(l, idx)

if length(l) > l.capacity div 4 then { *belum “sepi”* }

l.nEff ← l.nEff-1

i traversal [idx..length(l)-1]

l.contents[i] ← l.contents[i+1]

{ *else: bersambung...* }

Contoh: deleteAt (alt-2a dinamis)

```
else { length(l) ≤ l.capacity div 4 (sudah “sepi”) }  
  newCap ← max2(INITIAL_CAP, l.capacity div 2) { mencegah newCap = 0 }  
  alokasi(tmpArray, newCap)  
  i traversal [0..idx-1]  
    tmpArray[i] ← l.contents[i]  
  l.nEff ← l.nEff-1  
  i traversal [idx..length(l)-1]  
    tmpArray[i] ← l.contents[i+1]  
  dealokasi(l.contents)  
  alokasi(l.contents, newCap)  
  i traversal [0..length(l)-1]  
    l.contents[i] ← tmpArray[i]  
  l.capacity ← newCap  
  dealokasi(tmpArray)
```

{ temukan potensi masalah kinerja! }