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 databas som används till att hålla reda på vilka ord man har tittat på. Breddenförstsökning används för att finna vägen för att finna ett ord eller hur man kommer till det ordet. Använder sig av en kö där man lägger in alla ord som ingår i ordkedjan.

1. **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? KLAR!**

Metoden *contains* går igenom hela listan och tittar om ordet finns där. Tidskomplexiteten för att titta igenom hela ordlistan är i värstafall O(n) där *n* är antalet ord i ordlistan. *Contains* & *used* används i metoden MakeSons där *contains* används för att titta hur många ord det är som inte skiljer sig från slutordet och där *used* används för att markera det ordet man kommer byta från. *Used* används även i BFS för att märka att startordet är använt.

Använder en HashTabell istället då tidskomplexiteten average blir O(1) istället för O(n).

1. **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? KLAR!**

Det skapas 29\*4 Stringobjekt i MakeSons. Använder en char array istället för att kunna byta ut en bokstav i taget istället för ett helt ord i taget.

1. **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. KLAR!**Det går att göra genom att man istället tittar från slutvärdet och kör tills man inte kan något mer och tar det tidigare värdet och skriver ut det.