Datastrukturer och Algoritmer

Programmeringsuppgift 2 - Ordkedjor

Linus Forsberg
Olle Olsson
Jimmy Åkesson

Källor

Lösningen utgår från boken och även bokens hemsidas exempel på bredden-först-sökning (BFS algoritmen). Vi tyckte det var smidigast och passande att använda oss av kurslitteraturen för denna uppgift och eftersom exemplen var tydliga och väldokumenterade gav det oss tillräckligt med kunskap för att kunna lösa uppgiften.

Tidskomplexitet

Skapa en riktad graf: O(1) tid, Lägga till ord i graf: O(N) tid. Att lägga till element i Linked List tar O(1) tid och att lägga till element i HashMap tar O(1) tid. Vi är inte intresserade av konstanter i något fall vilket resulterar i en tidskomplexitet på O(N).

Den mest dominerande uppgiften är att iterera genom alla ord och sedan hitta kopplingar till dessa då vi behöver använda oss av två for-loopar för att kunna jämföra varje ord med resterande ord i listan (se metoden nedan). I den inre for-loopen finns även två if-satser som kontrollerar om orden är samma samt om orden har någon koppling baserat på kraven. Eftersom det är två for-loopar (vilka vi är intresserade utav) och två if-satser (som vi inte är intresserade utav) kan vi förhålla oss till for-looparna vilket ger oss en tidskomplexitet på $O(N^2)$.

Se kodsnutt nedan från BFS- klassen:

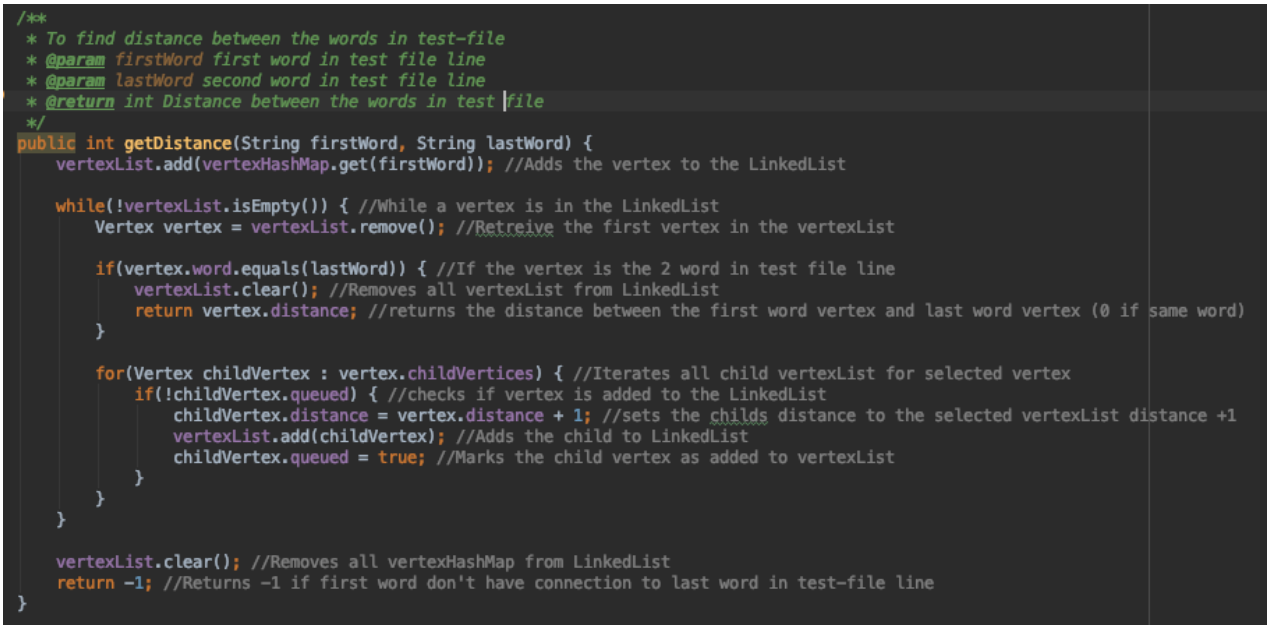
```java
/**
 * Adds a child node to a node in HashMap if words are connected
 * @param words ArrayList of all words from inputfile
 */
public void connectVertices(ArrayList<String> words) {

    for(int i = 0; i < words.size(); i++) { // Iterate all words in input file (word-1)
        for(int j = 0; j < words.size(); j++) { // Iterate all words in input file (word-2)

            String firstWord = words.get(i);
            String secondWord = words.get(j);

            if(i != j) { //Checks if word-1 is not the same as word-2
                if(isConnected(firstWord,secondWord)) { //Checks if word-1 and word-2 is connected
                    vertexHashMap.get(firstWord).childVertices.add(vertexHashMap.get(secondWord)); //Adds word-2 as childNode to word-1
                }
            }
        }
    }
}
```

Tidskomplexiteten för att göra bredden först- sökningen tar i värsta fall $O(N^2)$. Metoden getDistance i BFS- klassen har en while-loop som itererar genom kön så länge där finns noder att läsas genom vilket i värsta fall tar O(V) eftersom vi inte är intresserade av if-satsen. Däremot finns där en for-loop i denna while-loop som itererar genom alla barn-noder för vald nod, vilket ger oss O(E) eftersom detta utförs för varje nod (V). Detta kan i bästa fall vara O(1) och värsta fall $O(V^2)$. Se kodsnutt i nedan från BFS-klassen:



```java
/**
 * To find distance between the words in test-file
 * @param firstWord first word in test file line
 * @param lastWord second word in test file line
 * @return int Distance between the words in test file
 */
public int getDistance(String firstWord, String lastWord) {
    vertexList.add(vertexHashMap.get(firstWord)); //Adds the vertex to the LinkedList

    while(!vertexList.isEmpty()) { //While a vertex is in the LinkedList
        Vertex vertex = vertexList.remove(); //Retreive the first vertex in the vertexList

        if(vertex.word.equals(lastWord)) { //If the vertex is the 2 word in test file line
            vertexList.clear(); //Removes all vertexList from LinkedList
            return vertex.distance; //returns the distance between the first word vertex and last word vertex (0 if same word)
        }

        for(Vertex childVertex : vertex.childVertices) { //Iterates all child vertexList for selected vertex
            if(!childVertex.queued) { //checks if vertex is added to the LinkedList
                childVertex.distance = vertex.distance + 1; //sets the childs distance to the selected vertexList distance +1
                vertexList.add(childVertex); //Adds the child to LinkedList
                childVertex.queued = true; //Marks the child vertex as added to vertexList
            }
        }
    }

    vertexList.clear(); //Removes all vertexHashMap from LinkedList
    return -1; //Returns -1 if first word don't have connection to last word in test-file line
}
```

Klassen BFS lagrar alla objekt av klassen Vertex i en HashMap. Detta innebär att komplexiteten för minnesutrymmet är O(V). Varje Vertex sparar sedan alla barnnoder i en hash map, vilket ger en komplexitet för minnesutrymmet på O(E).

Det totala minnesutrymmet för minneshanteringen blir (O(V+E)). V står för antalet ord (vertices eller noder) och E står för antalet bågar (edges) mellan orden.