Teknillinen korkeakoulu TENTTI
Tietotekniikan laitos 15.5.2008
T-106.1220/1223 Tietorakenteet ja algoritmit T/Y

Kirjoita jokaiseen paperiin oma nimesi, oppilasnumerosi *tarkistusmerkkeineen*, koulutusohjelmasi, kurssikoodi ja kurssin nimi, päivämäärä, sali, palauttamiesi paperien lukumäärä sekä *allekirjoituksesi*.

1) Koodin analyysi (3p + 2p + 3p + 2p)

Liitteenä oleva koodi on kurssin valmiskoodikirjastosta. `TrakBigFloat`-luokkaa on riisuttu joidenkin metodien osalta ja osa kommenteista on poistettu. Lisäksi siihen on toteutettu kertolasku (`multiply`).

Tutustu liitteen koodiin huolella. Vastaa sen jälkeen seuraaviin kysymyksiin. Tämä tehtävä on *tentin pakollinen osa*, josta on saatava vähintään 5p/10p, jotta loput tentistä tarkistetaan. Tämä tehtävä ei kuitenkaan yksistään riitä tentin läpäisyyn.

a) *Nimeä* (2–3 sanaa/kohta) riveiltä `1, 3, 7, 13, 19, 26, 66` ja `84` alkavat *ohjelmakoodin osat*.

b) *Kirjoita* muutama rivi *koodia*, joista ilmenee miten `TrakBigFloat`-luokan kertolaskua (`multiply`) voi kutsua ja tulostaa vastauksen.

d) *Selitä* kertolaskun (`multiply`, rivit 26-58) toimintaperiaate *sanallisesti*.

c) *Analysoi* kertolaskun (`multiply`, rivit 26-58) *suoritusaika*. Perustele vastauksesi ja anna lopputulos *Iso O -notaatiossa*.

2) Terminologiaa (2p + 2p + 2p + 2p)

Määrittele seuraavat *käsitteet* (4 x 1p). *Anna* jokaisesta myös *esimerkki* (4 x 1p).

a) Kekoehto, b) Lineaarinen kokeilu, c) Valinta, d) Hajautusfunktio.

3) Järjestäminen (3p + 3p + 3p)

Joudut valitsemaan algoritmin tehtävään, jossa tulee järjestää annettu aineisto. Mitä asioita (kriteereitä) huomioit tehdessäsi valintaa? Lue ensin koko tehtävänanto.

a) *Valitse kolme* (3) keskeistä *kriteeriä* joiden valossa tarkastelet tilannetta. *Perustele* miksi tai miten valitsemasi kriteerit liittyvät järjestämisongelmaan.

b) *Nimeä* jokaisen kriteerin kohdalla erikseen ainakin yksi *algoritmi*, joka täyttää ko. kriteerin ja yksi joka ei täytä (algoritmien toimintaperiaatetta ei tarvitse selittää).

Anna b-kohdan vastauksesi matriisimuodossa, jossa sarakkeilla (3) on kriteerit ja niiden alapuolella riveillä (2) algoritmien nimiä, jotka täyttävät ja eivät täytä ko. kriteeriä.

c) *Nimeä algoritmi*, joka täyttää **kaikki** kriteereistäsi (1p). Saat kuitenkin kaksi pistettä, jos esität sellaiset 3 kriteeriä, joita kaikkia yksikään kurssilla esitetty menetelmä ei täytä. *Nimeä* myös *algoritmi*, joka ei täytä **monta** kriteereistäsi (1p). **KÄÄNNÄ**

4) Tasapainotetut hakupuut (3p + 4p + 2p)

a) *Määrittele* **jokin** seuraavista hakurakenteista: *AVL-puu* **tai** *Punamusta puu* **tai** *2-3-4-puu*.

b) *Anna esimerkki* määrittelemästäsi rakenteesta: *lisää* alun perin tyhjään rakenteeseen *avaimet* **A**, **L**, **G**, **O**, **R**, **I**, **T**, **M**, **I** ja **T** tässä järjestyksessä. Käytä bottom-up –tasapainotusta. *Esitä välivaiheina* rakenne jokaisen lisäyksen jälkeen sekä jokaisen tasapainotuksen jälkeen. Selitä lyhyesti jokainen operaatio myös sanallisesti.

c) *Mikä on* rakenteen haku-, lisäys- ja poisto-operaatioiden *suoritusaika iso O -notaatiossa*? Perustele vastauksesi.

5) Verkot (2p + 4p)

Maanviljelijällä on joen toisella rannalla kaali, vuohi ja susi, jotka pitäisi kuljettaa joen vastakkaiselle puolelle. Viljelijä voi siirtää veneellään vain yhden kerrallaan joen yli, eikä vuohta voi jättää valvomatta kaalin kanssa eikä sutta vuohen kanssa.

a) *Kuvaa ongelma verkon avulla* siten, että se voidaan ratkaista algoritmisesti. *Esitä* vastauksesi *vieruslistana*.

b) *Esitä algoritmi*, jolla voit hakea ja tulostaa verkosta ratkaisun, johon tarvitaan vähiten kuljetuksia. Mitä apurakenteita algoritmi tarvitsee? *Simuloi* kuinka algoritmi etenee a)-kohdan verkossa.

6) Palaute (2p)

Anna palautetta kurssista *kotisivujen kautta löytyvällä kurssipalautelomakkeella!* Mikäli tämä tentti menee läpi on siihen mahdollista saada kaksi (2) lisäpistettä (antamalla palautetta ei voi kuitenkaan korottaa tentin arvosanaa nollasta ykköseen). Vastausaikaa on yksi viikko eli 22.5.2008 saakka. Palautetta ei kannata kirjoittaa tenttipaperiin, koska sieltä se ei välity asianmukaisesti eteenpäin.

```
public class TrakBigFloat implements TrakToken {

    private boolean negative; // True if negative
    private int[] mantissa; // Digits after the decimal point
    private int exponent;

    public TrakBigFloat() {
        this.negative = false;
        this.mantissa = new int[0];
        this.exponent = 0;
    }

    public TrakBigFloat(boolean negative, int[] mantissa, int exponent) {
        this.negative = negative;
        this.mantissa = mantissa;
        this.exponent = exponent;
    }

    private void makeMantissa(String s) {
        mantissa = new int[s.length()];
        for (int i = 0; i < s.length(); i++) {
            mantissa[i] = s.charAt(i) - '0';
        }
    }

    public TrakBigFloat multiply(TrakBigFloat other) {
        TrakBigFloat result = new TrakBigFloat();

        if (this.negative != other.negative) {
            result.negative = true;
        } else {
            result.negative = false;
        }

        result.mantissa = new int[this.mantissa.length + other.mantissa.length];
        result.exponent = this.exponent + other.exponent;

        for (int i = 0; i < this.mantissa.length; i++) {
            for (int j = 0; j < other.mantissa.length; j++) {

                int product = this.mantissa[this.mantissa.length - i - 1]
                    * other.mantissa[other.mantissa.length - j - 1];

                int resultIndex = result.mantissa.length - i - j - 1;
                result.mantissa[resultIndex] += product % 10;       // Add result
                result.mantissa[resultIndex - 1] += product / 10;     // Add carry

                if (result.mantissa[resultIndex] >= 10) {
                    result.mantissa[resultIndex] -= 10;
                    result.mantissa[resultIndex - 1]++;
                }
            }
        }

        result.normalize();

        return result;
    }


    private void normalize() {
        int offset = 0;

        for (; offset < mantissa.length && mantissa[offset] == 0; offset++) {
            exponent--;
        }

        int trailing = 0;
        for (int i = mantissa.length - 1; i > offset && mantissa[i] == 0; i--) {
            trailing++;
        }

        int[] newMantissa = new int[mantissa.length - offset - trailing];
        System.arraycopy(mantissa, offset, newMantissa, 0,
                         mantissa.length - offset - trailing);
        this.mantissa = newMantissa;
    }

    public String toString() {
        String output = "";

        if (mantissa.length == 0) {
            return "0";
        }

        if (negative) {
            output += "-";
        }

        if (exponent >= 0) {
            int i = 0;

            for (i = 0; i < exponent && i < mantissa.length; i++) {
                output += mantissa[i];
            }

            if (i < mantissa.length) {
                output += ".";

                for (; i < mantissa.length; i++) {
                    output += mantissa[i];
                }
            }

            for (; i < exponent; i++) {
                output += "0";
            }

        } else {
            output += ".";

            for (long i = 0; i > exponent; i--) {
                output += "0";
            }

            for (int i = 0; i < mantissa.length; i++) {
                output += mantissa[i];
            }
        }

        return output;
    }
}
```