CSE 4502/5717 Big Data Analytics
Fall 2024; Homework 3 Solutions

1. (a) Note that for a pair of items to be frequent, it has to occur in at least one transaction. Thus the only two-itemsets we have to generate as candidates are the pairs of items we can generate from each transaction. As a result, the number of candidates is $O(n)$. For each such candidate we have to compute the support.

   Support for the candidates can be computed as described in class. In particular, we use a hash tree to store all the candidates. The expected size of each leaf in the hash tree is $O(1)$. Then, we do the following:

   ```
   for each transaction $T$ in DB do
       for each pair $(i, i')$ of items in $T$ do
           Use the hash tree to increment of the support of $(i, i')$ by 1.
   Output all the pairs of items that have a support of $\geq minSupport$.
   ```

   It is clear that the above algorithm has an expected run time of $O(n)$.

   (b) Note that if X is an itemset with k items, then we can form $2^k - 2$ association rules using X. This in turn means that the total number of association rules we can form from a set I of d items is $\sum_{k=2}^{d} \binom{d}{k}(2^k - 2) = \sum_{k=2}^{d} \binom{d}{k} 2^k - 2\sum_{k=2}^{d} \binom{d}{k}$ $= 3^d - 2d - 1 - 2(2^d - d - 1) = 3^d - 2^{d+1} + 1$.

2. Let the transactions in the database be $t_1, t_2, \ldots, t_q$. Note that any item in the database can be represented as an integer in the range $[1, n^c]$.

   ```
   $S$ is an empty sequence to begin with;
   for $i = 1$ to $q$ do
       for every item $a \in t_i$ do
           Add $a$ to the sequence $S$;
   Sort $S$ using the integer sorting algorithm;
   Scan through the sorted sequence to count the support for each item and
   output those that have enough support.
   ```

   Clearly, the above algorithm runs in $O(n)$ time.

3. Note that for a pair of items to be frequent, it has to occur in at least one transaction. Thus the only two itemsets we have to generate as candidates are the pairs of items we can generate from each transaction. As a result, the number of candidates is $O(n)$. For each such candidate we have to compute the support.

Let the candidates be $(i_1^1, i_2^1), (i_1^2, i_2^2), \ldots, (i_1^q, i_2^q)$, where $q = O(n)$. We keep any such 2-itemset in sorted order (i.e., $i_1^k < i_2^k$, for $1 \leq k \leq q$). We can think of each such candidate as an integer in the range $[1, n^{2c}]$. We sort these 2-itemsets in lexicographic order using the integer sorting algorithm. This will take $O(n)$ time. We scan through the sorted sequence, identify all the frequent 2-itemsets, and output them.

4. The coefficients of the polynomial are given by $\binom{n}{i}$, $i = 0, 1, \ldots, n$.
   Since $\binom{n}{i} = \binom{n}{i-1}$ $(n - i + 1)/i$, the coefficients can be computed in time $O(n)$.
   FFT can be used to multiply two nth degree polynomials in $O(n \log n)$ time. We can compute the coefficients of $(1+x)^n$ by multiplying $(1+x)^{n/2}$ and $(1+x)^{n/2}$. If $T(n)$ is the time needed to compute $(1+x)^n$, then, $T(n) = T(n/2) + O(n \log n)$, which solves to $O(n \log n)$.

5. Let A be a *Toeplitz* matrix and B be an $n \times 1$ vector. Let's consider the multiplication of the lower triangular part of A (including the main diagonal elements) with B.
   Let the elements of A be the following:
   $a_{n,n} = a_{n-1,n-1} = a_{n-2,n-2} = \ldots = a_{2,2} = a_{1,1} = a_1$
   $a_{n,n-1} = a_{n-1,n-2} = a_{n-2,n-3} = \ldots = a_{2,1} = a_2$
   $a_{n,n-2} = a_{n-1,n-3} = a_{n-2,n-4} = \ldots = a_{3,1} = a_3$
   $\vdots$
   $a_{n,1} = a_n$
   Let the elements of B be the following:
   $b_{1,1} = b_1, b_{2,1} = b_2, \ldots, b_{n,1} = b_n$
   Multiplication of the lower triangular part of A with B gives the following:
   $$\begin{bmatrix} a_1b_1 \\ a_2b_1 + a_1b_2 \\ a_3b_1 + a_2b_2 + a_1b_3 \\ \vdots \end{bmatrix}$$
   We can notice that the above is nothing but the multiplication of two polynomials $(a_1 + a_2x + a_3x^2 + \ldots)$ and $(b_1 + b_2x + b_3x^2 + \ldots)$.
   Since the plolynomials can be multiplied in $O(n \log n)$ time, the matrices can also be multiplied in $O(n \log n)$ time. Multiplication of the upper triangular elements of A with B is symmetrical to the above and it would not affect the asymptotic complexity.

6. Convert the polynomial $f(x)$, into the following form (into n/m groups):
   $$f(x) = x^{n-m}(a_nx^m + a_{n-1}x^{m-1} + \ldots + a_{n-m+1}x) +$$
   $$x^{n-2m}(a_{n-m}x^m + a_{n-m-1}x^{m-1} + \ldots + a_{n-2m+1}x) +$$
   $$\vdots$$

$$\begin{aligned}&(a_m x^m + a_{m-1}x^{m-1} + \ldots + a_1 x + a_0)\\ &= x^{n-m} f_1(x) + x^{n-2m} f_2(x) + \ldots + f_{n/m}(x)\end{aligned}$$

where $f_1(x)$, $f_2(x)$, $\ldots$, $f_{n/m}(x)$ are polynomials of degree m.

$f(x)g(x) = x^{n-m} f_1(x)g(x) + x^{n-2m} f_2(x)g(x) + \ldots + f_{n/m}(x)g(x)$

$f(x)g(x)$ requires multiplying (n/m) pairs of polynomials of degree m each and combining the n/m resulting polynomials into one polynomial of degree $n+m$. The second step requires a scan of all the n/m polynomials of degree $2m$.

Total time taken is $O(\frac{n}{m}(m \log m)) + O(\frac{n}{m}(2m)) = O(n \log m)$.