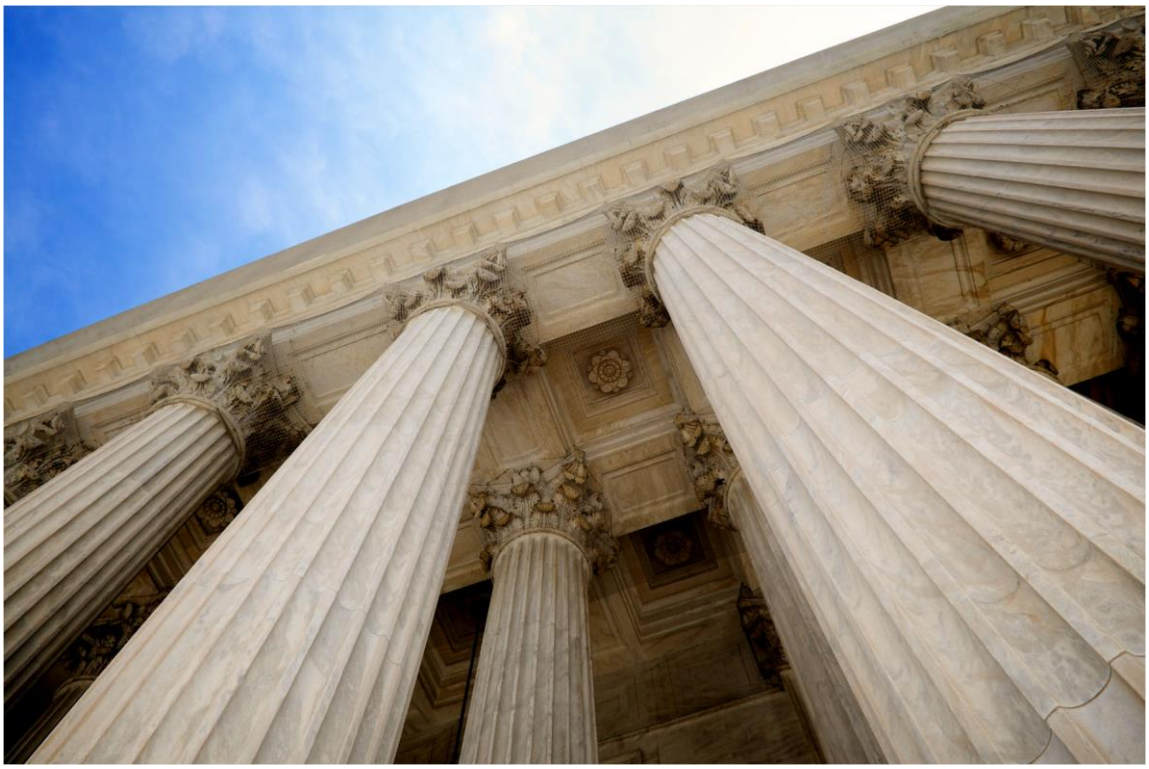


CSCI 4030U

PROJECT PART 1

Sheron Balasingam | Big Data Analytics | 02/23/2018

How To Run

make run	Compiles and runs the program (linux)
g++ -g -std=c++11 -Wall -Iinclude src/*.cpp src/algorithms/*.cpp -o main.exe	Compiles program (windows)
./main.exe	Runs program (windows)

A-Priori

PASS 1 (SRC/ALGORITHMS/A_PRIORI.CPP LINE 14)

Reads textfile line by line and stores count in a map<int, int> variable named itemCount. After the first pass, a vector<vector<int, int> variable called freqArr stores all the possible pairs made from the frequent items.

freqArr is set to have 3 columns to store the count in the first column and the pair in the other two columns. The number of rows is determined by the formula n!/((n-2)!*(2!)) which can be simplified to n*(n-1)*(n-2)/2!. For triples this would be n*(n-1)*(n-2)*(n-3)/3!. This formula is handled by the function getItemsetSize in src/util.cpp.

PASS 2 (SRC/ALGORITHMS/A_PRIORI.CPP LINE 21)

Goes through each line in the textfile and checks if the values of the second and third columns in each row of freqArr exist in that line. The count of the pair occurrences are stored in the first column. At the end of this pass, the contents of this array are copied to a variable of the same type freqPairs.

Next freqArr is resized to store triples instead of pairs and now will have 4 columns.

PASS 3 (SRC/ALGORITHMS/A_PRIORI.CPP LINE 28)

Does the same as pass 3 but checks a third value too. freqArr contents are copied to freqTrips. Storage in freqPairs and freqTrips are used for printing purposes.

OBSERVATIONS:

Support Threshold	# of Pairs	# of Triples	Time
100			29256.3
200	88	47	1166.13
300	43	21	154.196
400	27	11	36.2444
500	20	8	8.89526

750	10	7	1.83756
1000	9	5	0.576128
1500	8	1	0.576165
2000	6	1	0.58087
4000	2	0	0.447786
8000	0	0	0.424662
10000	0	0	0.421768

Park-Chen-Yu

PASS 1 (SRC/ALGORITHMS/PCY.CPP LINE 20)

The same code as A-Priori above for this Pass.

PASS 2 (SRC/ALGORITHMS/PCY.CPP LINE 23)

Gets the possible unique pairs the same as A-Priori using getItemsetSize to set the number of rows for freqArr, but in this case we do it line by line in the textfile. The count is added to a bucket array at the location using "(num1 XOR num2) mod sizeOfBucketArray". sizeOfBucketArray is represented by bucketNum in my code and it is pre-set.

The program will display the total counts for each bucket in the buckets array to allow the user to choose a support threshold for the buckets (time paused during this time).

A bit vector is created with the same storage number as the buckets array. It is initially set to 1 for all addresses in the vector. Then the program goes through the buckets array and compares to the threshold input by the user to set the bit to zero when the buckets number is less than threshold.

PASS 3 (SRC/ALGORITHMS/PCY.CPP LINE 64)

Now checks to see if the location the pairs hash to in the bit vector is a 1 and also checks if each item the pair is a frequent item. This item is added to hashPairs (of type vector<pair<int, int>>)

OBSERVATIONS:

Using bucket threshold of **357116** with **5** buckets:

Support Threshold	**# of Pairs**	**Time**
100	17486	7.25268
200	2430	2.34127

300	644	1.86283
400	259	1.66094
500	95	1.55875
750	29	1.49377
1000	6	1.45127
1500	6	1.45653
2000	6	1.45891
4000	1	1.43933
8000	0	1.39727
10000	0	1.411

Using bucket threshold of **89279** with **20** buckets:

Support Threshold	**# of Pairs**	**Time**
100	8833	3.36861
200	1255	1.8057
300	351	1.59283
400	144	1.51362
500	52	1.45644
750	19	1.42702
1000	6	1.39608
1500	6	1.3997
2000	6	1.39754
4000	3	1.38325
8000	1	1.37937
10000	0	1.36741

Using bucket threshold of **35711** with **50** buckets:

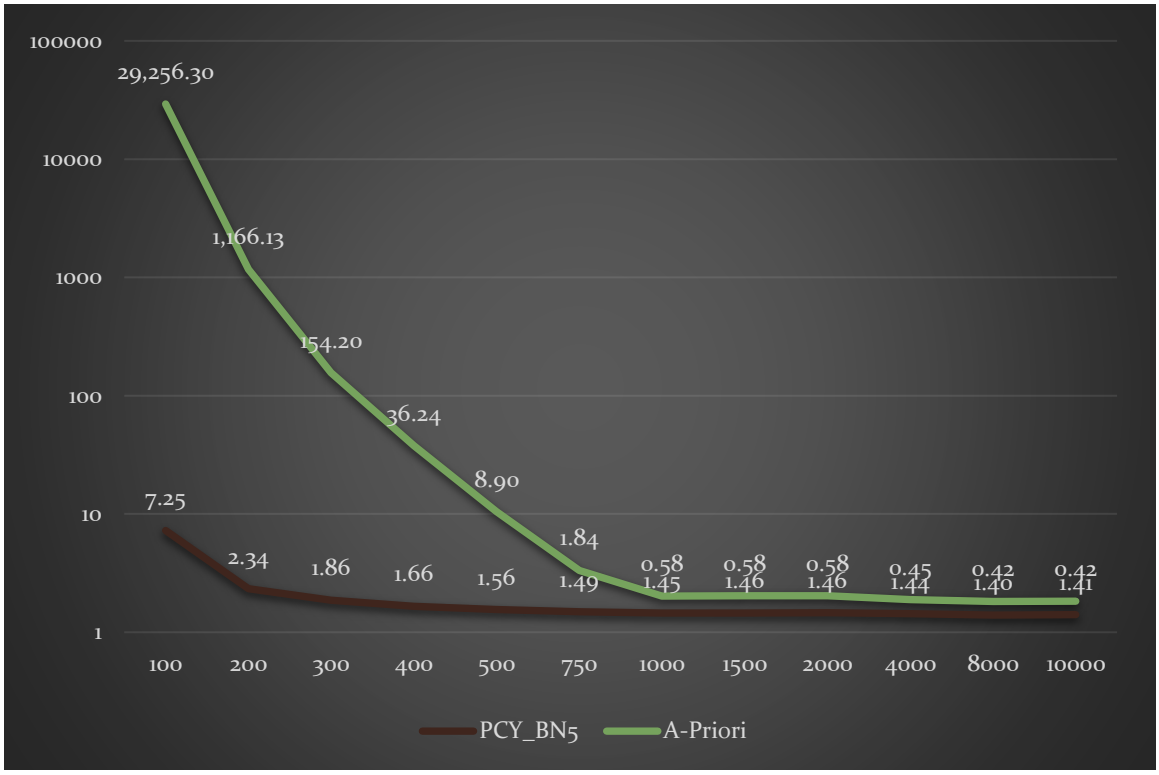
Support Threshold	**# of Pairs**	**Time**
100	10740	4.16315
200	1438	1.92436
300	404	1.62903
400	159	1.54022
500	66	1.48214
750	23	1.43081
1000	10	1.40493
1500	10	1.40379
2000	10	1.40932
4000	3	1.38823
8000	1	1.38321
10000	0	1.37119

Using bucket threshold of **17855** with **100** buckets:

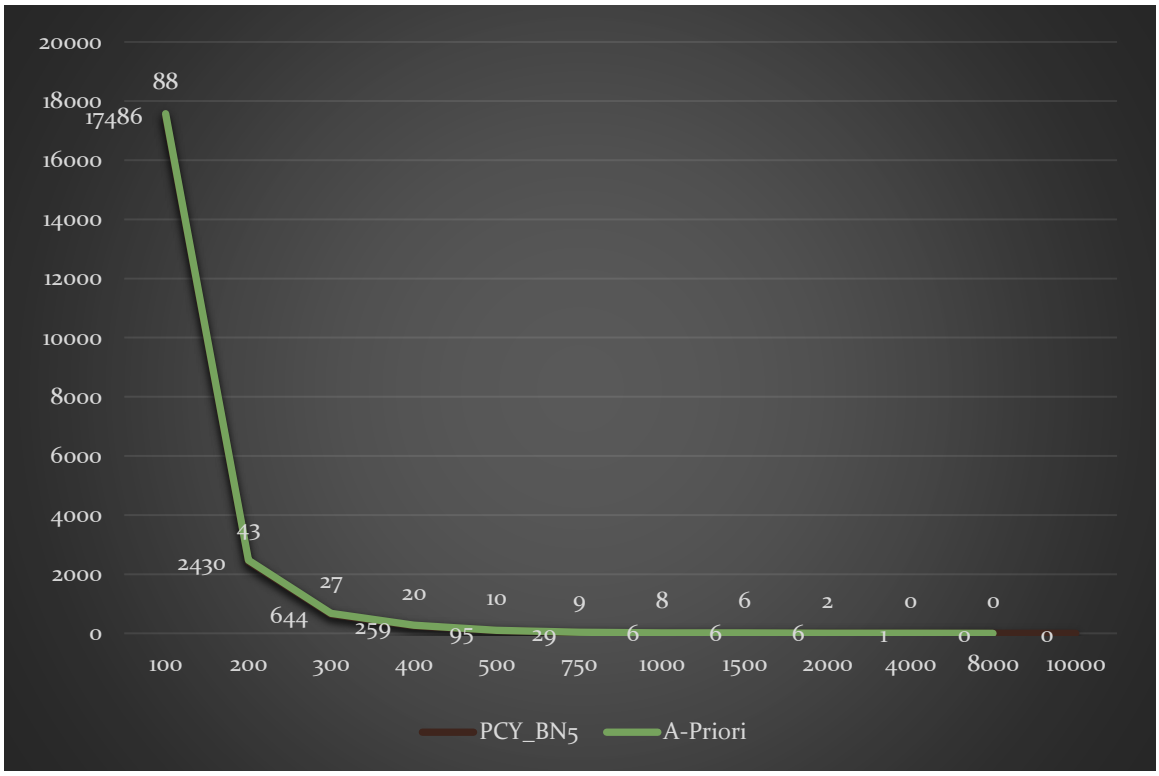
Support Threshold	# of Pairs	Time
10	228307	251.626
100	8524	3.11889
200	1152	1.74517
300	324	1.55944
400	131	1.48561
500	65	1.4483
750	29	1.41256
1000	10	1.39826
1500	10	1.39159
2000	10	1.39906
4000	3	1.7855
8000	1	1.3789
10000	0	1.7855

Comparing A-Priori and PCY

TIME VS SUPPORT THRESHOLD:

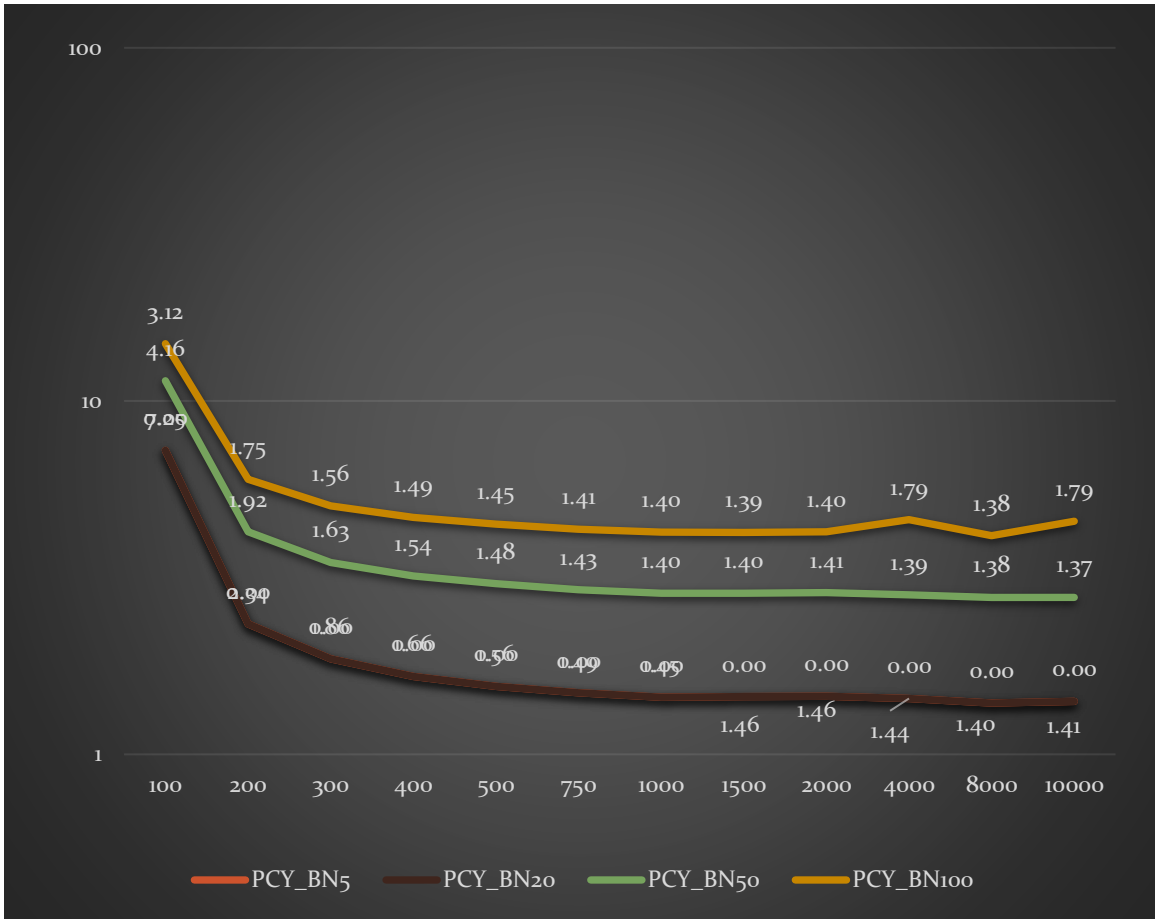


NUMBER OF CANDIDATE FREQUENT PAIRS VS SUPPORT THRESHOLD:



Comparing PCY Different Bucket Numbers (mod number)

TIME VS SUPPORT THRESHOLD:



NUMBER OF CANDIDATE FREQUENT PAIRS VS SUPPORT THRESHOLD: