

ASSIGNMENT 2

ALGORITHMS & COMPLEXITY (CIS 522-01)

Javier Arechalde

February 18, 2018

1. Time-series data mining

1.1 Problem description

In this problem we will have a sequence of events, and we want to find out if this sequence of events is a subsequence of other longer sequence, but the events don't necessarily need to be consecutive.

For example, we will have a sequence of events as follows:

buy Yahoo, buy eBay, buy Yahoo, buy Oracle

And another longer sequence, which may or may not contain the subsequence of events shown above.

buy Amazon, buy Yahoo, buy eBay, buy Yahoo, buy Yahoo, buy Oracle

The goal is to quickly detect if S' is a subsequence of S .

Then we will formulate this problem this way:

Given two sequence of events, S' of length m and S of length n each containing an event possibly more than once, we want to find in time $O(m + n)$ if S' is a subsequence of S .

1.2 Proposed solution

In our implementation, we will iterate over the two arrays, to find if S' is a subsequence of S .

We will start by taking the first element in S' , then we will start checking if we can find that element in S . If we find that element in S , we move to the next element in S' , and start searching for the second element in S , starting from the S position right after we found the first element of S' . If we can't find that element in S , that means we reached the final position in S without finding that element, we will return that S' is not a subset of S . If we find all the elements of S' in S , we will return that S' is a subset of S .

1.3 Pseudo code

Algorithm 1 Checking if S' subset of S

```
1: We initialize  $i_{pos} = j_{pos} = 0$ 
2: while We didnt reach the end of  $S$  or  $S'$  do
3:   Take  $S(i_{pos})$ 
4:   for  $j$  in range  $j_{pos} \rightarrow length(S)$  do
5:     if  $S'(i_{pos}) == S(j_{pos})$  then
6:       if  $i_{pos} == length(S')$  then
7:          $S'$  is a subsequence of  $S$ 
8:       end if
9:       if  $i_{pos} \neq length(S')$  then
10:         $i_{pos}++$ 
11:         $j_{pos} = j$ 
12:        Break the loop
13:      end if
14:    end if
15:    if  $S'(i_{pos}) \neq S(j_{pos})$  then
16:       $j++$ 
17:    end if
18:  end for
19: end while
```

1.4 Example

Here we should prove that our algorithm is correct too.

#These are the two sets that we want to compare

```
s1 = [ 'buy_Yahoo', 'buy_eBay', 'buy_Yahoo', 'buy_Oracle' ]
```

```
s2 = [ 'buy_Amazon', 'buy_Yahoo', 'buy_eBay', 'buy_Yahoo', 'buy_Yahoo', 'buy_Oracle' ]
```

```
def comp(s1, s2):
```

```
    #Getting the length of the two sets
```

```
    l1 = len(s1)
```

```
    l2 = len(s2)
```

```
    #We initialize the operators
```

```
    ipos = 0
```

```
    jpos = 0
```

```
    while (jpos < l2 - 1):
```

```
        el1 = s1[ipos]
```

```
        for j in range(jpos, l2):
```

```

el2 = s2[j]

#If the element is in that position of the list...
if el1==el2:
    if (ipos==l1-1):
        ipos = ipos+1
        print("S1 is a subset of S2")
        return
    elif (ipos!=l1):
        ipos = ipos+1
        jpos = j
        break #We exit the for loop

#If we cant find the element in that position of the list
#We move to the next element
elif (el1!=el2):
    jpos = j
    continue

#If the loop above didn't return that S1 is a subset of S2 we print the opposit
print('S1 is not a subset of S2')
return

comp(s1 , s2)

```

1.5 Time complexity

In worst case scenario, the last element of S' will be in the last position of S , therefore, we would have iterated over both lists to check if S' is a subset of S . As the length of S' is m and the length of S is n , the time complexity of our implementation will be $O(m + n)$.

2. Competition scheduling

1.1 Problem description

1.2 Proposed solution

1.3 Pseudocode

1.4 Example (Implementation)

Here we should prove that our algorithm is correct too.

1.5 Time complexity