Opbygningen

Opgaven lyder på at bygge en suffix træ, vha. en implementation af McCreight algoritmen.

Et suffixtræ er en datastruktur som tillader en udtømmende søgning i lineær tid, tid propotional med længden af input-teksten som der søges i. Det suffixtræ vi bygger, skal alle de steder en given delstren optræder i input-teksten.

Vi valgte at vi ville implementere i java-1.5 da det er det sprog hvis funktionsbibliotek vi kender bedst. Vi indser at vi mister noget effektivitet ifht. en implementation i C eller C++, men først og fremmest må det gælde om at få en implementation som fungerer.

Implementationen

I førse omgang forsøgte vi en naiv implementation, hvor vi uden brug af et suffixtræ som datastruktur, laver et Scanner objekt og læser input ind dertil. Den har vi til at give korrekte indexes som resultat, til System.out og til en output fil <input-filnavn>.out

Men det som opgaven egentlig lyder på er at implementere et suffixtræ som datastruktur, og derefter søge i det vha. McC-algoritmen.

Vi gør her det, at vi laver et suffixtræ med createSuffixTree, hvor vi angiver en root og påbegynder en traversering med slowScan. Og da vi ikke har fået fastScan til at køre, har vi slowScan til at gennemløbe hele input. Den fungerer ved at den tager førnævnte root, og med den som rod laver den en træstruktur. Hvis en struktur allerede findes, tjekker den om strengen allerede findes. Hvis dette ikke er tilfældet, laver den en ny knude med en kant nedtil denne med den del af strengen der ikke matchede. Den tjekker blot naivt tegn for tegn, og er derfor langsom.

Koden

Vores kode findes i biblioteket */users/lobner/StrAlg/* på daimi, og køres med flg. kommando:

./search <filnavn> <delstreng>

Den outputter derefter, afhængig af valg i scriptet, som nævnt enten de eksisterende indexes til System.out og en outputfil med den naive implementation. Ellers skriver den en Suffix-struktur til System.out, men finder desværre ikke indexes heri, da vi ikke kunne få det til at fungere.