

1. Testipeti etsintä- ja/tai lajittelualgoritmeille

Tässä harjoitustyössä tehdään valikko-ohjattu testipetiohjelma opetuksessa käsitellyille etsintä- ja lajittelualgoritmeille. Harjoitustyöstä saatava pistemäärä on jälleen työmääräpohjainen; mitä enemmän ominaisuuksia / toimintoja teet ohjelmaan sitä enemmän saat siitä pisteitä. Mitään jo tehtyjä ominaisuuksia ei pidä poistaa ohjelmasta siinä kohden kun teet siihen uutta toimintoa; vanhat toiminnot ovat myös ajettavissa milloin vain.

Tulemme käyttämään tämän harjoitustyön tekemiseen joulukuun opetusaikoja. Harjoitusten käsittelyosuudessa tarjotaan tähän harjoitustyöhön liittyvää konsultointia ja oppilaat tekevät sitä eteenpäin tuntien aikana. Vastaavasti muiden harjoitustehtävien tekeminen vähenee; tavallaan etsintä- ja lajittelualgoritmit teoriaa harjoitellaan tekemällä tätä harjoitustyötä. Annan tunneilla tehdystä työskentelystä harjoituspisteitä niitä tarvitseville.

Ohjelman ominaisuudet on ryhmitelty viiteen ryhmään, joista 1 ja 2 ryhmän ominaisuudet / toiminnot liittyvät etsintäalgoritmeihin ja ryhmien 3 – 5 ominaisuudet lajittelualgoritmeihin. Etsintäalgoritmeihin liittyvät toiminnot on tehtävä järjestyksessä 1 -> 2 (tämä on muutenkin mielekästä / järkevää) ja lajittelualgoritmeihin liittyvät toiminnot on tehtävä järjestyksessä 3 – 5. Etsintä- ja lajittelualgoritmitoiminnot ovat kuitenkin sen verran itsenäisiä että voit sivuuttaa etsintäalgoritmiominaisuudet ohjelmastasi (silloin voit tavoitella maksimissaan 3 pistettä työstä) tai vastaavasti lajittelualgoritmiominaisuudet (tavoittelet silloin maksimissaan kahta pistettä tästä harjoitustyöstä). Toki voit tehdä ihan kaikki ominaisuudet (jaossa max. viisi pistettä).

Alla on lueteltuna ohjelman toiminnot ryhmiteltyinä näihin viiteen ryhmään. Toiminnoista löytyy tarkempaa kuvausta listan alta.

Ryhmän numero	Ominaisuudet / toiminnot, joita ryhmän osalta pitää tehdä
1	Ohjelma suorittaa peräkkäisetsinnän (sequential_search) käyttäjän haluamasta aineistosta käyttäjän haluamalla avainarvolla. Etsinnän tulos tulostetaan näytölle. Yksityiskohdista kerrotaan tarkemmin alla.
2	Ohjelma suorittaa binäärietsinnän (jokin luennoilla esitetty binary -variantti) halutulle aineistolle halutulla avainarvolla. Etsinnän tulos tulostetaan näytölle. Näytölle tulostetaan myös suorituskysytietoja sekä binääri- että kohdan 1 peräkkäisetsinnästä; yksityiskohdat löytyvät alta.
3	Ohjelma suorittaa lisäyslajittelun (insertion_sort) sekaisin olevalle aineistolle. Yksityiskohdat löytyvät alta.
4	Ohjelma suorittaa valitun vaativuudeltaan $n * \log(n)$ olevan lajittelualgoritmin (mergesort, quicksort ovat tällaisia) sekaisin olevalle aineistolle. Yksityiskohdat löytyvät alta.
5	Ohjelmaan lisätään vielä yksi n^2 -tyyppinen lajittelualgoritmi ajettavaksi sekä yksi $n * \log(n)$ -tyyppinen lajittelualgoritmi ajettavaksi (jota ei tehty kohdassa 4). Ajotulosten lisäksi ohjelma tulostaa suorituskysyvertailun eri lajittelualgoritmeista. Yksityiskohdat selviävät alta.

1.1. Toimintoryhmän 1 vaatimuksista tarkemmin

Tässä vaihtoehdossa etsittävä aineisto generoidaan luentokalvosarjan tr7.pdf sivulla 9 esitetyllä tavalla; käyttäjä syöttää tässä ainoastaan aineiston koon. Esim. jos kooksi syötetään 10, generoidaan aineisto: 1, 3, 5, ... , 19. Etsittävä aineisto sisältää ainoastaan kokonaislukuja niin tässä kohtaan kuin muissakin etsintäalgoritmin soveltamiskohdissa. Samoin lajittelussa lajittelemme ainoastaan kokonaislukuja.

Etsinnän tuloksena ohjelma tulostaa onnistumistiedon ja jos etsittävä avainarvo löytyi myös sen paikan listassa.

Etsittävän avainarvon syöttämisen osalta on kaksi mahdollisuutta: käyttäjä syöttää sen suoraan; esimerkiksi 11:n, tai etsittävä avainarvo generoidaan satunnaislukuna kuten sivulla 9 kalvosarjassa tr7.pdf on kerrottu.

1.2. Toimintoryhmän 2 vaatimuksista tarkemmin

Binäärietsinnässä etsinnän kohteena oleva aineisto ja etsittävä avainarvo generoidaan samoin kuin kohdassa 1.1 on esitetty. Samoin tulosten esittäminen tehdään samoin kuin luvussa 1.1 on esitetty.

Jos käyttäjä valitsee tämän toimintoryhmän suorituskykyvertailutoiminnon ajetaan peräkkäisetsintä- ja binäärietsintäalgoritmi täsmälleen samalle aineistolle samoin hakuavaimin. Etsinnän aikana havainnoidaan vertailujen määriä sekä kulunutta seinäkelloaika; nämä molemmat tulostetaan molempien etsintäalgoritmien osalta. Suorituskykyvertailussa sekä etsittävä aineisto että etsittävä avain generoidaan satunnaisesti kalvosarjan tr7.pdf sivun 9 ohjeita noudattaen.

Jotta suorituskykyvertailun tulokset olisivat luotettavampia on testiajossa molempia etsintäalgoritmeja kutsuttava monta kertaa (esim. 10 kertaa) aina eri hakuavaimilla, jotta tulokset olisivat luotettavampia. Muutoin esim. binäärihaussa saatetaan arpoa satunnaisena avaimena aineiston keskeltä löytyvä arvo ja etsintä päättyy heti siihen jolloin tulokset vääristyvät. Käyttäjä voi syöttää myös toistojen määrän.

Lisäksi suorituskykyvertailu on vieläkin luotettavampaa, jos noin puolet hausta päättyvät onnistumiseen ja puolet ei. Kun generoimme hakuavaimen satunnaisesti tämä asia on kohtuullisen kunnossa.

Alla eräs ehdotelma suorituskykyajon tulostuksesta – tässä on ajettu lävitse kaikki opetuksessa läpikäydyt etsintäalgoritmivariantit; sinun riittää ajaa niistä lävitse vain peräkkäis- ja jokin binäärietsintä:

```
C:\WINDOWS\system32\cmd.exe
sequential_search:
Status: Successful
Elapsed per search: 0.2
Comparisons per search: 1391
Searches: 10
Status: Unsuccessful
Elapsed per search: 665.3
Comparisons per search: 5000001
Searches: 10
run_recursive_binary_1:
Status: Successful
Elapsed per search: 0
Comparisons per search: 24
Searches: 10
Status: Unsuccessful
Elapsed per search: 0
Comparisons per search: 24
Searches: 10
binary_search_1:
Status: Successful
Elapsed per search: 0
Comparisons per search: 24
Searches: 10
Status: Unsuccessful
Elapsed per search: 0
Comparisons per search: 24
Searches: 10
run_recursive_binary_2:
Status: Successful
Elapsed per search: 0
Comparisons per search: 39
Searches: 10
Status: Unsuccessful
Elapsed per search: 0
Comparisons per search: 44
Searches: 10
binary_search_2:
Status: Successful
Elapsed per search: 0
Comparisons per search: 39
Searches: 10
Status: Unsuccessful
Elapsed per search: 0
Comparisons per search: 44
Searches: 10
Press any key to continue . . .
```

1.3. Toimintoryhmän 3 vaatimuksista tarkemmin

Käyttäjä syöttää tässä lisäyslajittelun kohteena olevan aineiston koon; esimerkiksi syöte 1000 tarkoittaa sitä, että ohjelma generoi 1000 satunnaista int-arvoa taulukkoon. Esimerkkitilanteessa kunkin satunnaisluvun generoinnin rajat voisivat olla 0:sta 10 000:een, jotta samojen lukujen generointi aineistoon olisi kohtuullisen harvinaista. Siis kunkin yksittäisen arvottavan luvun mahdollinen arvoalue on kymmenkertainen verrattuna arvottujen lukujen määrään.

Ohjelma tulostaa jonkin pienen määrän lajiteltavan taulukon alkioita ennen lajittelua; esim. 200 kpl, ja sama alue taulukosta tulostetaan lajittelun jälkeen jotta voidaan varmistua, että algoritmi toimii oikein ainakin tämän alueen osalta. Alueen koko kysytään käyttäjältä; haluttaessa siis koko aineistokin tulostetaan ennen ja jälkeen lajittelun ja varmistetaan siten koko lajittelun onnistuminen.

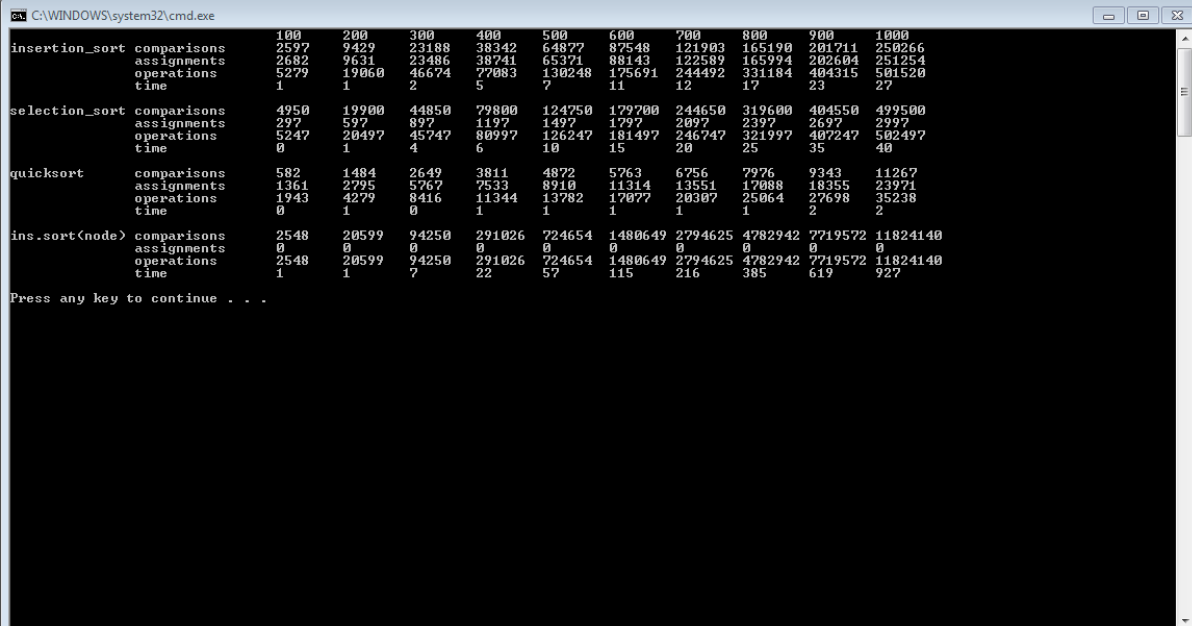
1.4. Toimintoryhmän 4 vaatimuksista tarkemmin

Syötteiden ja tulosten osalta tässä pätee samat vaatimukset kuin kohdassa 1.3. uudelle toteutetulle lajittelualgoritmile.

1.5. Toimintoryhmän 5 vaatimuksista tarkemmin

Tässäkin syötteet ja tulokset tehdyille uusille lajittelualgoritmeille ovat samanmuotoiset kuin kohdassa 1.3 on esitetty. Lisäksi käyttäjä valitsee tässä kumman lisätystä kahdesta lajittelualgoritmista ajetaan.

Suorituskykyvertailutoiminnossa voisi pyrkiä seuraavanlaiseen ajotulokseen (vain esimerkki; paremminkin asian saa tehdä!):

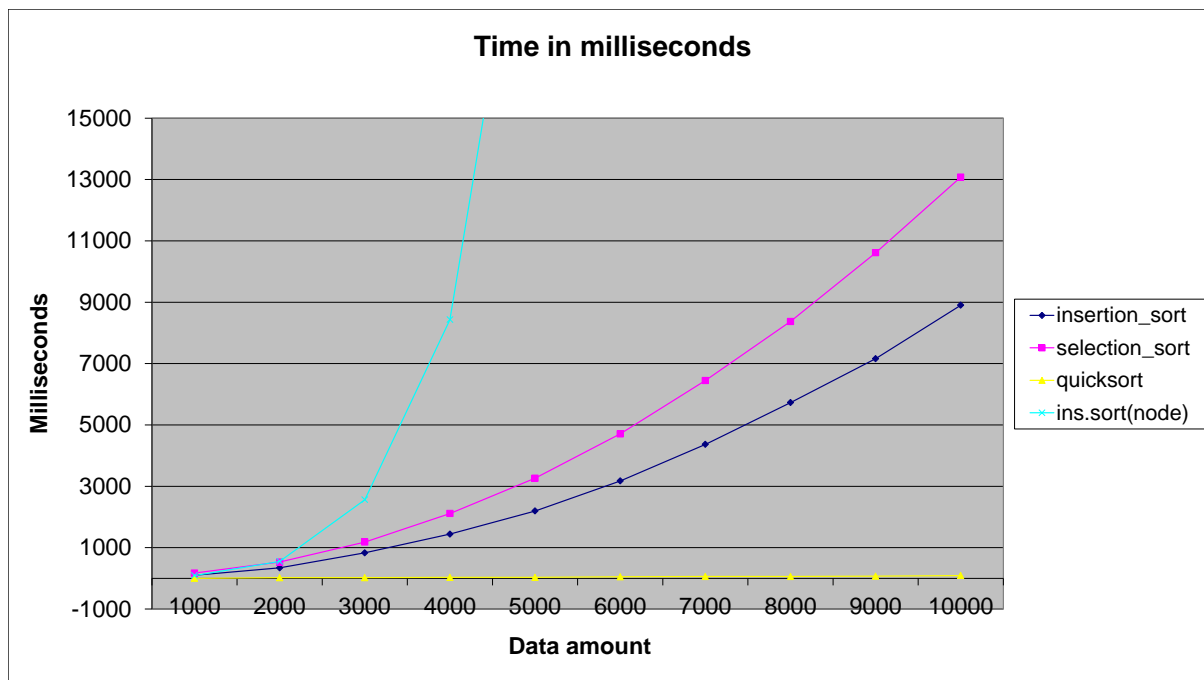


	100	200	300	400	500	600	700	800	900	1000
insertion_sort comparisons	2597	9429	23188	38342	64877	87548	121903	165190	201711	250266
insertion_sort assignments	2602	9631	23486	38741	65371	88143	122589	165994	202604	251254
insertion_sort operations	5279	19860	46674	77083	130248	175691	244492	331184	404315	501520
insertion_sort time	1	1	2	5	7	11	12	17	23	27
selection_sort comparisons	4950	19900	44850	79800	124250	179700	244650	319600	404550	499500
selection_sort assignments	297	597	897	1197	1497	1797	2097	2397	2697	2997
selection_sort operations	5247	20497	45747	80997	126247	181497	246747	321997	407247	502497
selection_sort time	0	1	4	6	10	15	20	25	35	40
quicksort comparisons	502	1484	2649	3811	4872	5763	6756	7976	9343	11267
quicksort assignments	1361	2795	5267	7533	8910	11314	13551	17088	18355	23971
quicksort operations	1943	4279	8416	11344	13782	17077	20307	25064	27698	35238
quicksort time	0	1	0	1	1	1	1	1	2	2
ins.sort(node) comparisons	2548	20599	94250	291026	724654	1480649	2794625	4782942	7719572	11824140
ins.sort(node) assignments	0	0	0	0	0	0	0	0	0	0
ins.sort(node) operations	2548	20599	94250	291026	724654	1480649	2794625	4782942	7719572	11824140
ins.sort(node) time	1	1	7	22	57	115	216	385	619	927

Press any key to continue . . .

, missä 1. rivillä kerrotaan lajiteltavan aineiston koot (näin monta int-arvoa lajiteltiin). operations-kohtaan on laskettu yhteen comparisons- ja assignments-arvot. time kerrotaan millisekunneina. Kyseisessä suorituskykytestissä ajettiin kolmea n^2 -tyyppistä lajittelualgoritmia; teidän työssä suorituskykyvertailussa näitä ajetaan vain kahta erilaista. Huom: jotta voit laskea sijoitusten (assignments) lukumäärää algoritmin ajon aikana sinun tulee ylimääritellä Key-luokan (ja samalla Record-luokan) sijoitusoperaattori; omassa sellaisessa pidetään kirjaa myös sijoitusten lukumäärästä.

Suorituskykyajon osalta tulee tuloksista vielä piirtää graafi seuraavan tyyliä (vain esimerkki):



Tässä on tavoitteena, että kuvan perusteella pystyt luokittelemaan lajittelualgoritmit paremmuusjärjestykseen suoritusajan tai jonkin toisen kriteerin perusteella omien testiesi pohjalta.

1.6. Valikko-ohjauksesta

Ohjelman käyttöliittymä on tekstimuotoinen (ajetaan cmd-ikkunassa) valikko-ohjattu liittymä, jossa päävalikosta (esitetään heti ohjelman käynnistyttyä) on mahdollisuus valita toimintoryhmät 1 – 5. Esimerkiksi valinnan 1 valittuana käyttäjä haluaa tehdä peräkkäisetsinnän ja ohjelma kysyy tarvittavat tiedot sen ajamiseksi. Alivalikot voit suunnitella itse mutta noudata selkeitä hierarkista tiedon syöttämistä ohjelmalle. Numeroi alivalikot kirjaimin: a), b), ... jne. Muut syötteet ohjelmalle annetaan kysymys-vastaus –tyylisillä dialogeilla; esim:

Anna aineiston koko:

, missä harmaalla taustavärillä on merkitty käyttäjän syöte.

1.7. Vaatimuksia palautukselle

- Palautus tehdään tabulaan kurssisivulla osoitettuun paikkaan pakattuna tiedostona, jonka nimessä pitää olla sisällytettynä ohjelman tekijän nimi; esim. TIRA_3_AlexanderStub.zip .
- Palautetaan joko koko ohjelmointiprojekti Microsoft Visual Studio 2013:n ymmärtämässä muodossa tai sitten tarvittavat lähdekoodit siten, että niistä saa toimivan ohjelman tekemällä pelkästään tyhjän Win32 console project –tyyppisen projektin, johon tiedostot liitetään ja saadaan siten käännettyä toimivaksi ohjelmaksi.
- Palautukseen (= pakattuun tiedostoon) liitetään mukaan myös README.DOCX –tiedosto (huom: Word-dokumentti), johon on dokumentoitu:
 - Ohjelman käännös-/linkkaus-/asennus-/ajo-ohje niin, että testaaja pääsee testaamaan ohjelmaa seuraamalla näitä ohjeita
 - Ohjelman jokaisen tason (1-5) yhteydessä ajetut testitapaukset, joilla ohjelman toiminnasta on varmistuttu. Käytännössä dokumentoitte annetun syötteen ja ohjelman tulostamat tulosteet. Näppärä tapa tähän on käyttää Windows 7:sta

löytyvää Snipping Tool –työkalua (Accessories-ryhmässä), jolla voit osoittaa näytöllä olevan ikkunan, ja sen sisältö menee leikepöydälle. Liitä mukaan dokumenttiisi näitä näytönkuvia ja kerro miten ne todistavat ohjelman oikeellisen toiminnan (tekstinä kuvan alla). Kuvassa olevia yksityiskohtia voit osoittaa nuolilla tms.

- Jos teit pistemäärään 5 vaadittavat toiminnot ohjelmaasi liitä suorituskkyvertailun ajotulokset ja niistä tehdyn graafisen diagrammin raporttiin.
- Mitä pistemäärää tavoittelet palautetulla harjoitustyöllä. Tämän tiedon pitää olla realistinen ja oikea; jos tavoittelet esim. viittä pistettä on ohjelmastasi löydyttävä ko. pistemäärään vaaditut toiminnot.
- Palauttajan yhteystiedot: sähköpostiosoite ja puhelinnumero
- Tuntikirjanpito työn tekemiseen kuluneesta ajasta tyyliin pvm, käytetty aika puolen tunnin tarkkuudella, mitä tuona aikana tehtiin. Tällaisia rivejä kannattaa kerätä esim. Excel-tiedostoon, jonka sisällön liitätte README.DOCX -tiedostoon.
- Selitä lyhyesti sanallisesti kussakin toimintoryhmässä käyttämäsi suunnitteluratkaisu. Esimerkiksi miten hoidat aineiston generoinnin, lajittelualgoritmin ajon, tulosten generoinnin jne.
- Sisällytä liitteeseen kaikki tekemäsi lähdetekstitiedostot järjestyksessä ensin .h-loppuiset ja sitten .cpp-loppuiset. Näiden ryhmien sisällä järjestä tiedostosisällöt niiden nimien mukaan aakkosjärjestykseen. Noudata seuraavaa formaattia:

```
...

/*
 * List.h
 * =====
 */

< ja tähän liität sitten List.h:n olemassaolevan sisällön >

/*
 * Node.h
 * =====
 */

< ja tähän liität sitten Node.h:n olemassaolevan sisällön jne. >

...
```

- Huomioitavaa: ohjelman lähdekoodien tulee kääntyä Microsoft Visual Studio 2013 – kääntäjällä ilman virheilmoituksia ja varoituksia. Käännöstuloksena syntyvän ohjelman käyttöliittymän tulee olla raportissa kuvatun kaltainen (ei raportoida toisenlaista käyttöliittymää kuin millaisen ohjelma oikeasti tuottaa!). Ylimääräiset huomautus- ja korjauskierrokset johtavat siihen, että ohjelmistotoimittajan on maksettava viivästyssakkoja asiakkaalle (käytännössä harjoitustyöstä saatava pistemäärä putoaa)!

README.DOCX-tiedoston ulkoasun on oltava TAMKin kirjallisten töiden raportointiohjeen mukainen (löydät tämän intran hakutoiminnolla; tähän on tehty valmis Word-template).

Huom! harjoitustyöratkaisun kopioiminen on kielletty ja tämä tullaan tarkastamaan!