UJIAN TENGAH SEMESTER II TH. AJARAN 2002/2003

STRUKTUR DATA & ALGORITMA (IKI10100)

FAKULTAS ILMU KOMPUTER UNIVERSITAS INDONESIA
SELASA, 15 APRIL 2003 (90 menit)
Closed Book

Sorting

1. **(25 point)** Kita tahu bahwa algoritma sorting melibatkan banyak proses *swapping*. Seandainya *cost* terbesar adalah pada *space* (misalnya karena keterbatasan jumlah register), maka performance bisa ditingkatkan jika proses swapping tidak melibatkan variable tambahan. Tunjukkan bahwa untuk menukar nilai integer dalam dua variable tidak perlu memanfaatkan variable ketiga sebagai *temporary storage*. *Hint:* gunakan operasi penjumlahan dan/atau pengurangan!

2. Perhatikan code fungsi XSort berikut ini!

```
public void XSort (int data[], int n) {
// pre: 0 <= n <= data.length
// post: values in data[0..n-1] are in ascending order
   int index, max, numUnsorted = n;
   while( numUnsorted > 0 ) {
      max = 0;
      for (index = 1; index < numUnsorted; index++) {
         if (data[max] < data[index]) max = index;
      }
   swap(data, max, numUnsorted-1); // elemen di [index max]
                                   // ditukar dgn elemen
                                   // di index [numUnsorted-1]
   numUnsorted--;
}
```

   a) **(10 point)** Berdasar algoritma yang sudah anda kenal, code ini menggambarkan proses sorting apa? (BubbleSort, SelectionSort, InsertionSort, QuickSort atau MergeSort)

   b) **(30 point)** Sebuah algoritma sorting yang bekerja terhadap elemen-elemen integer dikatakan *stable* jika memenuhi syarat berikut:

      "Beberapa item yang sama nilanya, **urutan posisi** satu **relatif** terhadap yang lain akan **tetap** selama proses sorting".

      Misalnya ada dua buah item dengan nilai 5, satu disebut a dan satunya disebut b, dengan posisi awal index a < index b. Jika selama proses sorting selalu terpenuhi index a < index b, maka algoritma sorting tersebut dikatakan *stable*.

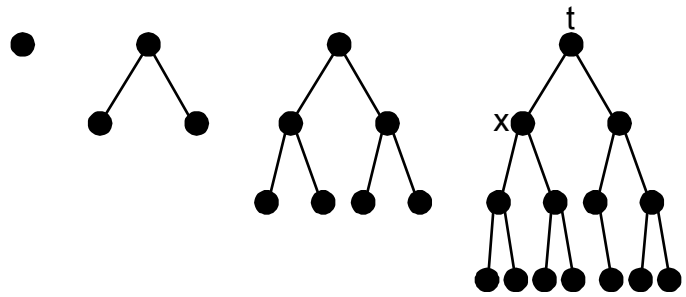
      Modifikasi code di atas agar XSort menjadi *stable*!

Mathematical Induction

3. **(25 point)** Buktikan dengan matematika induksi bahwa $n^3 + 2n$ selalu habis dibagi 3 untuk $n > 0$!

BinaryTree & Recursion

4. **(30 point)** Sebuah node dalam binary tree dikatakan *full* jika memiliki *degree* 2 (dua anak). Sebuah binary tree dengan *height* h dikatakan *full* jika hanya punya *leaves* pada *level* h dan setiap *internal node*-nya adalah *full*. Penambahan sebuah node saja pada sebuah binary tree yang sudah full akan menyebabkan peningkatan height pada tree tersebut. Hanya tiga tree pertama di bawah ini yang full:



Implementasikan secara recursive method **boolean isFull(BinaryTreeNode n)** yang me-return true jika tree yang root-nya n adalah full, dan false jika tidak. Dalam contoh tree di atas, isFull(t) = false, sedangkan isFull(x) = true.

Linked-List

5. **(30 point)** Implementasikan sebuah method Split() yang dipakai untuk membagi list *source* menjadi dua list yang panjangnya berimbang. List paruh pertama ditunjuk oleh *front*, dan list paruh kedua ditunjuk oleh *back*. Jika jumlah elemen dalam *source* genap, maka *front* dan *back* memiliki *length* yang sama. Jika jumlah elemen dalam *souce* ganjil, maka elemen *front* selisih satu lebih banyak daripada *back*. Return false jika gagal, misal jumlah elemen hanya 1.

   Contoh: Split( ) pada *source* = {2,3,5,7,11} menghasilkan dua list: *front* = {2,3,5} dan *back* = {7,11}.

```
boolean Split(List source, List front, List back) {
   …… code anda di sini ……
}
```

   Catatan: *full-mark* untuk implementasi Split tanpa menghitung terlebih dahulu jumlah elemen dalam list. Asumsikan method length() cost-nya sangat besar.

-oOo-

Soal selesai di sini. Selamat bekerja!
PM - 150403