CSE 4502/5717 Big Data Analytics. Spring 2026

Model Exam III Solutions

1. Since each item of I occurs in $\leq c$ transactions, it follows that $\sum_{t \in DB} |t| \leq cd$, i.e., $\sum_{t \in DB} |t| = O(d)$. We form a sequence S of all the items in all the transactions in DB. Note that $|S| = O(d)$. Also, we can think of each item as an integer in the range $[1, d]$. Thus we can sort S in $O(d)$ time. We scan through the sorted S to identify all the frequent items. The total runtime will be $O(d)$.

2. Pick a random sample S of s transactions from DB (The value of s will be finalized in the analysis). Let q and m be the number of transactions in DB and S, respectively, that contain X. m can be computed in $O(sk)$ time. We output $\mu = m\frac{n}{s}$. The expected value of m is $s\frac{q}{n}$. Using Chernoff bounds, $Prob.\left[m \geq 1.1\frac{sq}{n}\right] \leq \exp\left(-0.01\frac{sq}{3n}\right)$. This probability will be $\leq \frac{n^{-\alpha}}{2}$ if $s \geq 600\alpha\sqrt{n}\log_e n$. Also, using Chernoff bounds, $Prob.\left[m \leq 0.9\frac{sq}{n}\right] \leq \exp\left(-0.01\frac{sq}{2n}\right)$. This probability will be $\leq \frac{n^{-\alpha}}{2}$ if $s \geq 400\alpha\sqrt{n}\log_e n$.

   In summary, if we pick s to be $\geq 600\alpha\sqrt{n}\log_e n$, μ will be in the interval $[0.9q,\ 1.1q]$ with a probability of $\geq (1 - n^{-\alpha})$. The tun time of this algorithm will be $O(k\sqrt{n}\log n)$.

3. Note that there are $\binom{d}{k} < d^k$ possible k-itemsets. We can generate all possible k-itemsets in $O(1)$ time using $\binom{d}{k}$ processors. These are the candidates. Followed by this, we count the support for each possible k-itemset. We assign $\frac{n}{\log n}$ processors for each k-itemset candidate. The support can be computed in $O(\log n)$ time using a prefix computation. Details follow.

   1) **for** each candidate c **in parallel do**
   2) **for** each transaction t **in parallel do**
   3) Processor $P_{c,t}$ computes $b_{c,t}$ as 1 if c is in t and zero otherwise;
   4) $\frac{n}{\log n}$ processors perform a prefix sums computation on $b_{c,1}, b_{c,2}, \ldots, b_{c,n}$.
   5) If this sum is $\geq minSupport$, output c as a frequent k-itemset;

   Analysis: All the candidates can be generated in $O(1)$ time. For a given candidate c, if we have n processors, we can complete step 3 in $O(1)$ time. Using the slow-down lemma, this can be done in $O(\log n)$ time using $\frac{n}{\log n}$ processors. Step 4 takes $O(\log n)$ time and step 5 takes $O(1)$ time. Thus the total run time is $O(\log n)$.

4. Here are the steps in the hierarchical clustering algorithm:

   (a) Compute the distance matrix $D[1:n,\ 1:n]$. This can be done in $O(1)$ time using n^2 processors. Using the slow-down lemma, this can also be done in $O(n \log n)$ time using $\frac{n}{\log n}$ processors.

   (b) For each row compute the row minimum and store it in array $A[1:n]$. We do this for each row in sequence. For any given row, we can find the minimum in $O(\log n)$ time with $\frac{n}{\log n}$ processors using a prefix computation. Thus the computation for all the rows can be completed in $O(n \log n)$ time using $\frac{n}{\log n}$ processors.

(c) From there on we find the minimum in the array $A[1:n]$. This takes $O(\log n)$ time using $\frac{n}{\log n}$ processors. As a result, we identify the two clusters i and j that have to be merged. We then merge these clusters by updating row i. The updating also takes $O(\log n)$ time and $\frac{n}{\log n}$ processors. This process is repeated at most n times and hence the total runtime will be $O(n \log n)$.

In summary, the entire algorithm runs in $O(n \log n)$ time using $\frac{n}{\log n}$ processors.

5. At the beginning the state of the register is $|0000\rangle$. After the application of the two Hadamard gates, we get: $\frac{1}{2}|0000\rangle + \frac{1}{2}|0010\rangle + \frac{1}{2}|1000\rangle + \frac{1}{2}|1010\rangle$. When we apply the CCNOT gate, the state does not change. When we apply the next two Hadamard gates, the state becomes a uniform superposition: $\frac{1}{4}\sum_{i=0}^{15}|i\rangle$. The following CCNOT gate does not change the state. Thus, the final output is: $\frac{1}{4}\sum_{i=0}^{15}|i\rangle$.

6. Here the idea is to pick a random element and keep it as the current minimum m'. Use the Grover's algorithm to find an element m that is less than m' (if there is one). Update m' to m and repeat this process until no element can be found that is less than the current minimum.

   For the first time when we pick a random element m', the expected number of elements that are less than m' will be $n/2$. Thus Grover's algorithm will take an expected $O(\sqrt{\frac{n}{n/2}})$ time to find an element m that is less than m'. The expected number of elements that are less than m in X will be $n/4$, and so on. Thus the expected runtime of the entire algorithm will be $O\left(\sqrt{2} + \sqrt{4} + \cdots + \sqrt{n}\right) = O(\sqrt{n})$.