Obligatorisk Oppgave 3

Algoritmer og Datastrukturer ITF20006

Lars Vidar Magnusson

Frist 27.03.15

Den tredje obligatoriske oppgaven tar for seg forelesning 9 til 13, som dreier seg om datastrukturer.

Praktisk Informasjon

Det er med disse oppgavene dere skal få den nødvendige erfaringen dere trenger, så det kreves at alle innleveringer skal være individuelt arbeid. Det er selvsagt til lov å hjelpe hverandre, men den endelige besvarelsen skal være et resultat av eget arbeid. Besvarelser som viser tegn plagiering vil bli underkjente.

Dere kan selv velge i hvilket språk dere implementerer algoritmene, men hjelpefunksjoner vil bli gitt i Java. Hvis dere skulle trenge direkte hjelp med koden, kan det være greit å holde seg til enten Java eller C#.

Dere skal lage en enkel rapport som beskriver besvarelsen og samler alle tekstlige svar. Rapporten skal struktureres i henhold til oppgaven. Koden som løser programmeringsoppgavene skal være kjørbar og skal ta en inputfil som argument fra kommandolinjen, for så å skrive ut løsningen. Koden skal leveres i ren kildekode (ikke et prosjekt fra en eller annen IDE med en masse unødvendige filer), og det skal legges ved et enkelt skript som kompilerer koden.

Både rapport, kode og skript skal pakkes i en tarball og skal sendes til meg på lars.v.magnusson@hiof.no med emne "ITF20006 $OPPGAVE $STUDENT".

Oppgave 1 - Stakker og Køer

A - FIFO vs. LIFO

Forklar forskjellen mellom FIFO og LIFO og hvordan begrepene er relatert til stakker og køer.

B - Representasjon

Man kan implementere en stakk og en kø med forskjellige underliggende strukturer. Forklar kort hvordan man ville implementert begge disse med både en

direkteadressert array og en lenket liste. Hvordan ville implementasjonene skilt seg fra hverandre?

Oppgave 2 - Dictionaries

Vi har i forelesningene presentert en rekke datastrukturer som implementerer dictionary grensesnittet/operasjonene INSERT, DELETE og SEARCH.

A - Beskrivelse

Gi en beskrivelse av de ulike datastrukturene med fokus på kjøretiden til de ulike operasjonene.

B - Applikasjonsområder

Diskuter i hvilke situasjoner de ulike datastrukturene passer med fokus på fordelene hver datastruktur tilbyr i forhold til de andre. Bruk gjerne konkrete applikasjoner/applikasjonstyper som eksempel.

Oppgave 3 - Implementasjon av Søketrær

I denne oppgaven skal dere implementere to av søketrærne vi har presentert i forelesningene, og dere skal sammenligne resultatet. Innleveringen skal bruke *ArrayIO* klassen for innlesing av data. Resultatene skal skrives ut som en mellomromseparert liste ved hjelp av en inorder traversering av trærne.

Det skal være mulig å kjøre programmene på to måter. Den første måten skal ta en inputfil med en liste som argument. Listen skal settes inn i et tre fra venstre mot høyre og resutatet skal skrives ut. Den andre måten skal ta to inputfiler med lister. Den første listen skal brukes til å bygge opp et tre, den andre skal brukes til å angi hvilke elementer som skal slettes fra treet. Resultatet etter slettingen skal skrives ut.

A - Binære Søketrær

Implementer binære søketrær i henhold til pseudokoden gitt i forelesningen. Det holder at dere implementerer dictionary operasjonene INSERT, DELETE og SEARCH.

B - Rød-Svarte Trær

Implementer rød-svarte trær i henhold til pseudokoden gitt i forelesningen. Det holder også her at dere implementerer dictionary operasjonene INSERT, DELETE og SEARCH.

C - Sammenligning

I denne oppgaven skal de to ulike implementasjonene for et søketre testes mot hverandre. Den faktiske kjøretiden for de ulike operasjonene skal måles for

ulike input og inputstørrelser. Alle de tre operasjonene må testes grundig. Resultatene skal diskuteres og presenteres i en tabell og/eller en graf. Diskusjonen bør trekke inn kjøretidsanalysen gitt i forelesningene.

Lykke til!