STRUKTUR DATA

JULIO ADISANTOSO
Departemen Ilmu Komputer IPB

Pertemuan 4 : 30 Juni 2015

Pengolahan Linked List

Pengolahan Linked List

- Sebelumnya telah dipelajari beberapa operasi dasar yang merupakan spesifikasi PRIMITIF dari Linked List (SLL dan DLL), yaitu: `make, push (front, after, back), find, del`.
- Pengolahan lanjut terhadap Linked List (LL) yang sering dilakukan adalah:
  - Merge : menggabungkan dua LL
  - Sort

Merge Dua LL

- Diketahui LL1 dan LL2, misalnya:
  - `LL1 = {5,10,12,3}`
  - `LL2 = {2,15,7}`
- Diinginkan untuk menggabungkan LL2 ke dalam LL1
- Cara penggabungan:
  - LL2 di belakang LL1 $\Rightarrow$ `LL1={5,10,12,3,2,15,7}` (append)
  - Dengan syarat tertentu, misalnya *merge sort*

$$\left.\begin{array}{l} \texttt{LL1=\{3,5,10,12\}} \\ \\ \texttt{LL2=\{2,7,15\}} \end{array}\right\} \texttt{LL=\{2,3,5,7,10,12,15\}}$$

Algoritme Append

Prosedur Append

Input: LL1,LL2
Output: LL1+LL2
if *(LL1.isEmpty())* **then**
 | $\text{head}_{LL1} \leftarrow \text{head}_{LL2}$
end
else
 | $(\text{tail}_{LL1}\text{->next}) \leftarrow \text{head}_{LL2}$
end

Bagaimana kode programnya? Harus ada fungsi publik untuk mengambil `head` dari LL.

Algoritme Append

Tambahkan fungsi getHead()

```
Node* getHead() {return head;}
```

Fungsi append(LL)

```
void SLL::append(SLL pL) {
   if (isEmpty()) head=pL.getHead();
   else tail->next=pL.getHead();
}
```

Contoh driver

```
list.append(list2); list.print();
```

Algoritme Append

Append menggunakan STL (tambahan p03a.cpp)

```
SLLP::iterator getHead() {return dt.begin(); }
SLLP::iterator getTail() {return dt.end(); }
void append(SLL pL) {
    SLLP::iterator it=dt.begin();
    SLLP::iterator pra=it;
    for (; it!=dt.end(); pra=it, ++it);
    dt.insert_after(pra, pL.getHead(), pL.getTail());
}
```

Algoritme Merge

Prosedur Merge

Input: LL1,LL2
Output: LL1+LL2

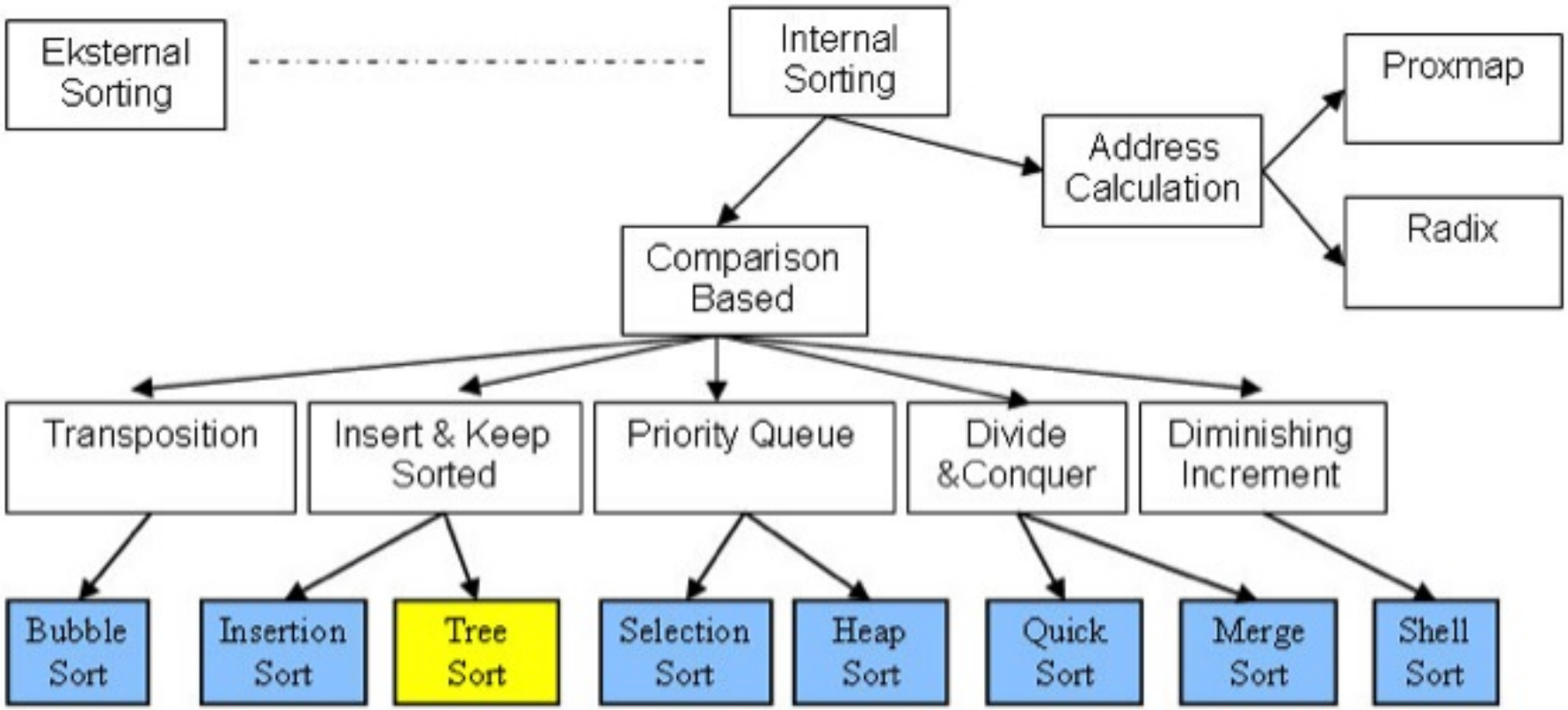
1: LL1.sort()
2: LL2.sort()
3: LL1.merge(LL2)

- Membutuhkan fungsi sort.
- Kita bahas dahulu algoritme sort

Sort

- Menyusun sekelompok elemen data yang tidak terurut menjadi terurut berdasarkan suatu kriteria tertentu.
- Mempermudah dan mempercepat proses pencarian data
- Jika pencarian data mudah, maka proses manipulasi data juga akan lebih cepat.

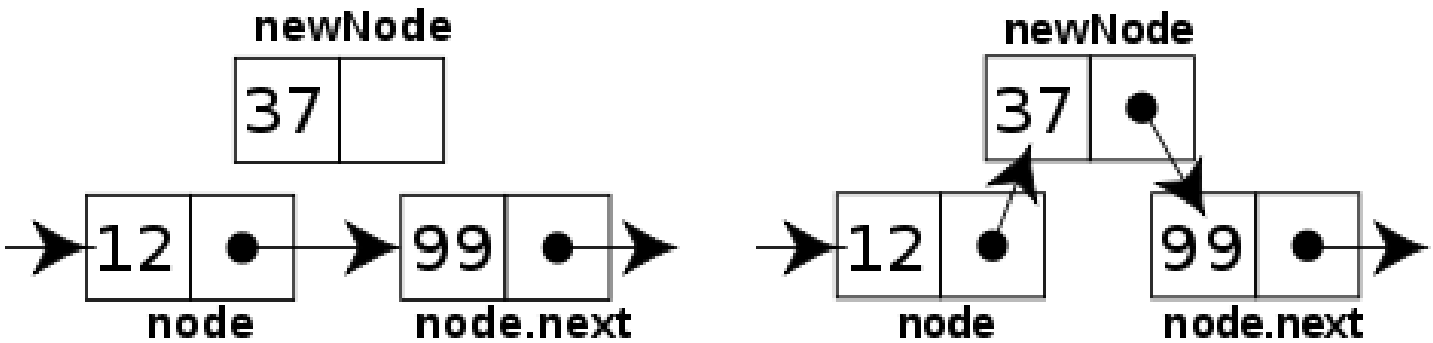
Metode Sort



Prosedur dan Kinerja dibahas dalam MK Analisis Algoritme

Ordered Linked List

- Melakukan sort terhadap suatu linked list membutuhkan sumberdaya yang tidak sedikit (mengapa?)
- Dijaga agar suatu linked list tetap terurut berdasarkan key tertentu $\Rightarrow$ **Ordered Linked List**
- Spesifikasi PRIMITIF `push` berusaha agar linked list tetap terurut.



Ordered Linked List : Contoh Kasus

Diinginkan DLL untuk menyimpan n data nim, jenkel, dan tinggi:

Contoh input

```
5
G64090120 L 165.4
G64090160 P 169.7
G65090025 L 167.9
G64090140 L 170.2
G64090050 P 168.3
```

Susunan dalam DLL

```
G64090050 P 168.3
G64090120 L 165.4
G64090140 L 170.2
G64090160 P 169.7
G65090025 L 167.9
```

Ordered Linked List : Contoh Kasus

Driver

```
int main() {
    OrderedDLL list;
    int n;
    string nim; char jenkel; float tinggi;
    cin >> n;
    while (n--) {
        cin >> nim >> jenkel >> tinggi;
        list.push(nim,jenkel,tinggi);
    }
    list.print();
    return 0;
}
```

Ordered Linked List : Contoh Kasus

Spesifikasi TYPE (p04a.cpp)

```
#include <iostream>
#include <list>
#include <string>
using namespace std;
struct myData {
    string nim;
    char jenkel;
    float tinggi;
};
typedef struct myData MD;
typedef list<MD> myType;
```

Ordered Linked List : Contoh Kasus

Spesifikasi PRIMITIF (p04a.cpp)

```
class OrderedDLL {
   myType dt;
   MD makeNode(string nim, char jenkel, float tinggi) {
     MD t;
     t.nim=nim; t.jenkel=jenkel; t.tinggi=tinggi;
     return t;
   }
   myType::iterator posisi(string nim);
   myType getData() {return dt; }
   public:
     void push(string nim, char jenkel, float tinggi);
     void print();
};
```

Ordered Linked List : Contoh Kasus

Fungsi posisi (p04a.cpp)

```
myType::iterator OrderedDLL::posisi(string nim) {
   myType::iterator it=dt.begin();
   for(; it!=dt.end(); ++it)
      if (it->nim>=nim) return it;
   return it;
}
```

Ordered Linked List : Contoh Kasus

Fungsi push (p04a.cpp)

```
void OrderedDLL::
push(string nim, char jenkel, float tinggi) {
    MD node=makeNode(nim,jenkel,tinggi);
    if (dt.empty()) dt.push_front(node);
    else {
        myType::iterator it=posisi(nim);
        if (it==dt.end()) dt.push_back(node);
        else dt.insert(it, node);
    }
}
```

Ordered Linked List : Contoh Kasus

Fungsi print (p04a.cpp)

```
void OrderedDLL::print() {
   myType::iterator it;
   for (it=dt.begin(); it!=dt.end(); ++it)
      cout << it->nim << " "
           << it->jenkel << " "
           << it->tinggi << endl;
}
```

Ordered Linked List : Contoh Kasus

Fungsi merge (p04a.cpp)

```
void OrderedDLL::merge(OrderedDLL pL) {
   myType dt2=pL.getData();
   myType::iterator it=dt.begin();
   myType::iterator it2=dt2.begin();
   for(;it2!=dt2.end();++it2) {
      MD t={it2->nim,it2->jenkel,it2->tinggi};
      while (it->nim<it2->nim && it!=dt.end())
         ++it;
      dt.insert(it,t);
   }
}
```

Doubly Linked List

Bagaimana jika DLL biasa? Bukan Ordered DLL?
Sangat mudah karena menggunakan STL `list`.

Fungsi membandingkan NIM (p04b.cpp)

```
static bool cmpNIM(MD a, MD b) {
   return (a.nim<b.nim); }
```

Fungsi merge

```
void DLL::merge(DLL pL) {
   myType dt2=pL.getData();
   dt.sort(cmpNIM);
   dt2.sort(cmpNIM);
   dt.merge(dt2,cmpNIM);
}
```

Tugas #4

Pak Agri memiliki banyak petak kebun penelitian tanaman jagung. Setiap petak berisi sejumlah tanaman jagung yang berbeda-beda, dan dicatat tinggi tanaman masing-masing. Di akhir penelitian, Pak Agri mengambil n petak sampel secara acak. Sampel ini kadang ditambah, kadang dikurang, maupun ditukar. Ingin dicetak data sampel tersebut terurut berdasarkan kode petak (URUTKODE) atau rata-rata tinggi tanaman di dalam petak (URUTRATA).

Format input

Baris pertama adalah banyaknya petak (n). Sebanyak n baris berikutnya adalah kode petak, banyaknya sampel, dan tinggi tanaman setiap sampel. Baris terakhir adalah key untuk sort, URUTKODE atau URUTRATA.

Tugas #4

Contoh input

```
5
F123 5 56.2 32.4 14.2 43.7 10.5
B651 2 89.9 87.2
G769 1 87.5
A902 3 11.9 13.6 20.4
C300 2 56.5 67.3
URUTRATA
```

Contoh output

```
A902 : 11.9->13.6->20.4->NULL
F123 : 56.2->32.4->14.2->43.7->10.5->NULL
C300 : 56.5->67.3->NULL
G769 : 87.5->NULL
B651 : 89.9->87.2->NULL
```