Faculteit Wiskunde & Informatica Toets Datastructuren

Vrije Universiteit 18 mei 1998

tijdsduur : 1.5 uur

U I T W E R K I N G E N

==========

Opgave 1.

a) orde linear search: O(n)

orde van een somple sort: O(n^2)

orde van binary search: O(log n)

aantal handelingen bij linear search: 1000

aantal handelingne bij een simple sort: 1000000

aantal handelingen bij binary search: 10

b) Zie boek blz. 185 - 186

de data zijn gesorteerd

- neem het middelste element
- kijk of dit het gezochte element is
- zo nee, zoek dan verder, op dezelfde manier, in de linker deelrij

als het gezochte element kleiner is dan het middelste element en in de rechter deelrij als het gezochte groter is dan het middelste element

c) Zie boek blz. 553, 554 en 573

Een hash tabel is een array waarin data wordt opgeslagen op een door een functie (hash function) aan de hand van de data berekende index-plaats (de key). Niet alle data hashed op een unieke key. Indien twee data op dezelfde key hashen wordt dit een collision genoemd.
Bij hashing met seperate chaining wordt het probleem van collisions opgelost doordat de hash tabel voor iedere key een lijst bevat van de data die op die key hashen.

♀Opgave 2

```
class Lijst {
        Knoop first,
              last;
        int   aantalKnopen;

        Lijst () {
           init();
        }

        Lijst (Lijst l) {
           init();
```

```
        for (int i=0, Knoop k=l.first; i < l.aantalKnopen; i++, k=k.next) {
                insert(k.data);
        }
    }

void init () {
        first = last = null;
        aantalKnopen = 0;
    }

    void insert (int data) {
        if (aantalKnopen == 0) {
                last.next=first .prior=last=first=new Knoop(data, null, null);
        } else { // aantalKnopen >= 1
            first = last.next = first.prior = new Knoop(data, last, first);
        }
        aantalKnopen += 1;
    }

    Knoop find (int data) {
        for (int i=0, Knoop k=first; i < l.aantalKnopen; i++, k=k.next) {
                if (k.data == data) {
                    return k;
                }
        }
```

```
    return null;
}

boolean aanwezig (int data) {
    return find(data) != null;
}

void remove (int data) {
    if (aantalKnopen == 1) {
            init();
    } else { // aantalKnopen > 1
            // Zoek op waar de te verwijderen knoop zit.
        Knoop k = find(data);

            // Verwijder de knoop.
            k.prior.next = k.next;
            k.next.prior = k.prior;

            // Pas eventueel first of last aan.
            if (k == first) {
                first = first.next;
            } else
            if (k == last) {
                last = last.prior;
            }
```

```
            }
    }
}
```