Big Data Computing

Homework 3

Pietro Spadaccino

Assignment 1

Definitions: Let $G(V, E)$ a connected graph and $M(x, h)$ the set of nodes within h hops from $x \in V$.

The exact algorithm for calculating $M(x, h)$ is based on this observation:

$$(x, y) \in E \implies M(x, h) \supseteq M(y, h-1)$$

In words, if there exists an edge between (x, y) then the nodes within h steps from x must include nodes within $h-1$ steps from y.

By definition of diameter d we know $d = \min_h\{h \,|\, M(x, h) = V,\ \forall x\}$, and since ANF algorithm expands $M(x, h)$ iteratively for each h, going from $M(x, 0) = \{x\}$ to $M(x, h \geq d) = V$, we can rewrite it as:

$$d = \min_h\{h \,|\, M(x, h) = M(y, h),\ \forall x, y\}$$

Hence we can modify ANF by adding a check at the end of the h for-loop iteration: if $M(x, h) = M(y, h)\, \forall x, y$ then the diameter is $d = h$.

Assignment 2

Let be $h_1, \ldots, h_s$ ideal min-hashing functions and $res = [res_1, ..., res_s]$ be our sample.

Algorithm 1: Sample distinct

```
Result: Sample res = [res_1, ..., res_s]
Initialize res with first s distinct items;
Let max_i ← h_i(res_i) for i ∈ {1, ..., s};
while item ← stream.next() do
    for i ∈ {1, ..., s} do
        if h_i(item) > max_i then
            max_i ← h_i(item);
            res_i ← item;
```

The algorithm starts by populating the sample with the first s distinct items. It keeps variables max_i indicating, for each hash function h_i, the maximum hash value obtained so far. Each incoming item from the stream is hashed with h_i and if the hash value is greater than max_i then the item is added to the sample at position i. This is done for all hash functions h_i, $i \in \{1, \ldots, s\}$.

Observation 1: Let I be the set of distinct items observed so far and $n = |I|$, then for any $x \in I$ and $i \in \{1, \ldots, s\}$ we have $P(x = res_i) = P[x = \arg\max_{x'} h_i(x')] = \frac{1}{n}$, since the hashing functions are ideal and items have the same probability to maximize hash values.

Observation 2: Since outcomes of the hash functions are independent:

$$P(x \notin res) = P(x \neq res_1 \cap \ldots \cap x \neq res_s) = \left(1 - \frac{1}{n}\right)^s$$

Claim: After observing $n \geq s$ distinct items, the probability of having one of them in the sample is $\approx \frac{s}{n}$.

Proof:

$$P(x \in res) = 1 - P(x \notin res) = 1 - \left(1 - \frac{1}{n}\right)^s \approx 1 - \left(1 - \frac{s}{n}\right) = \frac{s}{n}$$

$\square$

Using the union bound, we find that the approximated value is an upper bound:

$$P(x \in res) = P\left(\bigcup_{i=1}^{s} x = res_i\right) \leq \sum_{i=1}^{s} P(x = res_i) = \frac{s}{n}$$

If the events $x = res_i$ were mutually exclusive it would have been an equality: this makes us understand that the approximation error derives from the intersection of these events or, in other words, the possibility for items to appear $k > 1$ times in the sample. It can be expressed as a binomial distribution:

$$P(x \text{ is } k \text{ times in } res) = \binom{s}{k}\left(\frac{1}{n}\right)^k\left(1 - \frac{1}{n}\right)^{s-k}$$

As n increases, this probability goes to 0 for any $k > 1$:

$$\binom{s}{k}\left(\frac{1}{n}\right)^k\left(1 - \frac{1}{n}\right)^{s-k} < \left(\frac{s}{n}\right)^k\left(1 - \frac{1}{n}\right)^{s-k} \leq \left(\frac{s}{n}\right)^k \approx 0, \text{ for } n >> s$$

Assignment 3

$$\mathbb{E}\left[\hat{\mathbf{x}}^T\hat{\mathbf{y}}\right] = \frac{1}{m}\mathbb{E}\left[\mathbf{x}^T S^T S\mathbf{y}\right] = \frac{1}{m}\mathbb{E}\left[\sum_i^d\sum_j^m\sum_k^d \mathbf{x}_k^T S_{kj}^T S_{ji}\mathbf{y}_i\right]$$

We split the sum in two components, with $k \neq i$ and $k = i$:

$$= \frac{1}{m}\mathbb{E}\left[\sum_i^d\sum_j^m\sum_{k\neq i}^d \mathbf{x}_k^T S_{kj}^T S_{ji}\mathbf{y}_i\right] + \frac{1}{m}\mathbb{E}\left[\sum_i^d\sum_j^m \mathbf{x}_i^T S_{ij}^T S_{ji}\mathbf{y}_i\right]$$

Since by hypothesis all S_{ij} are statistically independent, we can define independent random variables z_{ijk}, with $k \neq i$:

$$z_{ijk} = S_{kj}^T S_{ji} = \begin{cases} 1, & \text{prob. } \frac{1}{2} \\ -1, & \text{prob. } \frac{1}{2} \end{cases}$$

Since S only is stochastic, the first component becomes:

$$\frac{1}{m} \sum_{i}^{d} \sum_{k \neq i}^{d} \mathbf{x}_k^T \mathbf{y}_i \sum_{j}^{m} \mathbb{E}\left[S_{kj}^T S_{ji}\right] = \frac{1}{m} \sum_{i}^{d} \sum_{k \neq i}^{d} \mathbf{x}_k^T \mathbf{y}_i \sum_{j}^{m} \mathbb{E}\left[z_{ijk}\right] = 0$$

And the second component, since $S_{ij}^T S_{ji} = 1$ for all i, j, becomes:

$$\frac{1}{m} \sum_{i}^{d} \mathbf{x}_i^T \mathbf{y}_i \sum_{j}^{m} 1 = \mathbf{x}^T \mathbf{y}$$

Hence $\mathbb{E}\left[\hat{\mathbf{x}}^T \hat{\mathbf{y}}\right] = \mathbf{x}^T \mathbf{y}$. $\square$

Assignment 4

Definitions: Let $m = \log_2 n$, n_i the i-th bit of number n, where $i \in \{0, 1, \ldots, m-1\}$, and $\rho(n) = \min\{i \mid n_i = 1\}$. Also assume that we have an ideal hash function whose outputs are uniformly distributed in $[0, m-1]$.

The algorithm is based on m instances of Flajolet–Martin FM_i, one for each bit $i \in \{0, \ldots, m-1\}$, which count how many distinct items have the i-th bit set. By $FM_i[j]$ we denote the j-th bit of the i-th Flajolet-Martin instance.

Algorithm 2: Sum distinct - m Flajolet-Martin

```
Initialize FM_0, ..., FM_{m-1} ← 0;
while n ← stream.next() do
    j ← ρ(hash(n));
    for i ∈ {0, ..., m-1} do
        if n_i = 1 then
            FM_i[j] ← 1;
```

In any point of the stream, we can calculate the requested sum multiplying 2^i by the (estimated) number of items having the i-th bit set, for every $i \in \{0, \ldots, m-1\}$.

Algorithm 3: Sum distinct - Compute result

```
sum ← 0;
for i ∈ {0, ..., m-1} do
    R ← min{j | FM_i[j] = 0};
    sum ← sum + 2^R · 2^i;
```