Project Of Web Data Model
LIANG fei

Language: python3

1.assignment (1)

(a).implementation analysis

The fist assignment is to Implement a streaming algorithm for xpath queries of the form

$$//e1/e2/\cdot\cdot\cdot/e$$

Here I choose the stack to store the index positions. And use the Knuth-Morris-Pratt algorithm for xpath.

The Knuth-Morris-Pratt algorithm searches for occurrences of a "word" W within a main "text string" S by employing the observation that when a mismatch occurs, the word itself embodies sufficient information to determine where the next match could begin, thus bypassing re-examination of previously matched characters.

(b).algorithm

1.Initialization: creat a “partial macth” table for the path. Represent path query as an array for each step. A preorder ID to identify node;
2.Maintain a stack S of index positions;
3SstartDocument: empty stack S=[-1], id =-1;
4.StartElement:

id=id+1
if :Path[stack[-1]] and element match, push stack[-1]+1 to stack;
else :
if:Path[table[stack[-1]] and element match, push table[stack[-1]]+1 to stack;
else:
If: path[0] and element match, push 0 to stack;
else : Failure, push -1 to stack ;

5.endElement: Pop old i from S.
6.If stack[-1]=len(path), we found a match.print the id of the node

(c).Optimization

I don’t load the file, just open the file and read in line.

I just use a stack.

Define some functions to optimize code, like satisfy_or_not.

(d).Experimental Evaluation

I created 4 files which have different size. (I use the function getsize(flie) in pyhton,filesize = getsize()/float(1024)).

For the execution time, I use time.time() to compute.

Figure 1 is a plot has the size of the document on the x-axis and the execution time on the y-axis. We can see that document size and runtime are linear

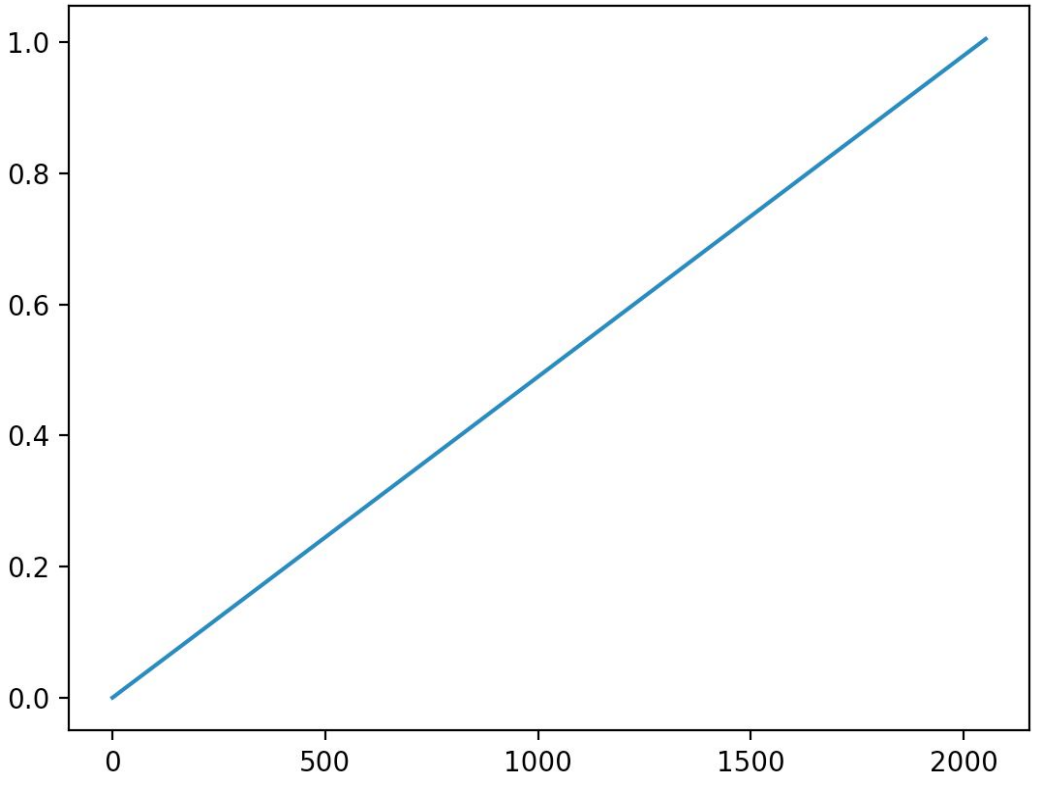

Figure 1

Figure 2 is a plot has the size of the query on the x-axis and the execution time on the y-axis.

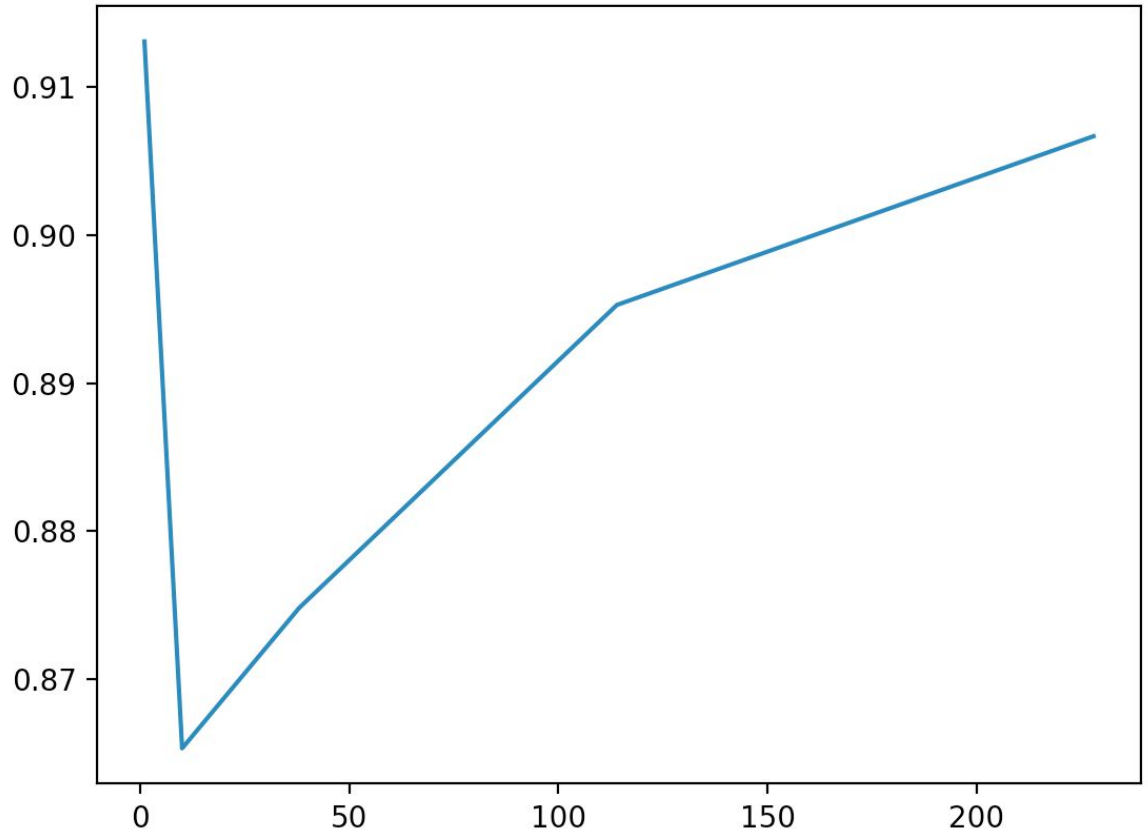

Figure 2

2.assignment(2)

(a).implementation analysis

This assignment is to Implement a streaming algorithm for xpath queries of the form

$$//p1//p2// \cdots //pn,$$

where each pi is an path of the form: and eij are element names.
//p1//p2// · · · //pn, ei1/ei2/ · · · /eim,

Firstly, I split the query to get the final state, epsilon state(state after"//"), transition rule(called state in code).
Final state: int
Epsilon state: list[int]

State: list[int]
For example:
If query is //a/b//a/b, we have five state 0,1,2,3,4, so final state is 4, epsilon state is [0,2] and transition is [a,b,a,b] which means for state 0, we can translate to 1 with “a” because state[0]=”a”.

Secondly, I use a list to store the dfa states.
dfa state:list[int]
Each time I update the dfa state with transition rule.

Thirdly, I use a stack to store the transitions between a new dfa state and previous dfa state.
transitions:list[list[int]]
For example:
If input is a,b,a,a,a … and query is //a/b//a/b
transition is [[1],[2],[1,3],[1,3],[1,3]]
dfa state is [0]->[0,1]->[0,2]->[0,2,1,3]->[0,2,1,3,1,3]->[0,2,1,3,1,3,1,3]
we know that transition is a stack, so the top of it is [1,3] which means last input a has translate 0 to 1 and 2 to 3.

Finally, I use a set to store the result.
result: set[int]

(b).algorithm

```
we have a stack[ ],a NFA table[ ],result set[ ]
initNfa()
initDfaState()
readStream(){
        StartElement:
                transition(element, index){
                        update lazy DFA state(element)
                        push transtions in stack
                        if (lazy DFA state have final state):
                                we find a match, add index in result set
```

```
                }

    EndElement:
            reverse_transition(){
            operation = stack.pop()
            update lazy DFA state(operation)
            }
    }
```

(c).Optimization

I only use one list to store all the information of transition between two state. The content of list is the transition condition. The index of the list identify the state.

(d).Experimental Evaluation

Figure 3 is a plot has the size of the document on the x-axis and the execution time on the y-axis. We can see that document size and runtime are linear

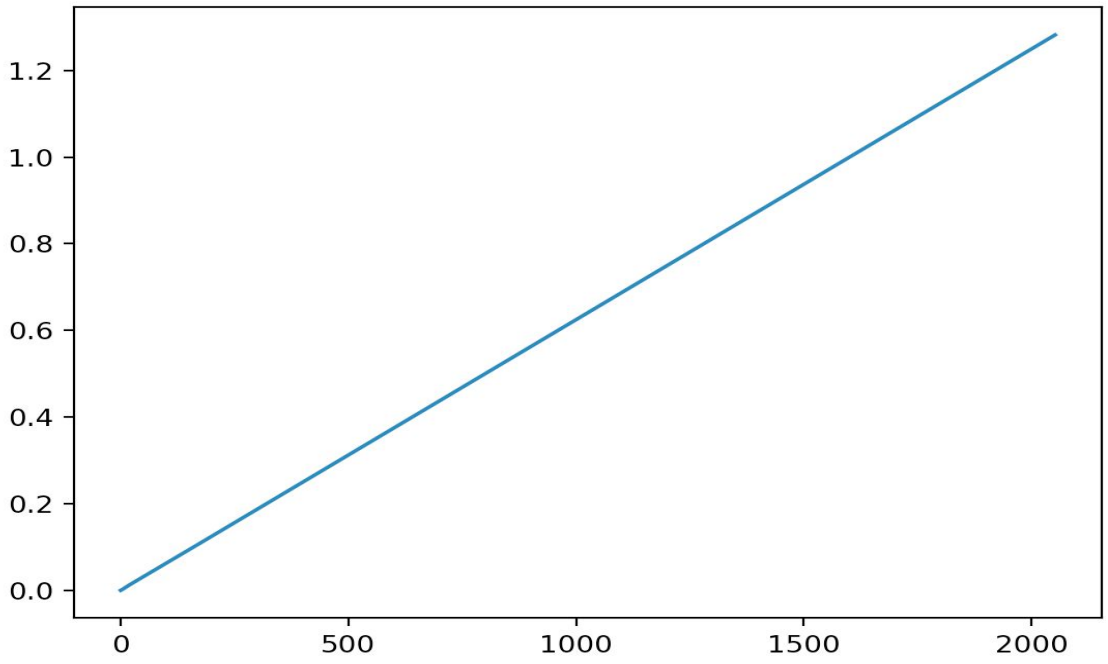

Figure 3

Figure 4 is a plot has the size of the query on the x-axis and the execution time on the y-axis. The execution time will increase if the size of the query increase

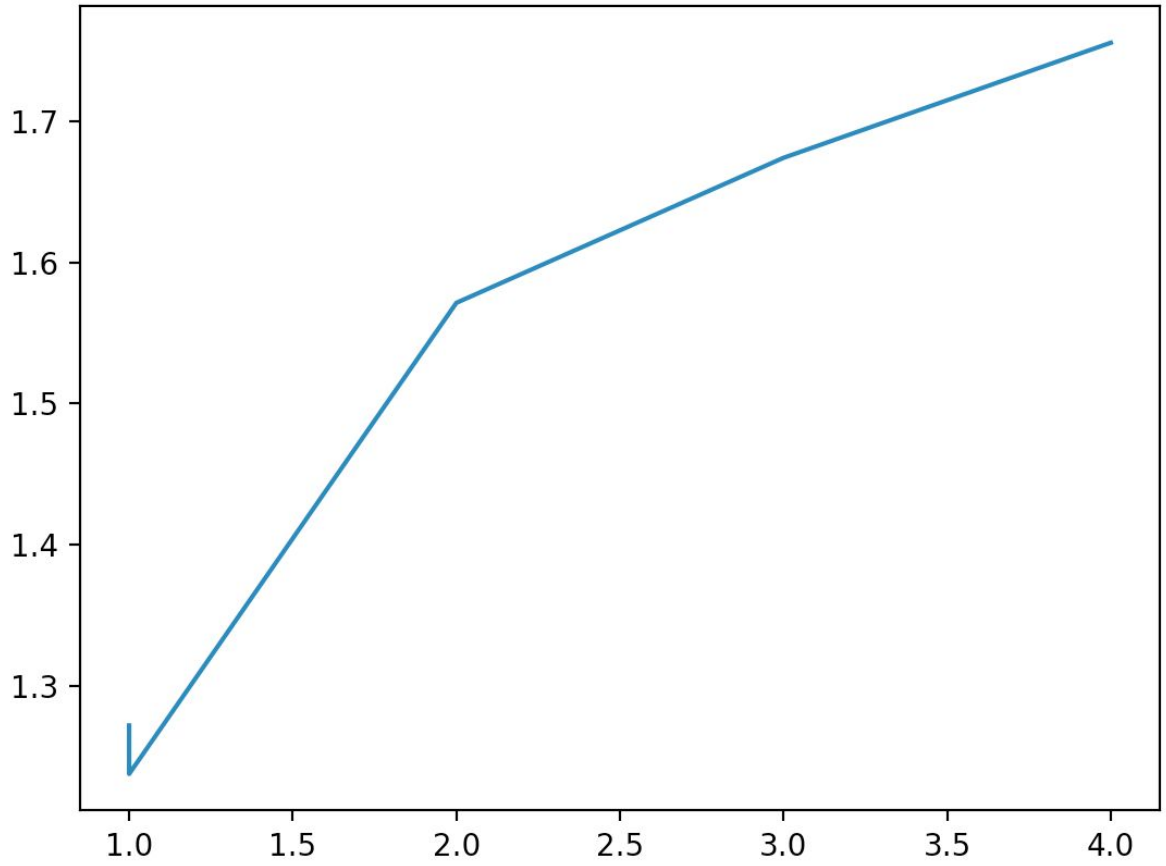

Figure 4

3.Conclusion

In this assignment, I used two different algorithms. The Knuth-Morris-Pratt algorithm and lazy DFA. I think their main idea is the same, to deal with the uncertain transfer. The assignment 1 can also be implemented in a simpler way.