Tema 8: Algoritmdesigntekniker

Henrik Järnbrand `heja4977`
Tomas Sandberg `tosa3589`

5 mars 2015

1 Frågeförslag

1.1 Förslag 1

Vilka fem algoritmdesigntekniker tar boken upp? Förklara vad varje teknik innebär, nämn en algoritm som använder denna teknik samt ett exempel på algoritmens användningsområde. För ett högre betyg ska du kunna diskutera för- respektive nackdelar med varje designteknik i olika situationer.

1.2 Förslag 2

Vad innebär dynamisk programmering (dynamic programming)? Nämn ett problem som är lämpligt att lösa med dynamisk programmering och beskriv lösningen.

1.3 Förslag 3

Till vilken kategori av algoritmdesigntekniker tillhör datastrukturen Skip List? Vad är det som gör att den tillhör just denna kategori? Förklara slutligen hur Skip List fungerar, för ett högre betyg ska du även kunna diskutera fördelar och nackdelar med denna implementation.

2 Implementationsförklaring – Huffman

Programmet startas med tre argument. Det första är en flagga, antingen encode eller decode beroende på om man vill komprimera eller dekomprimera. De andra två argumenten är två filnamn, en inputfil från vilken innehållet läses och en outputfil till vilken output skrivs. Mall för användning nedan:

Huffman -encode [input file] [output file]

> Komprimerar inputfilen och sparar den komprimerade filen som outputfilen.

Huffman -decode [input file] [output file]

> Dekomprimerar inputfilen och sparar den dekomprimerade filen som outputfilen.

Programmet är tänkt att komprimera text. Texten läses in från inputfilen och komprimeras med Huffmans algoritm av ett HuffmanCoderobjekt. Det Huffmanträd som då skapas och de

bytes som representerar texten sparas till outputfilen. HuffmanCoder tar in texten och räknar hur ofta varje tecken förekommer. Därefter skapas noder för varje tecken, antalet förekomster av ett tecken blir nodens vikt. Varje nod läggs i en prioritetskö som används för att bygga trädet, noderna med högst vikt placeras högst upp i trädet och de med lägst vikt hamnar längst ner. En djupetförstsökning traverserar sedan trädet och sparar samtliga teckens binära representation i listor innehållande heltal. Efter det komprimeras texten genom att skapa en bitrepresentation för texten som sparas i en bytearray. Komprimerings-metoden tittar på alla tecken och använder heltalsrepresentationen för att avgöra vilket värde varje bit ska ha. Slutligen returneras bytearrayen.

Programmet är även tänkt att dekomprimera bytes som har komprimerats med komprimeringen ovan. Först läses Huffmanträdet och de komprimerade bytesen in. Därefter skickas de vidare till ett HuffmanDecoderobjekt. Detta objekt använder sedan trädet för att dekryptera bytesen, så de åter omvandlas till text. Dekrypteringen fungerar så att den går igenom alla bytes och sedan maskar ut en bit i taget från en byte i taget. För varje utmaskad bit tas ett steg i Huffmanträdet. När ett löv nås sparas det tecken som är lagrat i noden och promenaden börjar åter från roten. Så länge det finns olästa bytes och det finns tecken kvar att läsa så fortsätter maskandet. Texten sparas sedan på outputfilen.