Toteutusdokumentti

Muutokset tehtävämäärittelyyn

Tehtävämäärittelyn mukaan tarkoituksena oli toteuttaa kolme binääristä hakupuuta tarvittavine tietorakenteineen, mutta lopulliseen toteutukseen sisältyy edellä mainittujen lisäksi myös yksinkertainen trie-puun toteutus. Lisäksi olen toteuttanut binääristen puiden alkioiden läpikäynnin määrittelydokumentista poiketen myös esijärjestyksessä.

Rakennekuvaus

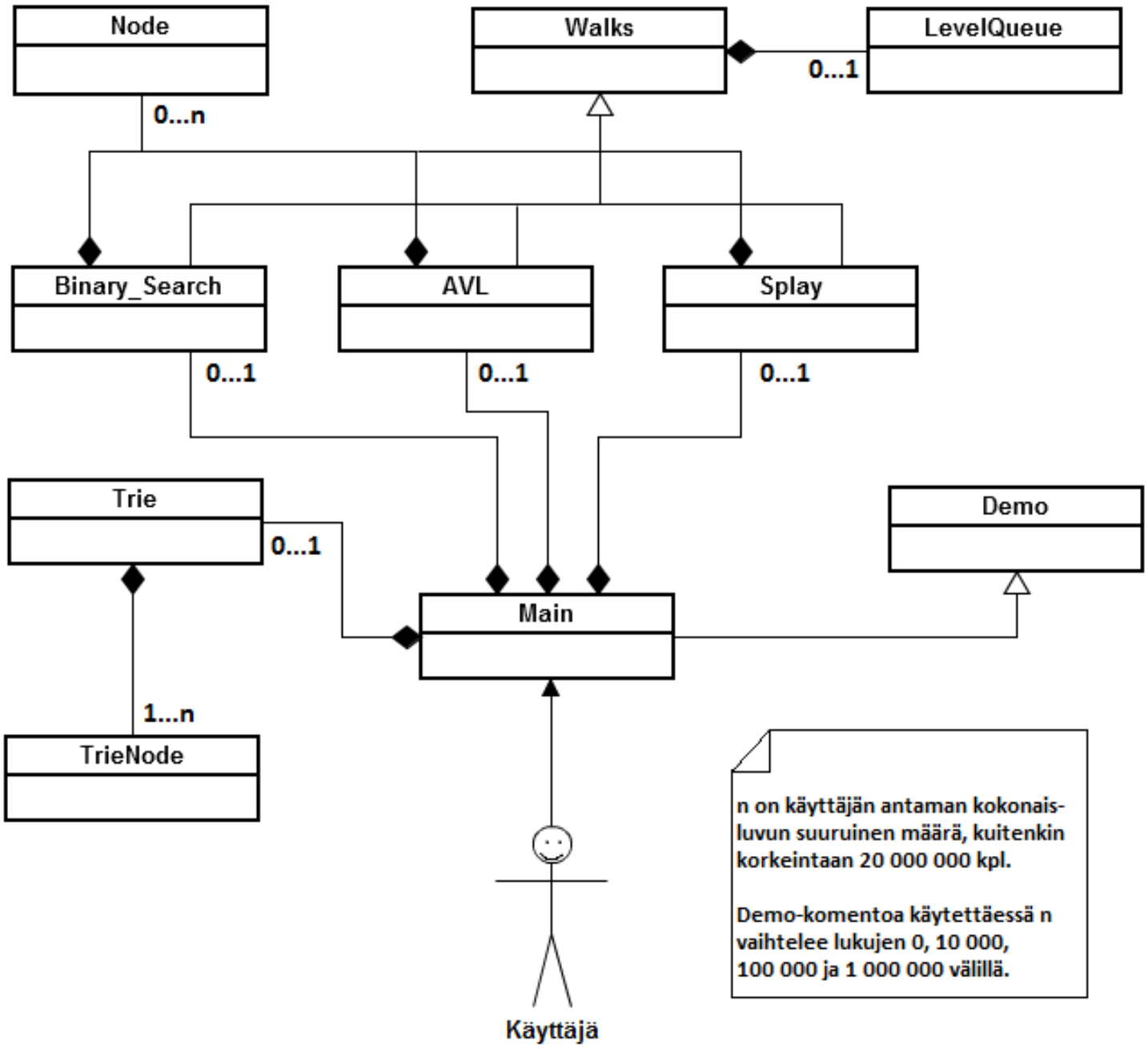
Kun käyttäjä on käynnistänyt pääohjelman, hän voi pyytää ohjelmaa luomaan binäärihaku-, AVL-, trie- tai splay-puun. Tällöin Main luo valitusta puusta uuden puuolion. Trie-puun tapauksessa Trie-luokka luo samalla itselleen tyhjän juuren luomalla ilmentymän TrieNode-luokasta. Binäärihakupuu, AVL-puu ja splay-puu luovat ensimmäiset solmuolionsa vasta solmujenlisäysvaiheessa.

Kun puuolio on luotu, ohjelma tekee siihen käyttäjän antaman määrän verran lisäyksiä, hakuja sekä poistoja. Kukin lisäys luo uuden ilmenytmän Node- tai TrieNode-luokasta.

Käyttäjä voi valita käyttöliittymässä tietyn puun lisäksi demo-komennon, joka esittelee eri puita ja niiden ominaisuuksia. Demo-luokka luo Mainin tapaan ilmentymiä puista, jotka taas luovat ilmentymiä Nodesta ja TrieNodesta. Lisäksi Demo käyttää Walks-luokan puunalkiontulostusmetodeja, joista leveyssuuntainen tulostus luo ilmentymän LevelQueue-luokasta.

LevelQueue on yksinkertaistettu jonoluokka. Se tallentaa jonoon erityisiä QueueNode-solmuja, jotka sisältävät ilmentymän yhdestä tai kahdesta Node-solmusta. QueueNode on LevelQueuen sisäinen yksityinen luokka, josta LevelQueue tekee ilmentymiä enqueue-toiminnon yhteydessä, kun jonoon lisätään uusi QueueNode-solmu.

Luokkakaavio



Yksityiskohtaisemmat UML-kaaviot ovat JavaDocissa (Forest\Forest\dist\javadoc\index.html).

Aika- ja tilavaativuudet

Binäärihakupuu

Binääripuu on toteutettu kevään 2012 Tietorakenteet-kurssin luentomateriaalin pseudokoodin mukaan. Niinpä on turvallista sanoa, että harjoitustyön aika- ja tilavaativuudet vastaavat luentokalvojen toteutuksen vaativuuksia. Search toteutettiin ei-rekursiivisen pseudokoodin mukaan.

AVL-puu

Myös AVL-puu on toteutettu luentomateriaalin pseudokoodin mukaan, joten senkin aika- ja tilavaativuudet vastaavat luentokalvojen toteutuksen vaativuuksia. Deletessä ja searchissa käytetään ei-rekursiivista hakua.

Splay-puu

Splay-puu on toteutettu samalla tavalla kuin AVL-puu, joskin toteutuksesta on poistettu joitakin splay-puulle turhia metodeja ja koodipätkiä. Lisänä splay-puussa on Sotirios Stergiopoulosin pseudokoodin pohjalta tehty splay-metodi, jota käytetään aina insertin, searchin ja deleten yhteydessä. Splayaaminen ei kuitenkaan nosta toteutuksen O-aikavaativuuksia AVL-puuhun nähden, paitsi operaatioiden pahimmissa tapauksissa. Deletessä ja searchissa käytetään ei-rekursiivista hakua.

Trie-puu

Harjoitustyön trie-puuta ei ole toteutettu minkään varsinaisen pseudokoodin pohjalta. Olen lukenut puun toimintaidean Wikipedia-artikkelista ja toteuttanut puun tämän pohjalta. Trie-puun ja sen operaatioiden tilavaativuuksiin vaikuttavat syötteiden määrä ja niiden merkkipituus. Yksittäisiin lisäämis-, haku- tai poisto-operaatioiden aikavaativuuksiin vaikuttaa kuitenkin ainoastaan kyseessä olevan arvon merkkipituus, sillä puun jokainen solmu vastaa yhtä käsiteltävän arvon merkkiä. Niinpä trien vaativuuksissa n tarkoittaa käsiteltävän arvon merkkien lukumäärää.

Puu	**Hakuaika (kesk.)**	**Hakuaika (pahin)**	**Tila (kesk.)**	**Tila (pahin)**
binääri	O(log n)	O(n)	O(1)	O(1)
AVL	O(log n)	O(log n)	O(1)	O(1)
splay	O(log n)	O(n)	O(1)	O(1)
trie	O(n)	O(n)	O(1)	O(1)

Puu	**Lisäysaika (kesk.)**	**Lisäysaika (pah.)**	**Tila (kesk.)**	**Tila (pahin)**
binääri	O(log n)	O(n)	O(1)	O(1)
AVL	O(log n)	O(log n)	O(1)	O(1)
splay	O(log n)	O(n)	O(1)	O(1)
trie	O(n)	O(n)	O(1)	O(1)

Puu	**Poistoaika (kesk.)**	**Poistoaika (pah.)**	**Tila (kesk.)**	**Tila (pahin)**
binääri	O(log n)	O(n)	O(1)	O(1)
AVL	O(log n)	O(log n)	O(1)	O(1)
splay	O(log n)	O(n)	O(1)	O(1)
trie	O(n)	O(2n)	O(1)	O(1)

Suorituskyky ja puiden vertailu

Ohjelman demotulostus, jonka tulokset vaikuttavat aiempiin tulostuksiin nähden melko tavanomaisilta (min & max poistettu tulosteesta):

INSERTS
Binary search, insert 10 000 times: 5 ms.
Binary search, insert 100 000 times: 33 ms.
Binary search, insert 1 000 000 times: 482 ms.
AVL, insert 10 000 times: 22 ms.
AVL, insert 100 000 times: 80 ms.
AVL, insert 1 000 000 times: 783 ms.
Splay, insert 10 000 times: 25 ms.
Splay, insert 100 000 times: 45 ms.
Splay, insert 1 000 000 times: 946 ms.
Trie, insert 10 000 times: 30 ms.
Trie, insert 100 000 times: 92 ms.
Trie, insert 1 000 000 times: 1513 ms.

SEARCHES
Binary search, 10 000 searches: 3 ms.
Binary search, 100 000 searches: 6 ms.
Binary search, 1 000 000 searches: 52 ms.
AVL, 10 000 searches: 4 ms.
AVL, 100 000 searches: 5 ms.
AVL, 1 000 000 searches: 39 ms.
Splay, 10 000 searches: 5 ms.
Splay, 100 000 searches: 3 ms.
Splay, 1 000 000 searches: 16 ms.
Trie, 10 000 searches: 11 ms.
Trie, 100 000 searches: 30 ms.
Trie, 1 000 000 searches: 275 ms.

DELETES FROM TREES WITH ~1 000 000 NODES
Binary search, 10 000 deletes: 6 ms.
Binary search, 100 000 deletes: 51 ms.
Binary search, 1 000 000 deletes: 488 ms.
AVL, 10 000 deletes: 9 ms.
AVL, 100 000 deletes: 47 ms.
AVL, 1 000 000 deletes: 460 ms.
Splay, 10 000 deletes: 12 ms.
Splay, 100 000 deletes: 101 ms.
Splay, 1 000 000 deletes: 1034 ms.
Trie, 10 000 deletes: 6 ms.
Trie, 100 000 deletes: 47 ms.
Trie, 1 000 000 deletes: 460 ms.

Binäärihakupuu

Binäärihakupuu on nopea kaikissa operaatioissa sattumanvaraisilla syötteillä, jotka tulevat sattumanvaraisessa järjestyksessä (tästä eteenpäin ”demosyötteillä”). Testausdokumentissa olevasta kuva 1:stä nähdään lisäksi, että puu toimii nopeasti sattumanvaraisilla syötteillä, vaikka operaatioita tehtäisiin 20 000 000 kertaa peräkkäin. Puu pysyy melko hyvin tasapainoisena tällaisilla syötteillä, vaikka sillä ei olekaan tasapainottavia operaatioita. Binäärihakupuun operaatioiden yksinkertaisuus näkyykin nopeassa suoritusajassa.

Binäärihakupuu ei kuitenkaan ole paras vaihtoehto, jos annettava syöte on järjestyksessä, minkä näkee selvästi testausdokumentin kuva 2:sta. Puusta tulee lineaarinen, jolloin sekä lisäys- että hakuoperaatiot ovat hitaita. Myös poistot vievät huomattavan paljon aikaa, jos ne tehdään lehtisolmujen lähellä.

AVL-puu

Myös AVL-puu on nopea demosyötteiden tapauksessa. Kuvasta 1 nähdään, että AVL-puut toimivat hyvin, lähes binäärihakupuiden nopeudella, vaikka puuhun syötetään suuriakin määriä sattumanvaraisia lukuja. AVL-puu on kuitenkin selvästi ylivertainen binäärihakupuuhun verrattuna tapauksessa, jossa puuhun lisätään suuri määrä järjestyksessä olevia lukuja. Toisin kuin binäärihakupuu, AVL-puu ei muutu koskaan lineaariseksi tasapainottuvuutensa ansiosta, eivätkä siten järjestyksessä tulevat syötteet hidasta puun toimintaa samassa määrin.

Splay-puu

AVL-puun tapaan splay-puu toimii nopeasti demosyötteillä. Kuvasta 1 huomataan kuitenkin, että puu on varsin hidas suurella määrällä sattumanvaraisia ei-toistuvia arvoja. Tämä johtuu puun splay-operaatiosta, joka suoritetaan jokaisen operaation yhteydessä. Puu onkin tarkoitettu sellaiseen käyttöön, jossa joitain solmuja käytetään selvästi useammin kuin muita.

Kun splay-puuhun lisätään järjestyksessä olevia lukuja, puu toimii splayaamisen ansiosta nopeasti, vaikka siitä tuleekin lineaarinen (kuva 2). Kun lineaarisen puun lehtisolmuun tai näiden läheisyydessä oleviin solmuihin tehdäänkin hakuja tai poistoja, puu ei juurikaan hidastu, sillä siitä muodostuu splayaamisen vuoksi kohtuullisen tasapainoinen muutaman haun tai poiston jälkeen. Niinpä splay-puukin päihittää binäärihakupuun tapauksessa, jossa puuta käytetään suurella määrällä järjestyksessä olevia lukuja.

Trie-puu

Toteutuksen trie-puu toimii kohtuullisen nopeasti suurellakin määrällä syötteillä, jotka eivät sisällä paljon merkkejä. Jos puuhun kuitenkin viedään paljon pitkiä lukuja, puun viemä tila kasvaa nopeasti niin isoksi, ettei puun ylläpitämiseen riitä muistia. Trie-puita on vaikea vertailla muiden harjoitustyön puurakenteiden kanssa, sillä niiden käyttötarkoitus ja tässä toteutettu rakenne poikkeavat työn muiden puiden vastaavista suuresti.

Parannusehdotukset ja puutteet

Binäärihakupuu, AVL-puu ja splay-puu käyttävät samaa Node-solmuluokkaa. Node-luokan ilmentymillä on height-attribuutti, jota käyttää ainoastaan AVL-puun toteutus. Tilan säästämiseksi olisi puiden vaativammassa käytössä hyvä erottaa binäärihakupuulle ja splay-puulle oma solmuluokkansa, jossa ei ole turhaa korkeusattribuuttia.

Koska alkujaan harjoitustyön oli tarkoitus sisältää vain kolme puuta, neljännen puun, trien, toteutus jäi jokseenkin puutteelliseksi: Trie-puun jokaisella solmulla on lopullisessa toteutuksessa kymmenen pituinen TrieNode-lista. Vaativammassa käytössä tilan säästämiseksi trie olisi kuitenkin kannattanut toteuttaa niin, ettei puun lehtisolmuilla ole kyseistä listaa, ennen kuin niiden alapuolelle halutaan lisätään uusi solmu. Lisäksi trie-puulla ei ole min- ja max-ominaisuuksia, eikä siihen talletettujen lukujen tulostusmahdollisuutta.

Puiden testaaminen järjestyksessä olevilla syötteillä on jokseenkin rajallinen. Ei esimerkiksi ole mahdollista lisätä puuhun järjestyksessä olevia arvoja, mutta etsiä siitä sen jälkeen sattumanvaraisia lukuja. Toteutuksessa ei myöskään ole mahdollista tehdä operaatioita syötteillä, joissa esiintyy joitakin lukuja selvästi useammin kuin toisia. Tämänkaltaiset testausmahdollisuudet olisivat omiaan puiden vertailuun worst case -skenaarioissa ja toisaalta splay-puun tehokkuuden osoittamiseen tilanteissa, joissa puista käytetään toistuvasti joitain tiettyjä solmuja.

Myöskään trie-puun toteutuksen tapauksen, jossa syötteet ovat järjestyksessä, vertaaminen tapaukseen, jossa syötteet ovat sattumanvaraisia ja sattumanvaraisessa järjestyksessä, ei anna kovin totuudenmukaista kuvaa puun toiminnasta. Sattumanvaraisten syötteiden merkkimäärä on todennäköisesti huomattavasti suurempi kuin järjestyksessä olevilla syötteillä, joiden esimerkiksi ensimmäiset tuhat arvoa ovat korkeintaan kolmemerkkisiä. Trie-puun operaatioiden nopeus kuitenkin riippuu täysin tallennettavan, haettavan tai poistettavan arvon merkkimäärästä.

Lähteet

Binäärihakupuu

Wikipedia-artikkeli binäärihakupuusta

Tietorakenteet-kurssin luentokalvot (Patrik Floréen, kevät 2012)

AVL

Wikipedia-artikkeli AVL-puista

Tietorakenteet-kurssin luentokalvot (Patrik Floréen, kevät 2012)

Splay

Wikipedia-artikkeli splay-puista

Stephen J. Allanin materiaali splay-puista

Splay-metodin pseudokoodi (Sotirios Stergiopoulos, Department od Computer Science, York University 2001)

Trie

Wikipedia-artikkeli trie-puista

(Linkkien materiaaleja on käytetty syys- ja lokakuussa 2012.)