VI. (12 pts) 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 large. 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? Demonstrate that you have the correct formula by indicating which of the following is a value for *S* and a value for *P* that is approximately (i.e., to within 10%) the largest possible value of *P* for that *S*.

Possible pairs:

i) S = 500,000,000 P = 10,000,000,000
ii) S = 1,000,000,000 P = 10,000,000,000
iii) S = 500,000,000 P = 5,000,000,000
iv) S = 200,000,000 P = 1,600,000,000
v) S = 300,000,000 P = 3,500,000,000

Discussion:

- Unlike the example in the textbook you don't know how many baskets there are and you don't know how many items are in each basket.
- We're interested in approximations accurate to within 10%. Therefore, if you come up with a value such as S/4 – 1,000,000 in your work, you can simplify that to S/4 since S/4 will be much larger than 1,000,000. In general, any term less than 10% of a sum can be ignored.
- You may want to figure out (approximately) how many of the P *infrequent* pairs will hash to a bucket.
- You may want to figure out (approximately) how many pairs will hash to a *frequent bucket*.
- You may want to figure out (approximately) how many candidate pairs will need to be considered in the second pass.
- As a simplification, ignore the space needed to store the bitmap between passes and ignore the space needed to store the pairs of frequent items on the second pass. Determine the size of the hash table needed in the second pass. That will be the dominant factor.