DM507 Forår 2019 Projekt Del III

Antal beståede: 154

Antal ikke-beståede: 6

Generelle fejl:

Fejl ID	*Beskrivelse*
1	Afleveringen på papir og på Blackboard er ikke den samme.
2	En afleveret fil er gemt som “utf8 med BOM”. Filer skal gemmes som standard utf-8 (uden BOM), da Java kræver at der ingen BOM er, og ikke kan kompilere “utf med BOM”. Desværre vil Notepad under Windows bruge BOM når der gemmes i formatet utf8. Løsning: vælg kodning “ANSI” under “Save As” i Notepad.
3	Man bruger ikke zip format til at pakke filerne sammen med (man bruger f.eks. rar i stedet). Hvis man bruger WinRAR kan man vælge i programmet at pakke til zip format i stedet for rar format.
4	Bruger kommentarer, som ikke beskriver, hvad koden reelt gør (f.eks. skriver man, at orderedTraversal *sorterer træet*, men i virkeligheden går den gennem træet in-order og skriver nøglerne i sorteret rækkefølge i et array). Kommentarer er rigtigt gode at have i kode, men hvis de er misvisende kan de også gøre det rigtigt svært for en læser at følge med i, hvad koden gør (og ræsonnere omkring korrektheden).
5	Ingen kommentarer eller kommentarer, som kun skriver det indlysende ned.
6	Decode tager ikke højde for hvor mange karaktere der skal være i filen.

	Fx. Input-fil: aabc Efter encode: 0 0 10 111 0 Efter decode: aabca Det sidste 0 (a) kommer fordi der skal skrives en hel byte ned og ikke kun 7 bites. Dvs i decode skal der kun decodes de første 7 bits og ikke den sidste. I opgavebeskrivelsen er dette beskrevet, dvs. opgavebeskrivelsen er ikke læst ordentlig.
7	Blade skal indeholde en int, som repræsenterer den byte, bladet står for. Hyppigheden for et blad (og for træer generelt) skal være key i det Element, som indeholder bladet (træet). Når Huffmans algoritme kører, skal hyppigheder summes sammen. Nogle laver den fejl at summe ints i blade sammen og lade disse være hyppigheder, når man sætter træer sammen. Dvs. at de bruger byteværdier som hyppigheder. Det svarer til at lave et Huffman træ over hyppighederne 0,1,2,...,255 for alle filer, uanset hvilke hyppigheder bytes i filen har. Dvs. at man får faktisk et kodetræ, men ikke et kodetræ som er optimalt (for andre hyppigheder end 0,1,2,...,255). Derfor “virker” Encode og Decode, men man bruger ikke optimale koder, og derved bliver filer, som godt kan komprimeres væsentligt med Huffman (dvs. med optimale kodetræer), f.eks. txt-filer, ikke komprimeret godt, eller bliver ligefrem længere.
8	Man laver træer ved at have en Node klasse, som har to børn, der er *Elements* (som så indeholder en Node som data). Dvs. at man blander Elements og Nodes, hvilket er både ulogisk og kompliceret. Den rette løsning er (som beskrevet i opgaveteksten), at de to børn af en Node selv er Nodes, og at data i et Element er roden i træet (en Node). [En endnu klarere struktur vil man få ved at indføre et Tree objekt, som egentlig kun indeholder en Node, der er træets rod, og så lade data i Element være et Tree (dette er klarere, fordi et træ er en *samling* Nodes, og derfor ikke bør repræsenteres ved en enkelt Node).]