ADK - Teoriuppgifter labb 3 2009-03-23

Name: _____

1. **Sätt dig in i hur det givna programmet fungerar. Svara speciellt på följande frågor: Vad används datastrukturen used till i programmet? Varför används just breddenförstsökning och inte till exempel djupetförstsökning? När lösningen hittats, hur håller programmet reda på vilka ord som ingår i ordkedjan i lösningen?**

   Datastrukturen used är en lista med redan använda ord och används för att inte hitta samma ord om och om. Breddenförstsökning används för att hitta den kortaste stigen. WordRec håller koll på från vilket ord vi fått nuvarande ord med hjälp av en father->son-struktur.

2. **Både ordlistan och datastrukturen used representeras med klassen Vector i Java och sökning görs med metoden contains. Hur fungerar contains? Vad är tidskomplexiteten? I vilka lägen används datastrukturerna i programmet? Hur borde dessa två datastrukturer representeras så att sökningen går så snabbt som möjligt?**

   Contains returnerar strängen om den finns, och null om den inte finns. Tidskomplexiteten är O(antalet element) i vektorn, eftersom den i värsta måste gå igenom alla element tills den hittar rätt. Datastrukturerna används i LongestChain i metoderna MakeSons, CheckAllStartWords och BreadthFirst. De två datastrukturerna kan lämpligen representeras med med en hashad datastruktur eftersom insättning och contains går på O(1), alternativt Tries eftersom vi på så sätt har liknande ord väldigt nära varandra i en trädstruktur.

3. **I programmet lagras varje ord som en String. Hur många Stringobjekt skapas i ett anrop av MakeSons? Att det är så många beror på att Stringobjekt inte kan modifieras. Hur borde ord representeras i programmet för att inga nya ordobjekt ska behöva skapas under breddenförstsökningen?**

   MakeSons skapar en ny sträng vid varje anrop. Ord borde representeras som arrayer av chars alternativt CharBuffers. På så sätt kan man byta ut en char utan att behöva skapa en ny sträng.

4. **Det givna programmet gör en breddenförstsökning från varje ord i ordlistan och letar efter den längsta kedjan. Visa att det räcker med en enda breddenförstsökning för att lösa problemet.**

   Det räcker med en enda BFS eftersom MakeSons skapar alla ord som skiljer på en bokstav från ordet vi har. Dessa nya ord läggs in i en kö om de inte finns i used, och sedan utförs MakeSons på varje ord i kön.