För att bygga grafen och söka i den har vi använt oss av ett python bibliotek. Detta paketet heter Networkx[1]. Denna används vanligtvis för att strukturera och organisera komplexa nätverk. Den har klasser och metoder för att bygga olika typer av grafer, och sökning.

Programmet börjar med att ladda in datafilen och kör en enkel formatering för att ta bort “\n” från slutet på varje ord. Därefter körs en funktion för att bygga grafen. Grafen är en Digraph från Network biblioteket. Grafen byggs med loopar som jämför ett ord med ett annat. Först kontrolleras så orden inte är samma, därefter kontrolleras bokstäverna. De fyra sista bokstäverna kontrolleras mot ett annat ord. Bokstäverna som matchas läggs till i en lista. Sedan om alla dessa bokstäver finns i första ordet, isåfall kommer en enkelriktad båge ritas mellan dessa orden.

När grafen är ritad laddas testfilen in, itererar över varje rad och delar upp varje rad i ett start och slutord (Startnod och slutnod).

Därefter körs en Bredden-först sökning mellan start och slutnod. Algoritmen för sökningen är en funktion som finns i biblioteket Networkx.

Komplexitet
Grafens utrymmeskomplexitet är O(V+E) där V motsvarar antalet noder och E motsvarar antalet bågar.

Tidskomplexiteten för att bygg grafen är $O(N^2)$. Detta eftersom varje del av inputen (varje ord) itereres (N), samt för varje del av den iterationen, itereras varje del av inputen igen (N). Alltså $N * N = N^2$.

Funktionen check_letters() kan ignoreras eftersom längden på varje ord är så liten, och skillnaden mellan storleken på indatan och längden på varje ord blir obetydlig när N (indatan) växer mot oändligheten.

Referenser
[1]"NetworkX — NetworkX", Networkx.github.io, 2020. [Online]. Available: https://networkx.github.io/. [Accessed: 17- Jan- 2020].

Anton Norberg
Jesper Svefelt