VE484 MP1 REPORT

Wei Yizhou
5133709269

Introduction

In this machine problem, I realized two algorithms to get the frequent subsequence; GSP algorithm and Spade algorithm. After that, I used several different test cases to measure the performance.

Assumption

While reading the pseudo code and doing the homework, I feel that Spade algorithm should be more efficient that GSP algorithm because it does not need to visit the whole database whenever we calculate the support.

Testing

For testing procedure, I take part of the given Database as my input. Because the original Databases are too complex.
First I cut all the sequences into sequences with length 29.
Then I change the number of the input sequence(N) and the minimum support to compare the performance. The result is shown below.

Sequence length = 29

N	**Minsup**	**Time (s)**	
		GSP	Spade
10	5	58.2864	1.1524
	6	8.7709	0.5372
	10	0.0026	0.011
50	25	11.721	1.6481
	50	0.0003	0.0060
100	50	21.942	4.1374
	100	0.0003	0.0072
200	100	23.544	7.9154
	200	0.0003	0.0081
1000	999	0.0010	0.0262
	1000	0.0007	0.0166

Then I change the length of the sequences to 59 and do the same procedure.
Sequence length = 59

N	Minsup	Time (s)	
		GSP	Spade
50	49	2.4764	2.4089
	50	0.1223	0.4470
100	99	0.4578	1.8464
	100	0.0378	0.4232
200	199	0.2669	2.5102
	200	0.0325	0.5988
1000	999	0.1494	2.9854
	1000	0.0715	1.6845

- **Result**

From the two tables, we can find that the time used by the Spade algorithm and GSP alters with different minimum support. After further studied the algorithm, I find that Spade algorithm is less efficient than GSP algorithm when the minsup is close to the total sequence number. Next I will analyze the time complexity for both algorithms.

GSP:

For GSP algorithm, we first need to compute the support for all nodes in the current level; then we need to extend the current level.

For support computation, we need to do three iterations; iterate all the sequences in the database, iterate all the sequences in the current level and iterate every character in the sequence to make sure it is a subsequence. Thus the time complexity is about O(n'nl). n' is the number of the nodes in current level. n is the total number of the

sequences in DB, and l is the length of the sequence.

For level extension, we have three iterations in the current level. Thus time complexity is $O(n'^3)$

Thus the total time complexity for GSP is $O(n'^3)+O(nn'l)$

Spade:

For spade algorithm, we have two iterations for the current level and in each iteration we have to derive the support for the new sequence. When deriving support, we have to compare every existence of the last symbol to n existence array. Thus the complexity is O(nl).

Thus the total time complexity is $O(n'^2nl)$

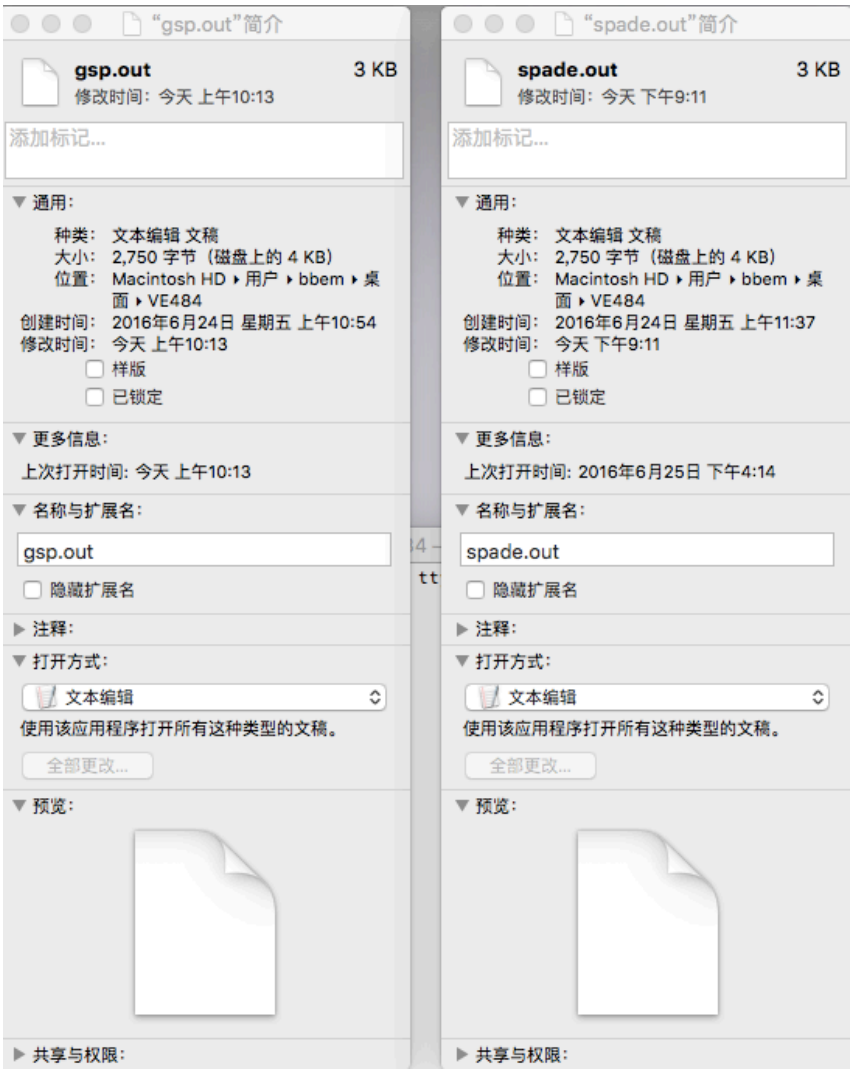
n': the number of the nodes in current level.

n: the total number of sequences in the database.

l: the length of the sequences in the database.

The analysis fits my testing result. The spade algorithm is less efficient than GSP when minimum support is close to the total number of the sequences because we are more likely to have fewer nodes in each level, causing nl > n'. However, when minimum support is rather smaller than total number of the sequences, we will have significant large number of nodes in each level causing n'>>nl thus making Spade become faster than GSP.

Appendix



The screenshot is the output file of two algorithms through the same input. The size of the file are the same which means the realization is correct.