2. For the PCY algorithm, create up to 5 compact hash tables. What is the difference in results and time of execution for 1,2,3,4 and 5 tables? Comment your results.

In [1]:
```
import itertools

def readdata(k, fname="groceries.csv", report=False):
    C_k = []
    b = 0

    with open("groceries.csv", "rt", encoding='utf-8') as f:
        for line in f:
            line = line.replace('\n', '')  # remove newline symbol

            if line != "":
                # gather all items in one basket
                C_k.append(line)
            else:
                # end of basket, report all itemsets
                for itemset in itertools.combinations(C_k, k):
                    yield frozenset(itemset)
                C_k = []

                # report progress
                # print every 1000th element to reduce clutter
                if report:
                    if b % 1000 == 0:
                        print('processing bin ', b)
                    b += 1

    # last basket
    if len(C_k) > 0:
        for itemset in itertools.combinations(C_k, k):
            yield frozenset(itemset)
```

In [2]:
```
import time
import numpy as np
N=50
```

In [3]:
```
# find frequent 1-tuples (individual items)
C1 = {}
for key in readdata(k=1, report=False):
    if key not in C1:
        C1[key] = 1
    else:
        C1[key] += 1

print("{} items".format(len(C1)))
```

```
7011 items
```

In [4]:
```
# filter stage
L1 = {}
for key, count in C1.items():
    if count >= N:
        L1[key] = count
print('{} items with >{} occurances'.format(len(L1), N))
```

```
8 items with >50 occurances
```

In [5]:
```
C2_items = set([a.union(b) for a in L1.keys() for b in L1.keys()])
```

In [6]:
```
len(C2_items)
```

Out[6]:
```
36
```

1-table

In [7]:
```
start_time_table1 = time.time()
# hash table
max_hash1_table1 = 5*1000000-673

H1_table1 = np.zeros((max_hash1_table1,), dtype=np.int)

for key in readdata(k=2, report=True):
    hash_cell_1_table1 = hash(key) % max_hash1_table1
    H1_table1[hash_cell_1_table1] += 1
```

In [8]:
```
# compact hash table
H_good_1_table1 = set(np.where(H1_table1 >= N)[0])

del H1_table1
```

In [9]:
```
# find frequent 2-tuples
C2_table1 = {}
for key in readdata(k=2):
    # hash-based filtering stage from PCY
    hash_cell_1_table1 = hash(key) % max_hash1_table1
    if hash_cell_1_table1 not in H_good_1_table1:
        continue


    # filter out non-frequent tuples
    if key not in C2_items:
        continue

    # record frequent tuples
    if key not in C2_table1:
        C2_table1[key] = 1
    else:
        C2_table1[key] += 1

print("{} items".format(len(C2_table1)))
```

```
36 items
```

In [10]:
```
# filter stage
L2_table1 = {}
for key, count in C2_table1.items():
    if count >= N:
        L2_table1[key] = count
print('{} items with >{} occurances'.format(len(L2_table1), N))
end_time_table1=time.time()
table1_time=end_time_table1-start_time_table1
```

```
36 items with >50 occurances
```

Generate rules A -> B

In [11]:
```
L2_table1 = [ elem for elem in list(L2_table1) if len(elem)>1] # clean our
count=1
for i in range(len(L2_table1)):

    A, B = list(L2_table1[i])
    support_AB = C2_table1[frozenset([A, B])]
    support_A = C1[frozenset([A])]
    conf_A_leads_to_B = support_AB / support_A

    support_B = C1[frozenset([B])]
    prob_B = support_B / 2750.0

    interest_A_leads_to_B = conf_A_leads_to_B - prob_B
    if interest_A_leads_to_B > 0.7:
        print("{}: {} --> {} with interest {:3f}".format(count,A, B, interes
        count+=1
```

```
1: whole milk --> rolls/buns with interest 108.960364
2: whole milk --> other vegetables with interest 61.977455
3: whole milk --> canned beer with interest 259.905455
```

```
4: whole milk --> bottled water with interest 66.975636
5: whole milk --> soda with interest 155.943273
6: newspapers --> whole milk with interest 120.956000
7: whole milk --> bottled beer with interest 119.956364
8: other vegetables --> rolls/buns with interest 108.960364
9: canned beer --> rolls/buns with interest 108.960364
10: bottled water --> rolls/buns with interest 108.960364
11: soda --> rolls/buns with interest 108.960364
12: newspapers --> rolls/buns with interest 108.960364
13: bottled beer --> rolls/buns with interest 108.960364
14: other vegetables --> canned beer with interest 259.905455
15: bottled water --> other vegetables with interest 61.977455
16: other vegetables --> soda with interest 155.943273
17: newspapers --> other vegetables with interest 61.977455
18: other vegetables --> bottled beer with interest 119.956364
19: bottled water --> canned beer with interest 259.905455
20: soda --> canned beer with interest 259.905455
21: newspapers --> canned beer with interest 259.905455
22: bottled beer --> canned beer with interest 259.905455
23: bottled water --> soda with interest 155.943273
24: newspapers --> bottled water with interest 66.975636
25: bottled water --> bottled beer with interest 119.956364
26: newspapers --> soda with interest 155.943273
27: bottled beer --> soda with interest 155.943273
```

2-table

In [12]:

```
start_time_table2 = time.time()
# hash table
max_hash1_table2 = 5*1000000-673
max_hash2_table2 = 5*1000000+673

H1_table2 = np.zeros((max_hash1_table2,), dtype=np.int)
H2_table2 = np.zeros((max_hash2_table2,), dtype=np.int)
for key in readdata(k=2, report=True):
    hash_cell_1_table2 = hash(key) % max_hash1_table2
    H1_table2[hash_cell_1_table2] += 1
    hash_cell_2_table2 = hash(key) % max_hash2_table2
    H2_table2[hash_cell_2_table2] += 1
```

In [13]:

```
# compact hash table
H_good_1_table2 = set(np.where(H1_table2 >= N)[0])
H_good_2_table2 = set(np.where(H2_table2 >= N)[0])

del H1_table2
del H2_table2
```

In [14]:
```
# find frequent 2-tuples
C2_table2 = {}
for key in readdata(k=2):
    # hash-based filtering stage from PCY
    hash_cell_1_table2 = hash(key) % max_hash1_table2
    if hash_cell_1_table2 not in H_good_1_table2:
        continue

    hash_cell_2_table2 = hash(key) % max_hash2_table2
    if hash_cell_2_table2 not in H_good_2_table2:
        continue


    # filter out non-frequent tuples
    if key not in C2_items:
        continue

    # record frequent tuples
    if key not in C2_table2:
        C2_table2[key] = 1
    else:
        C2_table2[key] += 1

print("{} items".format(len(C2_table2)))
```

```
36 items
```

In [15]:
```
# filter stage
L2_table2 = {}
for key, count in C2_table2.items():
    if count >= N:
        L2_table2[key] = count
print('{} items with >{} occurances'.format(len(L2_table2), N))
end_time_table2=time.time()
table2_time=end_time_table2-start_time_table2
```

```
36 items with >50 occurances
```

Generate rules A -> B

In [16]:

```
L2_table2 = [ elem for elem in list(L2_table2) if len(elem)>1] # clean our 
count=1
for i in range(len(L2_table2)):

    A, B = list(L2_table2[i])
    support_AB = C2_table2[frozenset([A, B])]
    support_A = C1[frozenset([A])]
    conf_A_leads_to_B = support_AB / support_A

    support_B = C1[frozenset([B])]
    prob_B = support_B / 2750.0

    interest_A_leads_to_B = conf_A_leads_to_B - prob_B

    if interest_A_leads_to_B > 0.7:
        print("{}: {} --> {} with interest {:3f}".format(count,A, B, interes
        count+=1
```

```
1: whole milk --> rolls/buns with interest 108.960364
2: whole milk --> other vegetables with interest 61.977455
3: whole milk --> canned beer with interest 259.905455
4: whole milk --> bottled water with interest 66.975636
5: whole milk --> soda with interest 155.943273
6: newspapers --> whole milk with interest 120.956000
7: whole milk --> bottled beer with interest 119.956364
8: other vegetables --> rolls/buns with interest 108.960364
9: canned beer --> rolls/buns with interest 108.960364
10: bottled water --> rolls/buns with interest 108.960364
11: soda --> rolls/buns with interest 108.960364
12: newspapers --> rolls/buns with interest 108.960364
13: bottled beer --> rolls/buns with interest 108.960364
14: other vegetables --> canned beer with interest 259.905455
15: bottled water --> other vegetables with interest 61.977455
16: other vegetables --> soda with interest 155.943273
17: newspapers --> other vegetables with interest 61.977455
18: other vegetables --> bottled beer with interest 119.956364
19: bottled water --> canned beer with interest 259.905455
20: soda --> canned beer with interest 259.905455
21: newspapers --> canned beer with interest 259.905455
22: bottled beer --> canned beer with interest 259.905455
23: bottled water --> soda with interest 155.943273
24: newspapers --> bottled water with interest 66.975636
25: bottled water --> bottled beer with interest 119.956364
26: newspapers --> soda with interest 155.943273
27: bottled beer --> soda with interest 155.943273
28: newspapers --> bottled beer with interest 119.956364
```

3-table

In [17]:

```
start_time_table3 = time.time()
# hash table
max_hash1_table3 = 5*1000000-673
max_hash2_table3 = 5*1000000+673
max_hash3_table3 = 5*1000000+673

H1_table3 = np.zeros((max_hash1_table3,), dtype=np.int)
H2_table3 = np.zeros((max_hash2_table3,), dtype=np.int)
H3_table3 = np.zeros((max_hash3_table3,), dtype=np.int)

for key in readdata(k=2, report=True):
    hash_cell_1_table3 = hash(key) % max_hash1_table3
    H1_table3[hash_cell_1_table3] += 1
    hash_cell_2_table3 = hash(key) % max_hash2_table3
    H2_table3[hash_cell_2_table3] += 1
    hash_cell_3_table3 = hash(key) % max_hash3_table3
    H3_table3[hash_cell_3_table3] += 1
```

In [18]:

```
# compact hash table
H_good_1_table3 = set(np.where(H1_table3 >= N)[0])
H_good_2_table3 = set(np.where(H2_table3 >= N)[0])
H_good_3_table3 = set(np.where(H3_table3 >= N)[0])

del H1_table3
del H2_table3
del H3_table3
```

In [19]:
```
# find frequent 2-tuples
C2_table3 = {}
for key in readdata(k=2):
    # hash-based filtering stage from PCY
    hash_cell_1_table3 = hash(key) % max_hash1_table3
    if hash_cell_1_table3 not in H_good_1_table3:
        continue

    hash_cell_2_table3 = hash(key) % max_hash2_table3
    if hash_cell_2_table3 not in H_good_2_table3:
        continue

    hash_cell_3_table3 = hash(key) % max_hash3_table3
    if hash_cell_3_table3 not in H_good_3_table3:
        continue


    # filter out non-frequent tuples
    if key not in C2_items:
        continue

    # record frequent tuples
    if key not in C2_table3:
        C2_table3[key] = 1
    else:
        C2_table3[key] += 1

print("{} items".format(len(C2_table3)))
```

```
36 items
```

In [20]:
```
# filter stage
L2_table3 = {}
for key, count in C2_table3.items():
    if count >= N:
        L2_table3[key] = count
print('{} items with >{} occurances'.format(len(L2_table3), N))
end_time_table3=time.time()
table3_time=end_time_table3-start_time_table3
```

```
36 items with >50 occurances
```

Generate rules A -> B

In [21]:

```
L2_table3 = [ elem for elem in list(L2_table3) if len(elem)>1] # clean our 
count=1
for i in range(len(L2_table3)):

    A, B = list(L2_table3[i])
    support_AB = C2_table3[frozenset([A, B])]
    support_A = C1[frozenset([A])]
    conf_A_leads_to_B = support_AB / support_A

    support_B = C1[frozenset([B])]
    prob_B = support_B / 2750.0

    interest_A_leads_to_B = conf_A_leads_to_B - prob_B

    if interest_A_leads_to_B > 0.7:
        print("{}: {} --> {} with interest {:3f}".format(count,A, B, interes
        count+=1
```

```
1: whole milk --> rolls/buns with interest 108.960364
2: whole milk --> other vegetables with interest 61.977455
3: whole milk --> canned beer with interest 259.905455
4: whole milk --> bottled water with interest 66.975636
5: whole milk --> soda with interest 155.943273
6: newspapers --> whole milk with interest 120.956000
7: whole milk --> bottled beer with interest 119.956364
8: other vegetables --> rolls/buns with interest 108.960364
9: canned beer --> rolls/buns with interest 108.960364
10: bottled water --> rolls/buns with interest 108.960364
11: soda --> rolls/buns with interest 108.960364
12: newspapers --> rolls/buns with interest 108.960364
13: bottled beer --> rolls/buns with interest 108.960364
14: other vegetables --> canned beer with interest 259.905455
15: bottled water --> other vegetables with interest 61.977455
16: other vegetables --> soda with interest 155.943273
17: newspapers --> other vegetables with interest 61.977455
18: other vegetables --> bottled beer with interest 119.956364
19: bottled water --> canned beer with interest 259.905455
20: soda --> canned beer with interest 259.905455
21: newspapers --> canned beer with interest 259.905455
22: bottled beer --> canned beer with interest 259.905455
23: bottled water --> soda with interest 155.943273
24: newspapers --> bottled water with interest 66.975636
25: bottled water --> bottled beer with interest 119.956364
26: newspapers --> soda with interest 155.943273
27: bottled beer --> soda with interest 155.943273
28: newspapers --> bottled beer with interest 119.956364
```

4-table

In [22]:

```
start_time_table4 = time.time()
# hash table
max_hash1_table4 = 5*1000000-673
max_hash2_table4 = 5*1000000+673
max_hash3_table4 = 5*1000000+673
max_hash4_table4 = 5*1000000+673

H1_table4 = np.zeros((max_hash1_table4,), dtype=np.int)
H2_table4 = np.zeros((max_hash2_table4,), dtype=np.int)
H3_table4 = np.zeros((max_hash3_table4,), dtype=np.int)
H4_table4 = np.zeros((max_hash4_table4,), dtype=np.int)

for key in readdata(k=2, report=True):
    hash_cell_1_table4 = hash(key) % max_hash1_table4
    H1_table4[hash_cell_1_table4] += 1
    hash_cell_2_table4 = hash(key) % max_hash2_table4
    H2_table4[hash_cell_2_table4] += 1
    hash_cell_3_table4 = hash(key) % max_hash3_table4
    H3_table4[hash_cell_3_table4] += 1
    hash_cell_4_table4 = hash(key) % max_hash4_table4
    H4_table4[hash_cell_4_table4] += 1
```

In [23]:

```
# compact hash table
H_good_1_table4 = set(np.where(H1_table4 >= N)[0])
H_good_2_table4 = set(np.where(H2_table4 >= N)[0])
H_good_3_table4 = set(np.where(H3_table4 >= N)[0])
H_good_4_table4 = set(np.where(H4_table4 >= N)[0])

del H1_table4
del H2_table4
del H3_table4
del H4_table4
```

In [24]:
```
# find frequent 2-tuples
C2_table4 = {}
for key in readdata(k=2):
    # hash-based filtering stage from PCY
    hash_cell_1_table4 = hash(key) % max_hash1_table4
    if hash_cell_1_table4 not in H_good_1_table4:
        continue

    hash_cell_2_table4 = hash(key) % max_hash2_table4
    if hash_cell_2_table4 not in H_good_2_table4:
        continue

    hash_cell_3_table4 = hash(key) % max_hash3_table4
    if hash_cell_3_table4 not in H_good_3_table4:
        continue
    hash_cell_4_table4 = hash(key) % max_hash4_table4
    if hash_cell_4_table4 not in H_good_4_table4:
        continue


    # filter out non-frequent tuples
    if key not in C2_items:
        continue

    # record frequent tuples
    if key not in C2_table4:
        C2_table4[key] = 1
    else:
        C2_table4[key] += 1

print("{} items".format(len(C2_table4)))
```

```
36 items
```

In [25]:
```
# filter stage
L2_table4 = {}
for key, count in C2_table4.items():
    if count >= N:
        L2_table4[key] = count
print('{} items with >{} occurances'.format(len(L2_table4), N))
end_time_table4=time.time()
table4_time=end_time_table4-start_time_table4
```

```
36 items with >50 occurances
```

Generate rules A -> B

In [26]:

```
L2_table4 = [ elem for elem in list(L2_table4) if len(elem)>1] # clean our 
count=1
for i in range(len(L2_table4)):

    A, B = list(L2_table4[i])
    support_AB = C2_table4[frozenset([A, B])]
    support_A = C1[frozenset([A])]
    conf_A_leads_to_B = support_AB / support_A

    support_B = C1[frozenset([B])]
    prob_B = support_B / 2750.0

    interest_A_leads_to_B = conf_A_leads_to_B - prob_B

    if interest_A_leads_to_B > 0.7:
        print("{}: {} --> {} with interest {:3f}".format(count,A, B, interes
        count+=1
```

```
1: whole milk --> rolls/buns with interest 108.960364
2: whole milk --> other vegetables with interest 61.977455
3: whole milk --> canned beer with interest 259.905455
4: whole milk --> bottled water with interest 66.975636
5: whole milk --> soda with interest 155.943273
6: newspapers --> whole milk with interest 120.956000
7: whole milk --> bottled beer with interest 119.956364
8: other vegetables --> rolls/buns with interest 108.960364
9: canned beer --> rolls/buns with interest 108.960364
10: bottled water --> rolls/buns with interest 108.960364
11: soda --> rolls/buns with interest 108.960364
12: newspapers --> rolls/buns with interest 108.960364
13: bottled beer --> rolls/buns with interest 108.960364
14: other vegetables --> canned beer with interest 259.905455
15: bottled water --> other vegetables with interest 61.977455
16: other vegetables --> soda with interest 155.943273
17: newspapers --> other vegetables with interest 61.977455
18: other vegetables --> bottled beer with interest 119.956364
19: bottled water --> canned beer with interest 259.905455
20: soda --> canned beer with interest 259.905455
21: newspapers --> canned beer with interest 259.905455
22: bottled beer --> canned beer with interest 259.905455
23: bottled water --> soda with interest 155.943273
24: newspapers --> bottled water with interest 66.975636
25: bottled water --> bottled beer with interest 119.956364
26: newspapers --> soda with interest 155.943273
27: bottled beer --> soda with interest 155.943273
28: newspapers --> bottled beer with interest 119.956364
```

5-table

In [27]:

```
start_time_table5 = time.time()
# hash table
max_hash1_table5 = 5*1000000-673
max_hash2_table5 = 5*1000000+673
max_hash3_table5 = 5*1000000+673
max_hash4_table5 = 5*1000000+673
max_hash5_table5 = 5*1000000+673

H1_table5 = np.zeros((max_hash1_table5,), dtype=np.int)
H2_table5 = np.zeros((max_hash2_table5,), dtype=np.int)
H3_table5 = np.zeros((max_hash3_table5,), dtype=np.int)
H4_table5 = np.zeros((max_hash4_table5,), dtype=np.int)
H5_table5 = np.zeros((max_hash5_table5,), dtype=np.int)

for key in readdata(k=2, report=True):
    hash_cell_1_table5 = hash(key) % max_hash1_table5
    H1_table5[hash_cell_1_table5] += 1
    hash_cell_2_table5 = hash(key) % max_hash2_table5
    H2_table5[hash_cell_2_table5] += 1
    hash_cell_3_table5 = hash(key) % max_hash3_table5
    H3_table5[hash_cell_3_table5] += 1
    hash_cell_4_table5 = hash(key) % max_hash4_table5
    H4_table5[hash_cell_4_table5] += 1
    hash_cell_5_table5 = hash(key) % max_hash5_table5
    H5_table5[hash_cell_5_table5] += 1
```

In [28]:

```
# compact hash table
H_good_1_table5 = set(np.where(H1_table5 >= N)[0])
H_good_2_table5 = set(np.where(H2_table5 >= N)[0])
H_good_3_table5 = set(np.where(H3_table5 >= N)[0])
H_good_4_table5 = set(np.where(H4_table5 >= N)[0])
H_good_5_table5 = set(np.where(H5_table5 >= N)[0])

del H1_table5
del H2_table5
del H3_table5
del H4_table5
del H5_table5
```

In [29]:
```
# find frequent 2-tuples
C2_table5 = {}
for key in readdata(k=2):
    # hash-based filtering stage from PCY
    hash_cell_1_table5 = hash(key) % max_hash1_table5
    if hash_cell_1_table5 not in H_good_1_table5:
        continue

    hash_cell_2_table5 = hash(key) % max_hash2_table5
    if hash_cell_2_table5 not in H_good_2_table5:
        continue

    hash_cell_3_table5 = hash(key) % max_hash3_table5
    if hash_cell_3_table5 not in H_good_3_table5:
        continue
    hash_cell_4_table5 = hash(key) % max_hash4_table5
    if hash_cell_4_table5 not in H_good_4_table5:
        continue
    hash_cell_5_table5 = hash(key) % max_hash5_table5
    if hash_cell_5_table5 not in H_good_5_table5:
        continue


    # filter out non-frequent tuples
    if key not in C2_items:
        continue

    # record frequent tuples
    if key not in C2_table5:
        C2_table5[key] = 1
    else:
        C2_table5[key] += 1

print("{} items".format(len(C2_table5)))
```

```
36 items
```

In [30]:
```
# filter stage
L2_table5 = {}
for key, count in C2_table5.items():
    if count >= N:
        L2_table5[key] = count
print('{} items with >{} occurances'.format(len(L2_table5), N))
end_time_table5=time.time()
table5_time=end_time_table5-start_time_table5
```

```
36 items with >50 occurances
```

Generate rules A -> B

In [31]:
```
L2_table5 = [ elem for elem in list(L2_table5) if len(elem)>1] # clean our 
count=1
for i in range(len(L2_table5)):

    A, B = list(L2_table5[i])
    support_AB = C2_table5[frozenset([A, B])]
    support_A = C1[frozenset([A])]
    conf_A_leads_to_B = support_AB / support_A

    support_B = C1[frozenset([B])]
    prob_B = support_B / 2750.0

    interest_A_leads_to_B = conf_A_leads_to_B - prob_B

    if interest_A_leads_to_B > 0.7:
        print("{}: {} --> {} with interest {:3f}".format(count,A, B, interes
        count+=1
```

```
1: whole milk --> rolls/buns with interest 108.960364
2: whole milk --> other vegetables with interest 61.977455
3: whole milk --> canned beer with interest 259.905455
4: whole milk --> bottled water with interest 66.975636
5: whole milk --> soda with interest 155.943273
6: newspapers --> whole milk with interest 120.956000
7: whole milk --> bottled beer with interest 119.956364
8: other vegetables --> rolls/buns with interest 108.960364
9: canned beer --> rolls/buns with interest 108.960364
10: bottled water --> rolls/buns with interest 108.960364
11: soda --> rolls/buns with interest 108.960364
12: newspapers --> rolls/buns with interest 108.960364
13: bottled beer --> rolls/buns with interest 108.960364
14: other vegetables --> canned beer with interest 259.905455
15: bottled water --> other vegetables with interest 61.977455
16: other vegetables --> soda with interest 155.943273
17: newspapers --> other vegetables with interest 61.977455
18: other vegetables --> bottled beer with interest 119.956364
19: bottled water --> canned beer with interest 259.905455
20: soda --> canned beer with interest 259.905455
21: newspapers --> canned beer with interest 259.905455
22: bottled beer --> canned beer with interest 259.905455
23: bottled water --> soda with interest 155.943273
24: newspapers --> bottled water with interest 66.975636
25: bottled water --> bottled beer with interest 119.956364
26: newspapers --> soda with interest 155.943273
27: bottled beer --> soda with interest 155.943273
28: newspapers --> bottled beer with interest 119.956364
```

Result

In [32]:
```
print(f'table 1:\n   time: {table1_time} sec,\n   items {len(C2_table1)}')
print(f'table 2:\n   time: {table2_time} sec,\n   items {len(C2_table2)}')
print(f'table 3:\n   time: {table3_time} sec,\n   items {len(C2_table3)}')
print(f'table 4:\n   time: {table4_time} sec,\n   items {len(C2_table4)}')
print(f'table 5:\n   time: {table5_time} sec,\n   items {len(C2_table5)}')
print()
```

```
table 1:
   time: 252.1904172897339 sec,
```

```
   items 36
table 2:
   time: 422.3184566497803 sec,
   items 36
table 3:
   time: 536.6535396575928 sec,
   items 36
table 4:
   time: 600.8096449375153 sec,
   items 36
table 5:
   time: 709.8312077522278 sec,
   items 36
```

In [ ]: