Data Science

Programming Assignment 1

apriori Algorithm

Design Document

2015004011

major in Software

Dohyun Kim

1. Summary of my algorithm

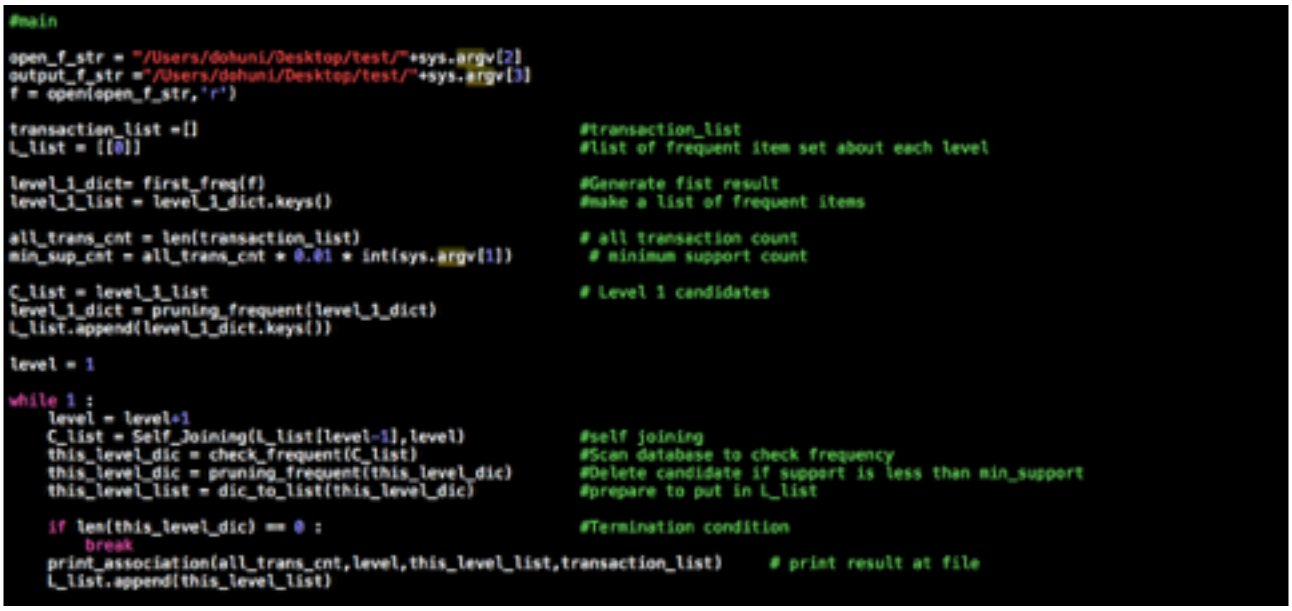
This assignment is to implement the ‘apriori algorithm’. This algorithm is a way to reduce test time by not creating its superset when there is any infrequent item set. I followed the pseudo-code of the lecture material.

I’ll briefly explain the sequence.

1. Read the input file and parse it to make a list for each transaction.
2. Find the frequency of each item to find the level 1 result.
3. Create the following candidates through Self_joining.
4. Scans the database for candidates and obtains frequency
5. Delete candidates with a frequency that is less than the minimum number of supports.
6. Output the result to a file according to a specific format.
7. Repeat steps 3 through 6 until the element is zero in the resulting list.

2. Detailed description of my codes

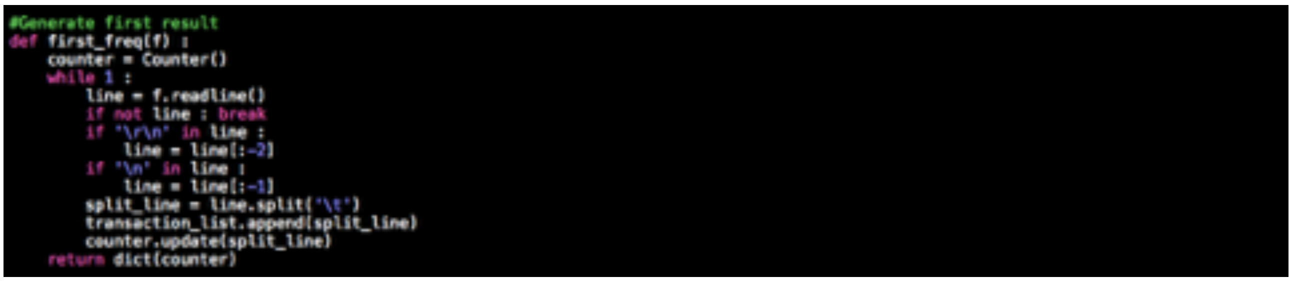
1.main



```
#main

open_f_str = "/Users/dohuni/Desktop/test/"+sys.argv[2]
output_f_str ="/Users/dohuni/Desktop/test/"+sys.argv[3]
f = open(open_f_str,'r')

transaction_list =[]                                        #transaction_list
L_list = [[0]]                                              #list of frequent item set about each level

level_1_dict= first_freq(f)                                 #Generate fist result
level_1_list = level_1_dict.keys()                          #make a list of frequent items

all_trans_cnt = len(transaction_list)                       # all transaction count
min_sup_cnt = all_trans_cnt * 0.01 * int(sys.argv[1])        # minimum support count

C_list = level_1_list                                       # Level 1 candidates
level_1_dict = pruning_frequent(level_1_dict)
L_list.append(level_1_dict.keys())

level = 1

while 1 :
    level = level+1
    C_list = Self_Joining(L_list[level-1],level)            #self joining
    this_level_dic = check_frequent(C_list)                 #Scan database to check frequency
    this_level_dic = pruning_frequent(this_level_dic)       #Delete candidate if support is less than min_support
    this_level_list = dic_to_list(this_level_dic)           #prepare to put in L_list

    if len(this_level_dic) == 0 :                           #Termination condition
        break
    print_association(all_trans_cnt,level,this_level_list,transaction_list)     # print result at file
    L_list.append(this_level_list)
```

first, It is used as an argument when executing, and it specifies the input file and the out file path. The main function follows a brief introduction to the algorithm.

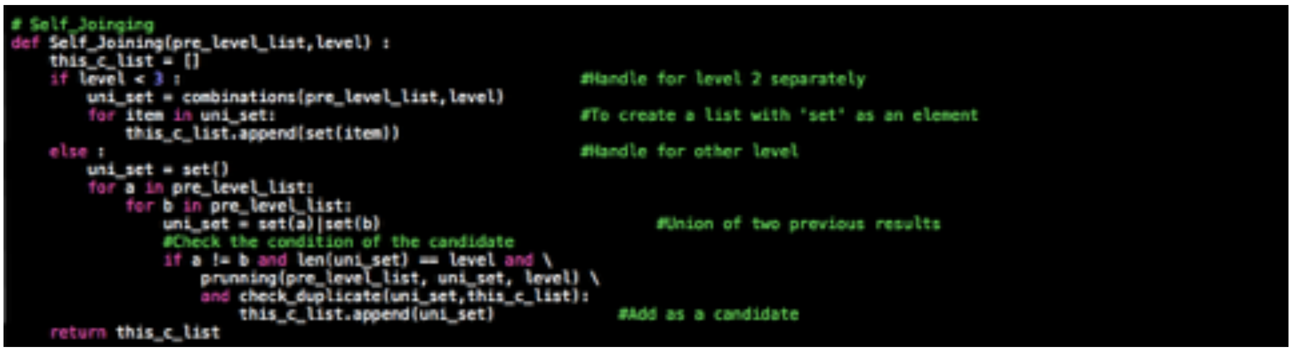
2.first_freq(f)



```
#Generate first result
def first_freq(f) :
    counter = Counter()
    while 1 :
        line = f.readline()
        if not line : break
        if "\r\n" in line :
            line = line[:-2]
        if "\n" in line :
            line = line[:-1]
        split_line = line.split('\t')
        transaction_list.append(split_line)
        counter.update(split_line)
    return dict(counter)
```

This function reads the file, makes lists of each transaction, and creates a one-element candidate by scanning.
Because there is a line break in the input file, remove '/ r / n' or '/ n' to put only item information into the list. On my computer it is '/ r / n' but on other computers it is sometimes recognized as '/ n', so I processed it separately.
Also, I used the Counter function. This function has the ability to automatically count the number of elements in the list. And like dictionary, it returns element type with key as key and value as value type.

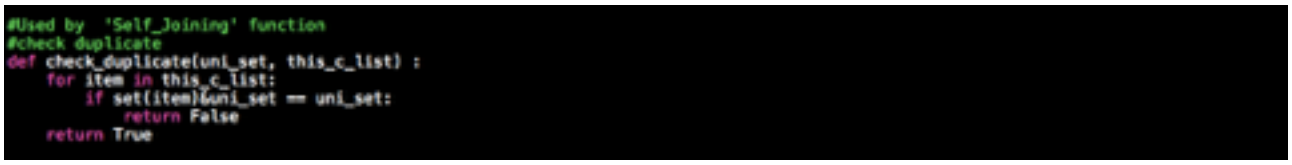
3. Self_Joining



```
# Self_Joinging
def Self_Joining(pre_level_list,level) :
    this_c_list = []
    if level < 3 :                                              #Handle for level 2 separately
        uni_set = combinations(pre_level_list,level)
        for item in uni_set:                                    #To create a list with 'set' as an element
            this_c_list.append(set(item))
    else :                                                      #Handle for other level
        uni_set = set()
        for a in pre_level_list:
            for b in pre_level_list:
                uni_set = set(a)|set(b)                                 #Union of two previous results
                #Check the condition of the candidate
                if a != b and len(uni_set) == level and \
                    prunning(pre_level_list, uni_set, level) \
                    and check_duplicate(uni_set,this_c_list):
                        this_c_list.append(uni_set)                 #Add as a candidate
    return this_c_list
```

The Self_joining function creates a candidate with the frequent elements of the previous level. The reason for dividing the level by the case of less than 3 and not by the case is the speed.Making a candidate with a combination is more efficient than examining multiple conditions.This is because in level 2, we make candidates with frequent items with one element. However, from the next level, too many candidates are created in combination, so it is efficient to union the previous level frequent item sets and pruning them.

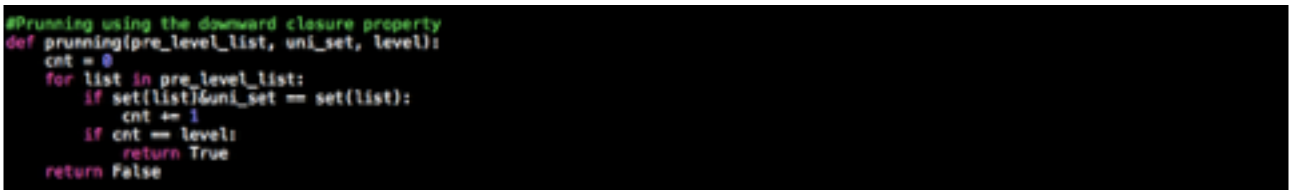
4. check_duplicate



```
#Used by  'Self_Joining' function
#check duplicate
def check_duplicate(uni_set, this_c_list) :
    for item in this_c_list:
        if set(item)&uni_set == uni_set:
            return False
    return True
```

This function is used in 'Self_joining'. When you create a candidate(uni_set), make sure the item is in the candidate list(this_c_list). Returns false if item is in candidate list.

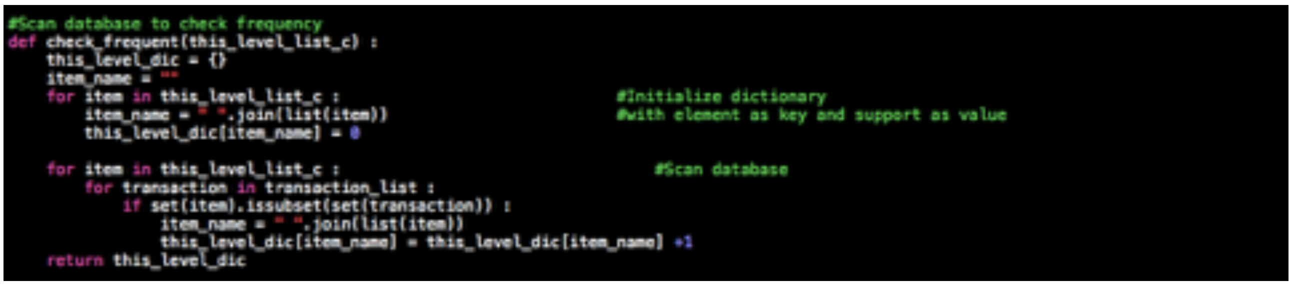
5.prunning



```
#Prunning using the downward closure property
def prunning(pre_level_list, uni_set, level):
    cnt = 0
    for list in pre_level_list:
        if set(list)&uni_set == set(list):
            cnt += 1
        if cnt == level:
            return True
    return False
```

This function deletes the candidate using the downward closure property. If the previous frequent item is intersected with the current candidate, then it is a subset of the current candidate.Returns true if the subset reaches the number of levels.

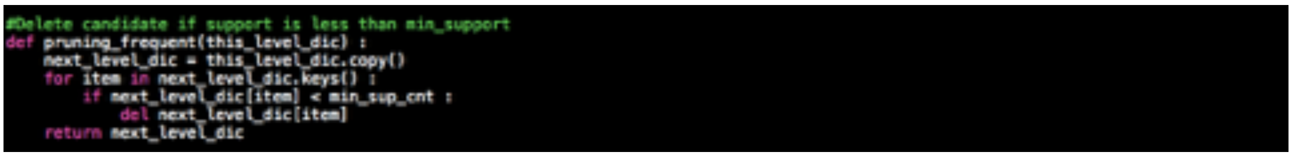
6.Check_frequent



```python
#Scan database to check frequency
def check_frequent(this_level_list_c) :
    this_level_dic = {}
    item_name = ""
    for item in this_level_list_c :                              #Initialize dictionary
        item_name = " ".join(list(item))                         #with element as key and support as value
        this_level_dic[item_name] = 0

    for item in this_level_list_c :                                 #Scan database
        for transaction in transaction_list :
            if set(item).issubset(set(transaction)) :
                item_name = " ".join(list(item))
                this_level_dic[item_name] = this_level_dic[item_name] +1
    return this_level_dic
```

This function scans the transactions and checks the frequency of each item. And this function returns a dictionary with item set as key and support as value.And because the dictionary key must be a string, use the join function. Separate each item with a space to make it a string.

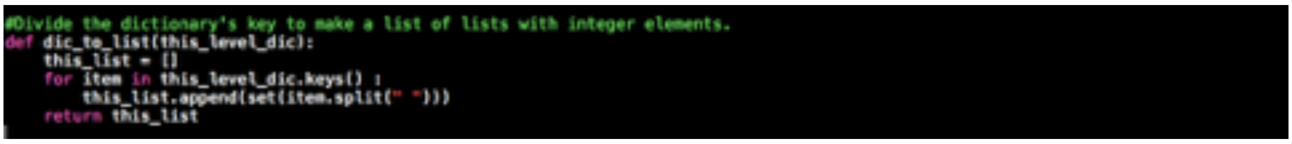
7. pruning_frequent



```python
#Delete candidate if support is less than min_support
def pruning_frequent(this_level_dic) :
    next_level_dic = this_level_dic.copy()
    for item in next_level_dic.keys() :
        if next_level_dic[item] < min_sup_cnt :
            del next_level_dic[item]
    return next_level_dic
```

This function delete candidate if support is less than minimum support count. Copy the element in the reference dictionary because the element should not be deleted during the loop.

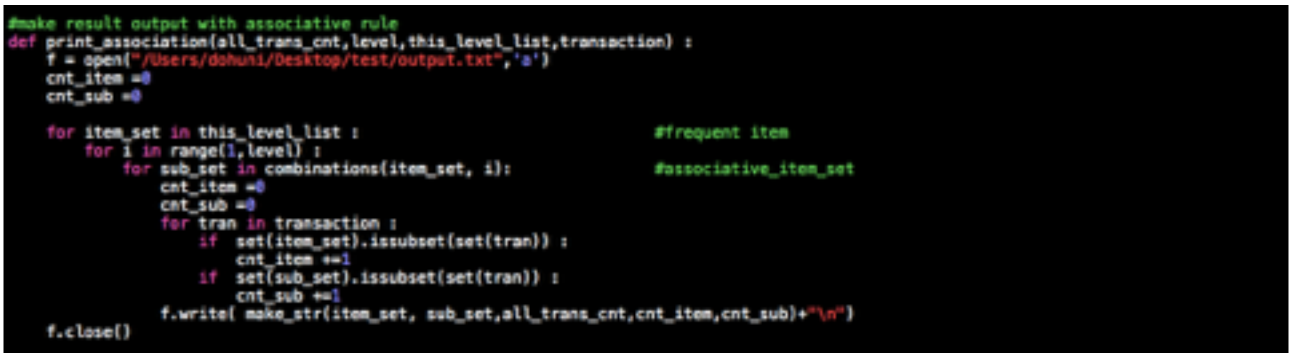
8.dic_to_list



```python
#Divide the dictionary's key to make a list of lists with integer elements.
def dic_to_list(this_level_dic):
    this_list = []
    for item in this_level_dic.keys() :
        this_list.append(set(item.split(" ")))
    return this_list
```

This function divide the dictionary’s key to make a list of list with integer elements. This is because when you create a candidate at the next level, I must have a list of item as integer type.

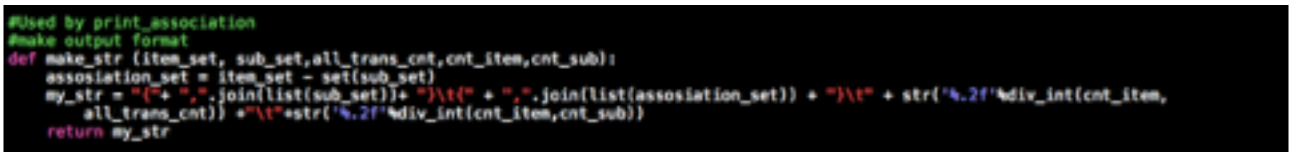
9.print_association



```
#make result output with associative rule
def print_association(all_trans_cnt,level,this_level_list,transaction) :
    f = open("/Users/dohuni/Desktop/test/output.txt",'a')
    cnt_item =0
    cnt_sub =0

    for item_set in this_level_list :                                  #frequent item
        for i in range(1,level) :
            for sub_set in combinations(item_set, i):                 #associative_item_set
                cnt_item =0
                cnt_sub =0
                for tran in transaction :
                    if  set(item_set).issubset(set(tran)) :
                        cnt_item +=1
                    if  set(sub_set).issubset(set(tran)) :
                        cnt_sub +=1
                f.write( make_str(item_set, sub_set,all_trans_cnt,cnt_item,cnt_sub)+"\n")
    f.close()
```

This function is a function for creating an output that applies Association rules.Create a subset of all levels to print support and confidence.

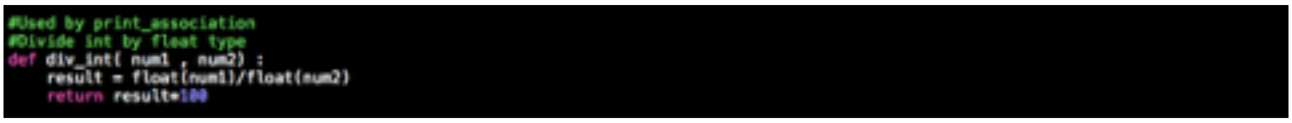
10. make_str



```
#Used by print_association
#make output format
def make_str (item_set, sub_set,all_trans_cnt,cnt_item,cnt_sub):
    assosiation_set = item_set - set(sub_set)
    my_str = "{"+ ",".join(list(sub_set))+ "}\t{" + ",".join(list(assosiation_set)) + "}\t" + str('%.2f'%div_int(cnt_item,
        all_trans_cnt)) +"\t"+str('%.2f'%div_int(cnt_item,cnt_sub))
    return my_str
```

It is a function that creates a string so that it can be output to a specific output format.

11. div_int



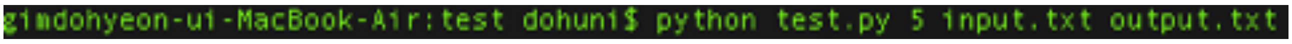
```
#Used by print_association
#Divide int by float type
def div_int( num1 , num2) :
    result = float(num1)/float(num2)
    return result*100
```

If the integers are divided, the result is an integer. So, it is a function that makes float type result. At the end, multiply by 100 for ease of use.

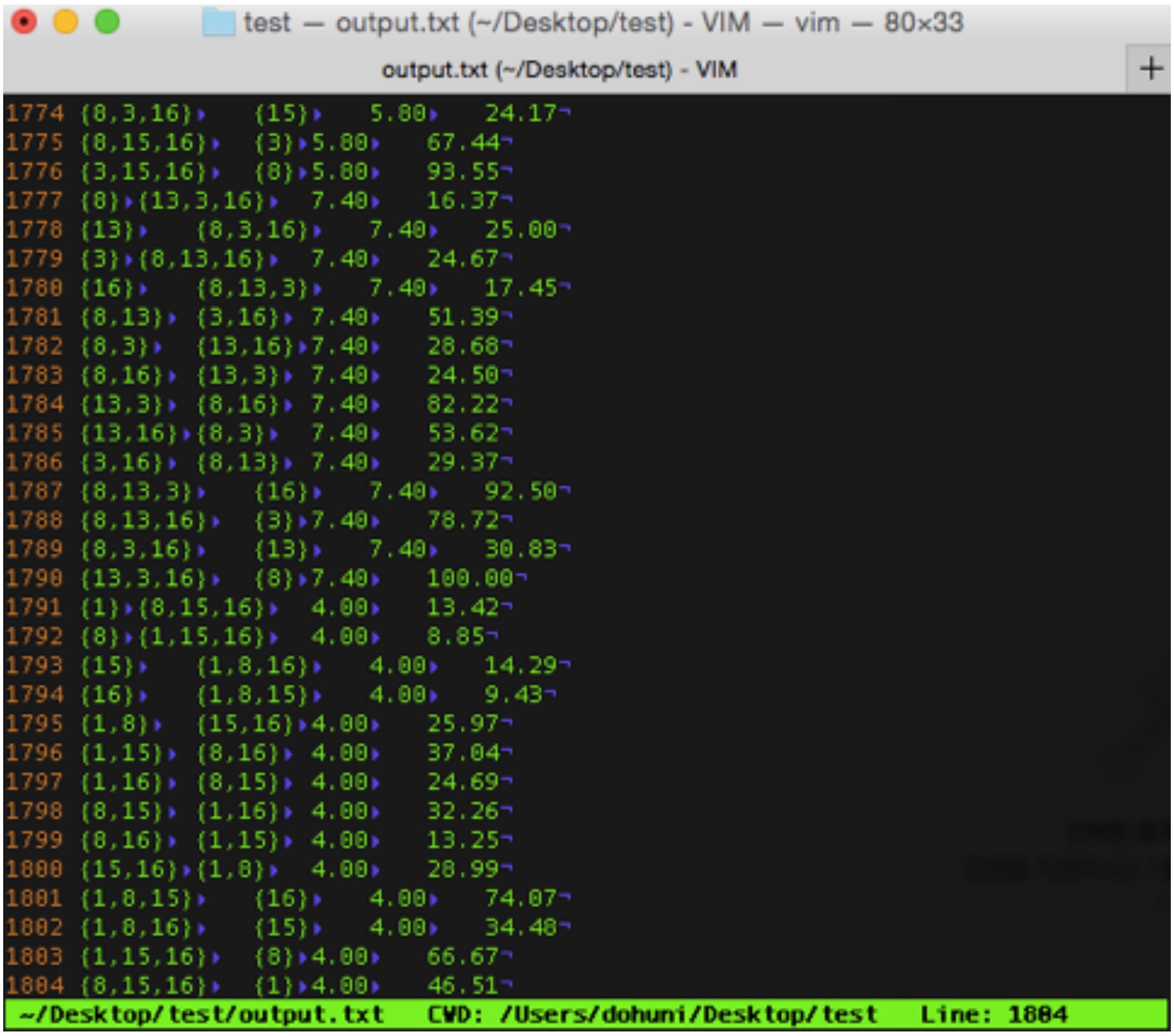
3. Instructions for compiling my source code

You need to make some changes to the source code before you run.If you scroll down, there is a comment marked “main". You need to modify the absolute addresses of "open_f_str" and "output_f_str" below it.The line number is 128,129.
My code was written in Python so when you run it you need to input "python apriori_2015004011.py 4 input.txt output.txt" in this way.
Enter python first, followed by the executable name, support, input file name, and output file name.

4. test result

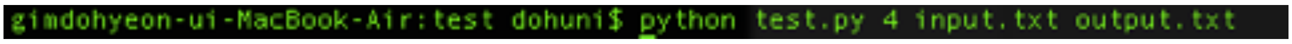


```
gimdohyeon-ui-MacBook-Air:test dohuni$ python test.py 5 input.txt output.txt
```

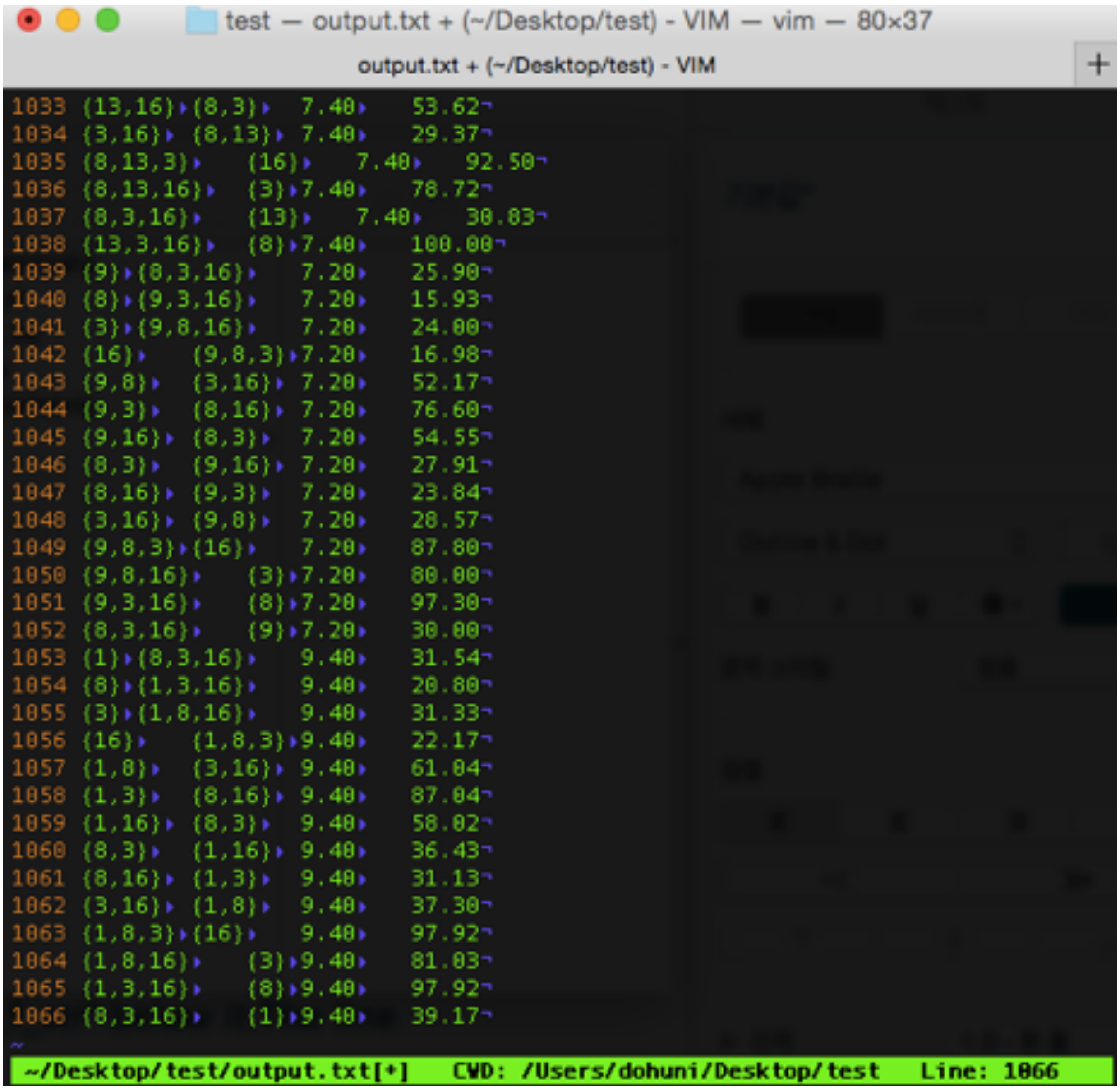


test — output.txt (~/Desktop/test) - VIM — vim — 80×33
output.txt (~/Desktop/test) - VIM

```
{8,3,16}▸    {15}▸    5.80▸    24.17¬
{8,15,16}▸   {3}▸5.80▸    67.44¬
{3,15,16}▸   {8}▸5.80▸    93.55¬
{8}▸{13,3,16}▸   7.40▸    16.37¬
{13}▸    {8,3,16}▸    7.40▸    25.00¬
{3}▸{8,13,16}▸   7.40▸    24.67¬
{16}▸    {8,13,3}▸    7.40▸    17.45¬
{8,13}▸  {3,16}▸ 7.40▸    51.39¬
{8,3}▸   {13,16}▸7.40▸    28.68¬
{8,16}▸  {13,3}▸ 7.40▸    24.50¬
{13,3}▸  {8,16}▸ 7.40▸    82.22¬
{13,16}▸{8,3}▸   7.40▸    53.62¬
{3,16}▸  {8,13}▸ 7.40▸    29.37¬
{8,13,3}▸    {16}▸    7.40▸    92.50¬
{8,13,16}▸   {3}▸7.40▸    78.72¬
{8,3,16}▸    {13}▸    7.40▸    30.83¬
{13,3,16}▸   {8}▸7.40▸    100.00¬
{1}▸{8,15,16}▸   4.00▸    13.42¬
{8}▸{1,15,16}▸   4.00▸    8.85¬
{15}▸    {1,8,16}▸    4.00▸    14.29¬
{16}▸    {1,8,15}▸    4.00▸    9.43¬
{1,8}▸   {15,16}▸4.00▸    25.97¬
{1,15}▸  {8,16}▸ 4.00▸    37.04¬
{1,16}▸  {8,15}▸ 4.00▸    24.69¬
{8,15}▸  {1,16}▸ 4.00▸    32.26¬
{8,16}▸  {1,15}▸ 4.00▸    13.25¬
{15,16}▸{1,8}▸   4.00▸    28.99¬
{1,8,15}▸    {16}▸    4.00▸    74.07¬
{1,8,16}▸    {15}▸    4.00▸    34.48¬
{1,15,16}▸   {8}▸4.00▸    66.67¬
{8,15,16}▸   {1}▸4.00▸    46.51¬
 ~/Desktop/test/output.txt   CWD: /Users/dohuni/Desktop/test   Line: 1804
```



```
gimdohyeon-ui-MacBook-Air:test dohuni$ python test.py 4 input.txt output.txt
```



test — output.txt + (~/Desktop/test) - VIM — vim — 80×37
output.txt + (~/Desktop/test) - VIM

```
1033 {13,16}▸{8,3}▸  7.40▸   53.62¬
1034 {3,16}▸ {8,13}▸ 7.40▸   29.37¬
1035 {8,13,3}▸   {16}▸   7.40▸   92.50¬
1036 {8,13,16}▸  {3}▸7.40▸   78.72¬
1037 {8,3,16}▸   {13}▸   7.40▸   30.83¬
1038 {13,3,16}▸  {8}▸7.40▸   100.00¬
1039 {9}▸{8,3,16}▸   7.20▸   25.90¬
1040 {8}▸{9,3,16}▸   7.20▸   15.93¬
1041 {3}▸{9,8,16}▸   7.20▸   24.00¬
1042 {16}▸   {9,8,3}▸7.20▸   16.98¬
1043 {9,8}▸  {3,16}▸ 7.20▸   52.17¬
1044 {9,3}▸  {8,16}▸ 7.20▸   76.60¬
1045 {9,16}▸ {8,3}▸  7.20▸   54.55¬
1046 {8,3}▸  {9,16}▸ 7.20▸   27.91¬
1047 {8,16}▸ {9,3}▸  7.20▸   23.84¬
1048 {3,16}▸ {9,8}▸  7.20▸   20.57¬
1049 {9,8,3}▸{16}▸   7.20▸   87.80¬
1050 {9,8,16}▸   {3}▸7.20▸   80.00¬
1051 {9,3,16}▸   {8}▸7.20▸   97.30¬
1052 {8,3,16}▸   {9}▸7.20▸   30.00¬
1053 {1}▸{8,3,16}▸   9.40▸   31.54¬
1054 {8}▸{1,3,16}▸   9.40▸   20.80¬
1055 {3}▸{1,8,16}▸   9.40▸   31.33¬
1056 {16}▸   {1,8,3}▸9.40▸   22.17¬
1057 {1,8}▸  {3,16}▸ 9.40▸   61.04¬
1058 {1,3}▸  {8,16}▸ 9.40▸   87.04¬
1059 {1,16}▸ {8,3}▸  9.40▸   58.02¬
1060 {8,3}▸  {1,16}▸ 9.40▸   36.43¬
1061 {8,16}▸ {1,3}▸  9.40▸   31.13¬
1062 {3,16}▸ {1,8}▸  9.40▸   37.30¬
1063 {1,8,3}▸{16}▸   9.40▸   97.92¬
1064 {1,8,16}▸   {3}▸9.40▸   81.03¬
1065 {1,3,16}▸   {8}▸9.40▸   97.92¬
1066 {8,3,16}▸   {1}▸9.40▸   39.17¬
~
 ~/Desktop/test/output.txt[+]   CWD: /Users/dohuni/Desktop/test   Line: 1066
```