TENTAMEN 1DV507

Inlämnad av: Fredrik Sandberg

Id: fs222mh

Datum: 2016-03-18

UPPGIFT 2(A)

The class *java.lang.Object*

Alla klasser ärver från java.lang.Object.
Den klassen innehåller olika metoder som kan appliceras på olika objekt.
Exempel som toString(), equals() etc .Dessa metoder kan med fördel ”overridas” när man implementerar en klass för att undvika dess default med @Override (att föredra), eller exakt samma sträng.

Method Overloading

Tillåter en klass att ha flera metoder med samma namn, om argumenten är olika.

Exempel.

Metod 1: public void show(string s)
Metod 2: public void show(strings s, int n)

Dessa kallas med olika argument och första metoden blir ”overloadad” eftersom andra metoden har fler argument alt. olika typer av parametrar, t.ex. int istället för string.

Polymorphic call

Man kallar på ett objekt som kan ha många olika klasser med denna metod. Detta är när man kan ha många klasser som ärver varandra. Om man har en en klass som overridar sin superklass så är det den metoden som gäller, annars går man till den klassen som ligger högre i hierarkin, dvs om metoden finns i den klass man kallar på som exekveras den, annars går man ett steg upp i hierarkin och ser om den klassen innehåller rätt metod.

UPPGIFT 2(B)

Para ihop:

T1:O(n log n)

Är ”nästan” linjär men har en logaritm på n. Därför sätter jag O(n log n) på denna.

T2: O(n)

Denna är linjär och bör då således vara O(n)

T3:O(n²)

Exponentiell och bör då vara $O(n^2)$.

T4: O(1)

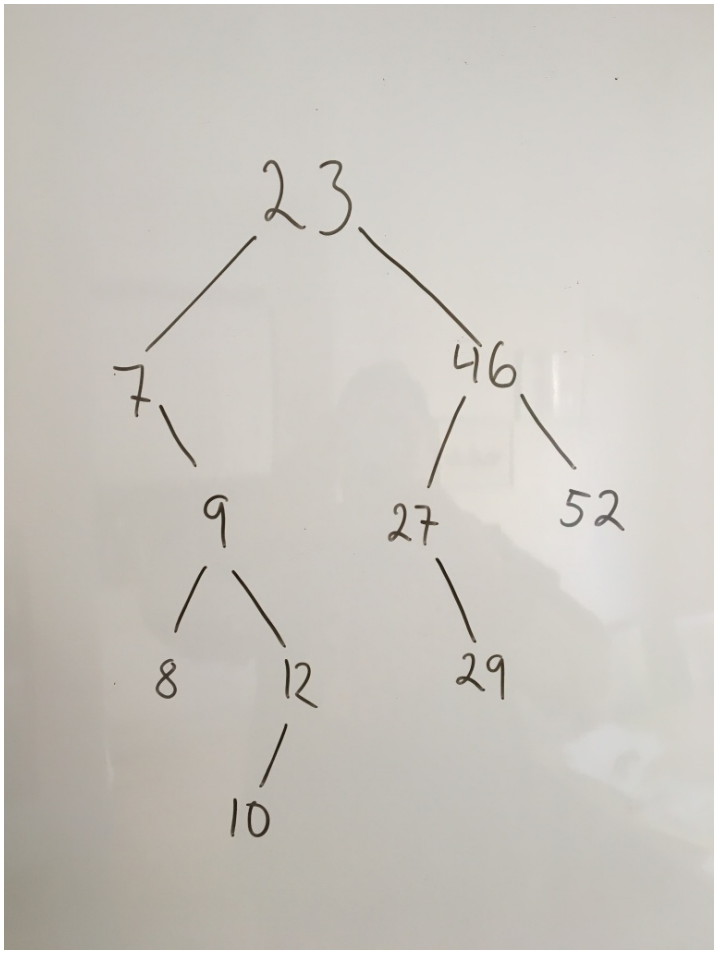
Denna är konstant och bör då således vara O(1)

Rankning:

1. O(1)
2. O(n)
3. O(n log n)
4. $O(n^2)$

UPPGIFT 3(A)

3A.1 Search tree



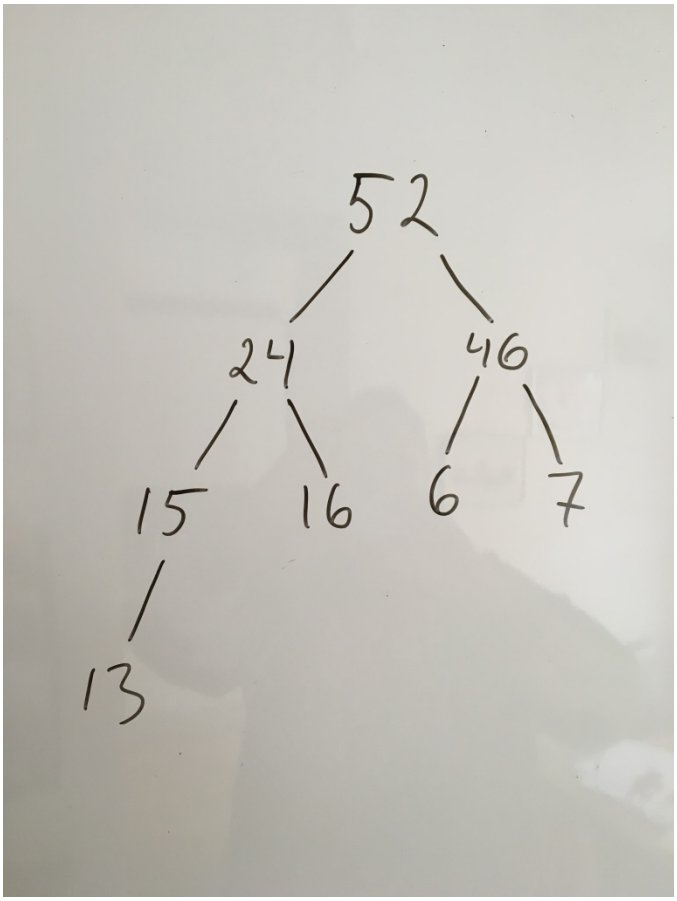
3A.2

Den printer värdet när den når noden, sen ser det efter rightchild. Vi går högervarv här. Finns inga högernode går senaste åt vänster istället och utför samma.

23,46,52,27,29,7,9,12,10,8

UPPGIFT 3(B)

3B.1 Binary heap tree



3B.2 Array

empty, 52, 24, 46, 15, 16, 6, 7, 13

0 används inte och man lägger till lager för lager.

3B.3 Heap after largest removed

Man flytta upp den node som är sist och när den är root så jämför man värdet på dess children och byter plats med den som har högst värde, sedan fortsätter man och jämför till den hamnar rätt. I detta fall byter 13 bara plats med 46.