Algorithms – Programming Assignment 6

WordNet

B06602035 李晴妍

E-Mail : b06602035@ntu.edu.tw

Phone : 0987883537

System information

OPERATING SYSTEM: Windows

COMPILER: IntelliJ IDEA

TEXT EDITOR / IDE: IntelliJ IDEA

Describe concisely the data structure(s) you used to store the information in synsets.txt. Why did you make this choice?

DATA TYPE:

HashMap<Integer, String> synSets
HashMap<String, Bag<Integer>> synMap

我用了兩個 HashMap 來存 synsets.txt 裡的資料。

第一個 *HashMap<Integer, String> synSets* 放的是每個 id 所對應的 synset 內容，在這次作業中我沒有處理 gloss，直接捨棄。

第二個 *HashMap<String, Bag<Integer>> synMap* 是以 string nouns 當 key，去記錄每個 nouns 有哪些 id 有，這些 id 包成一個 Bag 當作 value。

會用兩個 HashMap 來記錄是因為在之後計算 sca 和 distance 時都是以 nouns 作單位去考慮，而不是 synset。所以多一個 HashMap synMap 會比較方便處理接下來的問題。

```java
private void readSyn(String synsets) {
    if (synsets == null) throw new IllegalArgumentException();

    In in = new In(synsets);
    while (in.hasNextLine()) {
        size++;
        String line = in.readLine();
        String[] splits = line.split(",");
        int id = Integer.parseInt(splits[0]);
        synSets.put(id, splits[1]);
        String[] nouns = splits[1].split(" ");
        for (String n : nouns) {
            if (synMap.get(n) != null) {
                Bag<Integer> bag = synMap.get(n);
                bag.add(id);
            }
            else {
                Bag<Integer> bag = new Bag<Integer>();
                bag.add(id);
                synMap.put(n, bag);
            }
        }
    }
```

Describe concisely the data structure(s) you used to store the information in hypernyms.txt. Why did you make this choice?

DATA TYPE:

Digraph wordnet;

我用了 Digraph 來存 hypernyms.txt 裡的資料。

因為 hypernyms.txt 都是記錄 vertice 與其他 vertices 間 directed edges 的關係。所以直接將 id 以 v 和 w 的方式保存下來就可以了。

```java
private void readHyp(String hypernyms) {
    if (hypernyms == null) throw new IllegalArgumentException();

    In in = new In(hypernyms);
    while (in.hasNextLine()) {
        String line = in.readLine();
        String[] splits = line.split(",");
        int v = Integer.parseInt(splits[0]);
        for (int i = 1; i < splits.length; i++) {
            int w = Integer.parseInt(splits[i]);
            wordnet.addEdge(v, w);
        }
    }
}
```

Describe concisely your algorithm to compute the shortest common ancestor in ShortestCommonAncestor. For each method, give the order of growth of the best- and worst-case running times. Express your answers as functions of the number of vertices V and the number of edges E in the digraph. (Do not use other parameters.) Use Big Theta notation to simplify your answers.

If you use hashing, assume the uniform hashing assumption so that put() and get() take constant time per operation.

Be careful! If you use a BreadthFirstDirectedPaths object, don't forget to count the time needed to initialize the marked[], edgeTo[], and distTo[] arrays.

DESCRIPTION:

我是用 BFS 去確認 digraph 找 the shortest common ancestor，用 BreadthFirstDirectedPaths 實作。假如說今天有兩個 id v 和 w。

BreadthFirstDirectedPaths vPath = new BreadthFirstDirectedPaths(digraph, v);
BreadthFirstDirectedPaths wPath = new BreadthFirstDirectedPaths(digraph, w);

這樣可以找到從 v/w 到 diagraph 裡面每一個 vertex 中的 shortest paths (Θ(*V* + *E*) in the worst case)。接著透過迴圈去確認 v 與 w 之間的 shortest paths (Θ(V*2(*V* + *E)*) in the worst case)，就可以確認是否有一個 ancestor 存在在 v 與 w 之間。

```
// loop through each vertex and find the minimal ancestor length one
for (int vertex = 0; vertex < digraph.V(); vertex++) {
    if (vPath.hasPathTo(vertex) && vPath.distTo(vertex) < distance && wPath
            .hasPathTo(vertex)
            && wPath.distTo(vertex) < distance) {
        int sum = vPath.distTo(vertex) + wPath.distTo(vertex);
        if (distance > sum) {
            distance = sum;
            ancestor = vertex;
        }
    }
}
```

ORDER OF GROWTH OF RUNNING TIME: Θ(V*(V+E)^3)

Describe any serious problems you encountered.

* 計算 distance 有點複雜，因為 nouns 會出現很多個，所以要去交叉比對不同的距離，找到最近的 distance，這部分的方法想了很久

List any other comments here. Feel free to provide any feedback on how much you learned from doing the assignment, and whether you enjoyed doing it.

* 很感謝老師一學期的教導！老師上課講解的都很詳細，也會不斷詢問大家聽懂了沒有，如果有機會會再修老師的其他課程！