Frequent itemsets

Question 1:

Suppose we have transactions that satisfy the following assumptions:

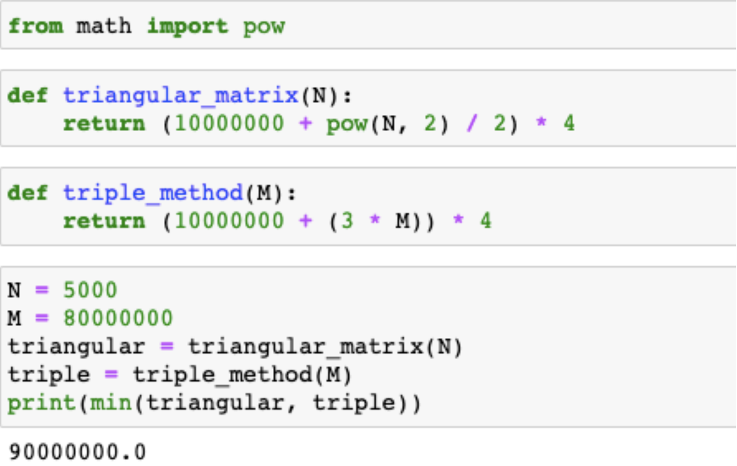
- *s*, the support threshold, is 10,000.
- There are one million items, which are represented by the integers 0,1,...,999999.
- There are *N* frequent items, that is, items that occur 10,000 times or more.
- There are one million pairs that occur 10,000 times or more.
- There are 2*M* pairs that occur exactly once. *M* of these pairs consist of two frequent items, the other *M* each have at least one nonfrequent item.
- No other pairs occur at all.
- Integers are always represented by 4 bytes.

Suppose we run the a-priori algorithm to find frequent pairs and can choose on the second pass between the triangular-matrix method for counting candidate pairs (a triangular array count[i][j] that holds an integer count for each pair of items (i, j) where i < j) and a hash table of item-item-count triples.

Neglect in the first case the space needed to translate between original item numbers and numbers for the frequent items, and in the second case neglect the space needed for the hash table. Assume that item numbers and counts are always 4-byte integers.

As a function of *N* and *M*, what is the minimum number of bytes of main memory needed to execute the a-priori algorithm on this data?

Ans:



```
from math import pow
```

```
def triangular_matrix(N):
    return (10000000 + pow(N, 2) / 2) * 4
```

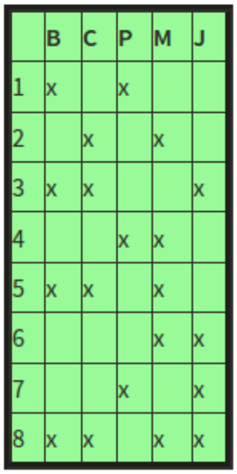
```
def triple_method(M):
    return (10000000 + (3 * M)) * 4
```

```
N = 5000
M = 80000000
triangular = triangular_matrix(N)
triple = triple_method(M)
print(min(triangular, triple))
```

```
90000000.0
```

Question 2:

Below is a table representing eight transactions and five items: Beer, Coke, Pepsi, Milk, and Juice. The items are represented by their first letters; e.g., "M" = milk. An "x" indicates membership of the item in the transaction.



	B	C	P	M	J
1	x		x		
2		x		x	
3	x	x			x
4			x	x	
5	x	x		x	
6				x	x
7			x		x
8	x	x		x	x

Compute the support for each of the 10 pairs of items. If the support threshold is 2, find out the pairs that are frequent itemsets.

Ans:

Support for each pairs of items:

s({B,C}) = 3, s({B,P}) = 1, s({B,M}) = 2, s({B,J}) = 2, s({C,P}) = 0, s({C,M}) = 3, s({C,J}) = 2, s({P,M}) = 1,s({P,J}) =1, s({M,J}) =2

Frequent item sets:

s({B,C}) = 3, s({B,M}) = 2, s({B,J}) = 2, s({C,P}) = 0, s({C,M}) = 3, s({C,J}) = 2, s({M,J}) =2

Question 3:

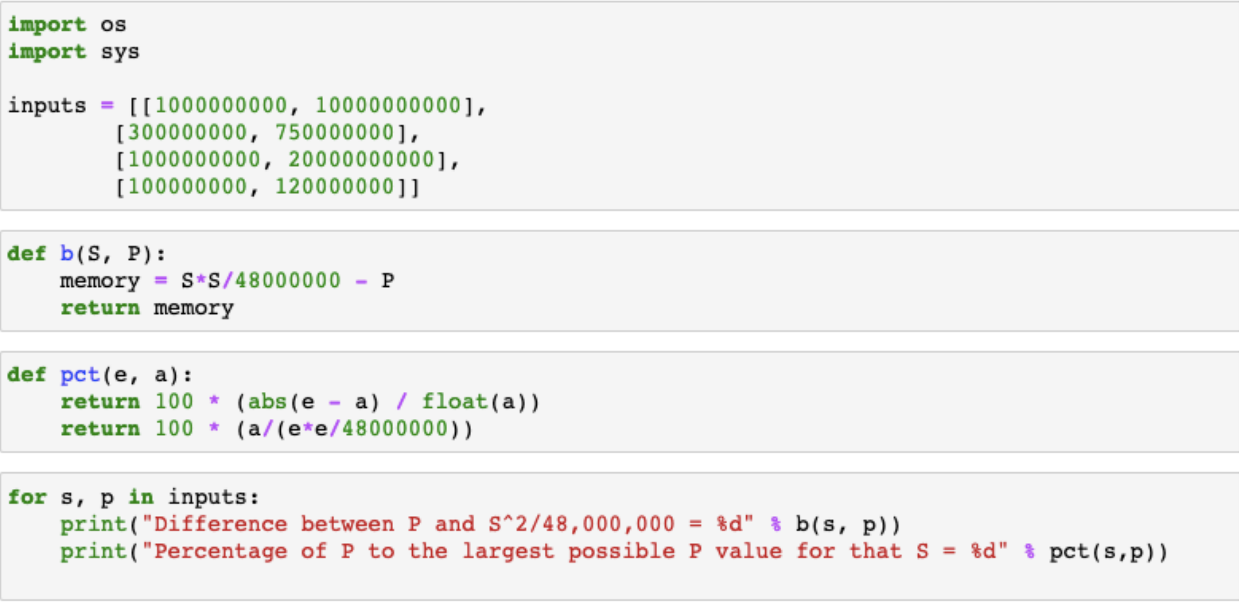
Suppose we perform the PCY algorithm to find frequent pairs, with market-basket data meeting the following specifications:

- *s*, the support threshold, is 10,000.

- There are one million items, which are represented by the integers 0,1,...,999999.
- There are 250,000 frequent items, that is, items that occur 10,000 times or more.
- There are one million pairs that occur 10,000 times or more.
- There are *P* pairs that occur exactly once and consist of 2 frequent items.
- No other pairs occur at all.
- Integers are always represented by 4 bytes.
- When we hash pairs, they distribute among buckets randomly, but as evenly as possible; i.e., you may assume that each bucket gets exactly its fair share of the *P* pairs that occur once.

Suppose there are *S* bytes of main memory. In order to run the PCY algorithm successfully, the number of buckets must be sufficiently large that most buckets are not frequent. In addition, on the second pass, there must be enough room to count all the candidate pairs. As a function of *S*, what is the largest value of *P* for which we can successfully run the PCY algorithm on this data? Find out the value for *S* and value for *P* that is approximately (i.e., to within 10%) the largest possible value of *P* for that *S*.

Ans:



```
import os
import sys

inputs = [[1000000000, 10000000000],
        [300000000, 750000000],
        [1000000000, 20000000000],
        [100000000, 120000000]]
```

```
def b(S, P):
    memory = S*S/48000000 - P
    return memory
```

```
def pct(e, a):
    return 100 * (abs(e - a) / float(a))
    return 100 * (a/(e*e/48000000))
```

```
for s, p in inputs:
    print("Difference between P and S^2/48,000,000 = %d" % b(s, p))
    print("Percentage of P to the largest possible P value for that S = %d" % pct(s,p))
```

Question 4: During a run of Toivonen's Algorithm with set of items {A,B,C,D,E,F,G,H} a sample is found to have the following maximal frequent itemsets: {A,B}, {A,C}, {A,D}, {B,C}, {E}, {F}. Compute the negative border.

Ans:

Given a set of items {A,B,C,D,E,F,G,H}

frequent itemsets: {A,B}, {A,C}, {A,D}, {B,C}, {E}, {F}

{A}, {B}, {C}, {D}, {G}, {H} are in the negative border, because it is not frequent in the sample, but its only immediate subset, ∅, is frequent.

Hence, negative border = {A}, {B}, {C}, {D}, {G}, {H}, **{E,F}, {A,B,C}, {A,B,D}**