Foundations of Data Mining - Homework 4

Idan Shabat, 203511597

January 20, 2018

This homework was done with the collaboration of Mor Geva, 200831618.

1 Reverse-reachability with k-mins sketches

(a) We use a dynamic programming algorithm, similar to the one we saw in class. We assume that the purpose of the algorithm is to get a vector of pairs for each node v:

$$\Big((v_1, h_1(v_1)), (v_2, h_2(v_2)), ..., (v_k, h_k(v_k))\Big)$$

where $h_1, ..., h_k$ are the chosen hash functions, and $h_i(v_i) = \min_{u|u\rightsquigarrow v} h_i(u)$.

We initialize each node v with the vector $\big((v, h_1(v)), (v, h_2(v)), ..., (v, h_k(v))\big)$, and then each node sends to all of its out-neighbors k updates: $(v, h_1(v)), (v, h_2(v)), ..., (v, h_k(v))$.

When a node v with a vector $\big((v_1, h_1(v_1)), (v_2, h_2(v_2)), ..., (v_k, h_k(v_k))\big)$ gets an update $(v^*, h_i(v^*))$, it checks if $h_i(v^*) < h_i(v_i)$. If so, it replaces the pair $(v_i, h_i(v_i))$ with the pair $(v^*, h_i(v^*))$ and sends the update $(v^*, h_i(v^*))$ to all of its out-neighbors. Otherwise, it ignores the update.

We may use a queue Q to handle the updates: Insert there the initial kn updates (k for each node), each time dequeue the next update and perform the comparison and if necessary insert updates to Q. The algorithm ends when $Q = \emptyset$.

Correctness: Let v_0 be a node and let $u \in \text{Reach}^{-1}(v_0)$ with minimal $h_i(u)$. Let $u = v_l, v_{l-1}, v_{l-2}, ..., v_0$ be some path from u to v_0 (we know that there is a path). Since u has minimal $h_i(u)$ among all nodes in $\text{Reach}^{-1}(v_0)$, and since $v_l, v_{l-1}, ..., v_0 \in \text{Reach}^{-1}(v_0)$, every v_j that gets an update $(u, h_i(u))$, immediately updates his vector and sends the same update to v_{j-1}. In the initialization, $u = v_l$ sends this update to v_{l-1}, so eventually v_0 will get the update and will change its vector to have $(u, h_i(u))$ in the ith coordinate. Notice that this pair will be changed afterwards iff there is some $u' \in \text{Reach}^{-1}(v_0)$ with $h_i(u') < h_i(u)$. But there is no such u', so v_0 will have the correct value, $(u, h_i(u))$, in the end of the algorithm.

Expected Complexity: Fix an $i \in \{1, ..., k\}$ and a node v. Let $(u_1, h_i(u_1)), (u_2, h_i(u_2)), ..., (u_m, h_i(u_m))$ be all the updates received by v, by the order of their arrival. Notice that v adds the update $(u_j, h_i(u_j))$ to Q iff $h_i(u_j) < h_i(u_1), h_i(u_2), ..., h_i(u_{j-1})$, which happens with probability $\frac{1}{j}$. Therefore, the expected number of updates to the ith coordinate that v adds to Q is:

$$1 + \sum_{j=1}^{m} \frac{1}{j} \leq 1 + \sum_{j=1}^{n} \frac{1}{j} \leq 2 + \ln(n) = O(\ln(n))$$

Summing over all $i \in \{1, ..., k\}$ and all nodes v, we get that the number of updates in Q will be $O(kn \ln(n))$, which is also the running time of the algorithm.

(b) First, given a sketch $\big((v_1, h_1(v_1)), (v_2, h_2(v_2)), ..., (v_k, h_k(v_k))\big)$ for $\text{Reach}^{-1}(v)$ and a sketch $\big((u_1, h_1(u_1)), (u_2, h_2(u_2)), ..., (u_k, h_k(u_k))\big)$ for $\text{Reach}^{-1}(u)$, we build a sketch for $\text{Reach}^{-1}(v) \cup \text{Reach}^{-1}(u)$, by defining:

$$w_i = \begin{cases} v_i & h_i(v_i) < h_i(u_i) \\ u_i & \text{otherwise} \end{cases}$$

and the sketch will be $\big((w_1, h_1(w_1)), (w_2, h_2(w_2)), ..., (w_k, h_k(w_k))\big)$. Note that w_i is the node with the minimum $h_i(w_i)$ among $\text{Reach}^{-1}(v) \cup \text{Reach}^{-1}(u)$. this makes the sketch a k-mins sketch of $\text{Reach}^{-1}(v) \cup \text{Reach}^{-1}(u)$. Therefore, we can apply the estimator from lecture 2, which is a Maximum Likelihood Estimation and was proven to be asymptotically optimal:

$$|\widehat{\text{Reach}^{-1}(v) \cup \text{Reach}^{-1}(u)}| = \frac{k-1}{\sum_{i=1}^{k} h_i(w_i)}$$

2 Seed-set focused centrality

(a) We use the same unbiased estimator $\widehat{I_{v\rightsquigarrow u}}$ we saw in class for the indicator $I_{v\rightsquigarrow u}$ (which equals to 1 iff $v \rightsquigarrow u$):

$$\widehat{I_{v\rightsquigarrow u}} = \begin{cases} \frac{1}{p_{v,u}} & u \in ADS(v) \\ 0 & \text{otherwise} \end{cases}$$

where $p_{v,u}$ is the probability that $u \in ADS(v)$. We set $p_{v,u}$ to be the conditional probability of $u \in ADS(v)$, given that we know $h(w)$ for all $w \in ADS(v)$ which are closer to v than u: $p_{v,u}$ is exactly the kth lowest value in the set

$$\{h(w) \mid w \in ADS(v) \text{ and } d_{v,w} < d_{v,u}\}$$

Obviously, $E[\widehat{I_{v\rightsquigarrow u}}] = p_{v,u} \cdot \frac{1}{p_{v,u}} + 0 = 1$, and therefore the following estimator is unbiased:

$$\widehat{C_{\alpha,S}(v)} = \sum_{u\in S} \widehat{I_{v\rightsquigarrow u}} \alpha(d_{v,u})$$

(b) I. We compute the variance of $\widehat{I_{v\rightsquigarrow u}}$:

$$E[\widehat{I_{v\rightsquigarrow u}}^2] = p_{v,u} \cdot \frac{1}{p_{v,u}^2} + 0 = \frac{1}{p_{v,u}}$$

$$Var(\widehat{I_{v\rightsquigarrow u}}) = E[\widehat{I_{v\rightsquigarrow u}}^2] - E[\widehat{I_{v\rightsquigarrow u}}]^2 = \frac{1}{p_{v,u}} - 1^2 = \frac{1}{p_{v,u}} - 1$$

If the estimators $\widehat{I_{v\rightsquigarrow u}}$ were independent, then by the properties of variance:

$$Var(\widehat{C_{\alpha,S}(v)}) = \sum_{u\in S} Var(\widehat{I_{v\rightsquigarrow u}}\alpha(d_{v,u})) = \sum_{u\in S} \alpha(d_{v,u})^2 Var(\widehat{I_{v\rightsquigarrow u}}) = \sum_{u\in S} \alpha(d_{v,u})^2 (\frac{1}{p_{v,u}} - 1)$$

In particular:

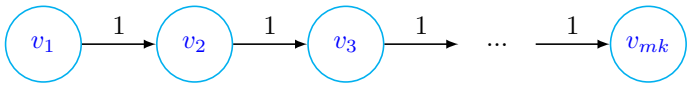
$$Var(\widehat{C_{\alpha,V}(v)}) = \sum_{u\in V} \alpha(d_{v,u})^2 (\frac{1}{p_{v,u}} - 1) \geq \sum_{u\in S} \alpha(d_{v,u})^2 (\frac{1}{p_{v,u}} - 1) = Var(\widehat{C_{\alpha,S}(v)})$$

So $\widehat{C_{\alpha,S}(v)}$ is more concentrated around its expectation than $\widehat{C_{\alpha,V}(v)}$ (that has higher expectation).
For the CV's, we can write for $C_{\alpha,V}(v)$:

$$CV^2 = \frac{\sum_{u\in V} \alpha(d_{v,u})^2 (\frac{1}{p_{v,u}} - 1)}{(\sum_{u\in V} \alpha(d_{v,u}))^2} = \sum_{u\in V} \frac{\alpha(d_{v,u})^2}{(\sum_{u\in V} \alpha(d_{v,u}))^2} (\frac{1}{p_{v,u}} - 1)$$

The largest coefficient is $\frac{\alpha(d_{v,v})^2}{(\sum_{u\in V} \alpha(d_{v,u}))^2} \geq \frac{1}{|V|^2}$. For $C_{\alpha,S}(v)$, the largest coefficient is $\frac{\alpha(d_{v,w})^2}{(\sum_{u\in S} \alpha(d_{v,u}))^2} \geq \frac{\alpha(d_{v,w})^2}{|S|^2 \alpha(d_{v,w})^2} = \frac{1}{|S|^2} \geq \frac{1}{|V|^2}$, where w is the closest to v in S. It means that in the CV of $C_{\alpha,S}(v)$ there is more weight to nodes that are far from v than in the CV of $C_{\alpha,V}(v)$. But that means that the CV of $C_{\alpha,S}(v)$ is higher than the CV of $C_{\alpha,V}(v)$, since far nodes u has larger value of $\frac{1}{p_{v,u}} - 1$ (note that $p_{v,u} = \frac{k}{N}$ where N is the number of nodes w with $d_{v,w} \leq d_{v,u}$, so far nodes has lower $p_{v,u}$).

II. This bound won't hold for a general $S \subseteq V$, as we show in the following example (which demonstrates the argument in (I)):
Consider the following graph



where m is a large integer. Let $S = \{v_{mk}\}$, $v = v_1$ and $\alpha \equiv 1$. Here $C_{\alpha,S}(v) = 1$ and

$$\widehat{Var(C_{\alpha,S}(v))} = \frac{1}{p_{v_1,v_{mk}}} - 1 = \frac{1}{\frac{k}{mk}} - 1 = m - 1 \implies$$
$$CV^2 = \frac{m-1}{1} = m - 1 \implies$$
$$CV = \sqrt{m-1}$$

which can be as large as we want.

(c) As we saw in (b.I), CV^2 can be written as a sum of expressions of the form $\frac{1}{p_{v,u}} - 1$, with coefficients $\frac{\alpha(d_{v,u})^2}{(\sum_{u \in S} \alpha(d_{v,u}))^2}$.

I. We saw that this coefficient is $\geq \frac{1}{|S|^2}$, so when $|S|$ is small, each term of CV^2 has more weight, so the CV is larger.

II. When the nodes in S are far from v, the expressions $\frac{1}{p_{v,u}} - 1$ for $u \in S$, which are larger than for $u \notin S$, get more weight, while the others get weight $= 0$. Therefore again, the CV is larger.

(d) Let k be some positive integer and h be a hash function. We define the sketches $k\text{-}ADS_S(v)$ for every node v in the graph:

$$u \in k\text{-}ADS_S(v) \iff h(u) \text{ is one of the lowest } k \text{ values in } \{h(w) \mid w \in N_v(d_{v,u}) \cap S\}$$

where $N_v(d) = \{w \mid d_{v,w} \leq d\}$. Note that $p_{v,u} = P(u \in k\text{-}ADS_S(v)) = \frac{k}{|N_v(d_{v,u}) \cap S|}$, but we can use conditional probability to set $p_{v,u}$ to be the k-lowest value in

$$\{h(w) \mid w \in N_v(d_{v,u}) \cap S \setminus \{u\}\}$$

For every $u \in S$ we use inverse probability estimator for the indicator $I_{v \rightsquigarrow u}$ (which equals to 1 iff $v \rightsquigarrow u$):

$$\widehat{I_{v \rightsquigarrow u}} = \begin{cases} \frac{1}{p_{v,u}} & u \in k\text{-}ADS_S(v) \\ 0 & \text{otherwise} \end{cases}$$

As usual, $\widehat{I_{v \rightsquigarrow u}}$ is unbiased, and so is:

$$\widehat{C_{\alpha,S}(v)} = \sum_{u \in S} \widehat{I_{v \rightsquigarrow u}} \alpha(d_{v,u})$$

For every $u \neq w \in S$, if $u \in k\text{-}ADS_S(v)$, then the probability that also $w \in k\text{-}ADS_S(v)$ is lower then this probability when we don't know that $u \in k\text{-}ADS_S(v)$, i.e.

$$P(w \in k\text{-}ADS_S(v) \mid u \in k\text{-}ADS_S(v)) \leq p_{v,w} \implies$$
$$\frac{P(u, w \in k\text{-}ADS_S(v))}{p_{v,u} p_{v,w}} \leq 1 \implies$$
$$Cov(I_{v \rightsquigarrow u}, I_{v \rightsquigarrow w}) = \frac{P(u, w \in k\text{-}ADS_S(v))}{p_{v,u} p_{v,w}} - 1 \leq 0$$

So, like in (a):

$$Var(\widehat{C_{\alpha,S}(v)}) \leq \sum_{u\in S} Var(\widehat{I_{v\rightsquigarrow u}}\alpha(d_{v,u})) = \sum_{u\in S} \alpha(d_{v,u})^2 Var(\widehat{I_{v\rightsquigarrow u}}) = \sum_{u\in S} \alpha(d_{v,u})^2(\frac{1}{p_{v,u}} - 1)$$

and:

$$CV^2 \leq \sum_{u\in V} \frac{\alpha(d_{v,u})^2}{(\sum_{u\in S}\alpha(d_{v,u}))^2}(\frac{1}{p_{v,u}} - 1)$$

Let r be the farthest node in S to v, and let q be the closest node in S to v. By the monotonicity of α and of $p_{v,u}$ (both decreasing with the distance):

$$CV^2 \leq |S| \cdot \frac{\alpha(d_{v,q})^2}{(|S|\alpha(d_{v,r}))^2}(\frac{1}{p_{v,r}} - 1) = |S| \cdot \frac{\alpha(d_{v,q})^2}{(|S|\alpha(d_{v,r}))^2}(\frac{|N_v(d_{v,r}) \cap S|}{k} - 1) \leq$$

$$\leq |S| \cdot \frac{\alpha(d_{v,q})^2}{(|S|\alpha(d_{v,r}))^2}\frac{|S|}{k} = \frac{\alpha(d_{v,q})^2}{\alpha(d_{v,r})^2}\frac{1}{k} \implies$$

$$CV \leq \frac{\alpha(d_{v,q})}{\alpha(d_{v,r})}\frac{1}{\sqrt{k}}$$

Knowing α, S in advance, for every $\varepsilon > 0$ we can choose $k = \lceil(\frac{\alpha(d_{v,r})}{\alpha(d_{v,q})})^2\varepsilon^{-2}\rceil = O(\varepsilon^{-2})$, and then we will get $CV \leq \varepsilon$.

For computing the expected size of the sketches, we write: $|k\text{-}ADS_S(v)| = \sum_{u\in S} X_u$, where X_u is an indicator that equals to 1 iff $u \in k\text{-}ADS_S(v)$, i.e. iff $h(u)$ is one of the lowest k values in $\{h(w) \mid w \in N_v(d_{v,u}) \cap S\}$. If $u_1, ..., u_{|S|}$ are the nodes of S ordered by $d_{v,u}$, then:

$$E[|k\text{-}ADS_S(v)|] = E[\sum_{i=1}^{|S|} X_{u_i}] = \sum_{i=1}^{|S|} \min\{\frac{k}{i}, 1\} \leq k\sum_{i=1}^{|S|}\frac{1}{i} = O(k\ln(|S|)) = O(\varepsilon^{-2})$$

for a fixed $|S|$.

For computing the sketches, we perform the same algorithm for the original ADS's (pruned Dijkstra), but we iterate only over the nodes $u \in S$:

For every $u \in S$ with increasing order of $h(u)$, perform Dijkstra's algorithm from u on the reverse graph. When visiting v, if $|\{w \mid w \in k\text{-}ADS_S(v),\ d_{v,w} < d_{v,u}\}| < k$, add u to $k\text{-}ADS_S(v)$. Else, prune the search in v.

Each pass on the edge $v' \to v$ is performed right after $k\text{-}ADS_S(v)$ was updated, which happens at most $|k\text{-}ADS_S(v)| = O(k\ln(S))$ times in expectation. Therefore the expected computation time of all the sketches is $O(mk\ln(|S|))$.

(e) For a given $U \subseteq V$, we first compute $k\text{-}ADS_S(U)$, which is the set of all pairs $(u, d_{U,u})$ s.t. $h(u)$ is one of k lowest values in $\{h(w) \mid w \in N_U(d_{U,u}) \cap S\}$ ($N_U(d_{U,u})$ is the set of all nodes w with $d_{U,w} \leq d_{U,u}$, where $d_{U,w}$ is the minimum distance between a node from U and w):

By the pruned Dijkstra algorithm for computing $k\text{-}ADS_S(v)$, we can assume that the nodes in $k\text{-}ADS_S(v)$ are sorted by increasing $h(u)$. For every $u \in S$, in increasing order of $h(u)$:

- If there is no $v \in U$ s.t. $u \in k\text{-}ADS_S(v)$, continue to the next $u \in S$.
- Else, compute the pair $(u, \min_{v\in U} d_{v,u})$ (the distances $d_{v,u}$ are taken from the sketches $k\text{-}ADS_S(v)$). Only if there are at most $k-1$ couples $(w, d) \in k\text{-}ADS_S(U)$ with $d < \min_{v\in U} d_{v,u}$, insert $(u, \min_{v\in U} d_{v,u})$ into $k\text{-}ADS_S(U)$.

Computation time: $O(|S|(|U| + \log(|k\text{-}ADS_S(U)|))) = O(|S||U| + |S|\log(k)\log\log(|S|))$ (when we use a balanced tree to store $k\text{-}ADS_S(U)$'s elements, sorted by their distances).

Now, notice that

$$INF_{\alpha,S}(U) = \sum_{u\in S} \max_{v\in U} \alpha(d_{v,u}) = \sum_{u\in S} \alpha(\min_{v\in U} d_{v,u}) = \sum_{u\in S} \alpha(d_{U,u}) = C_{\alpha,S}(U)$$

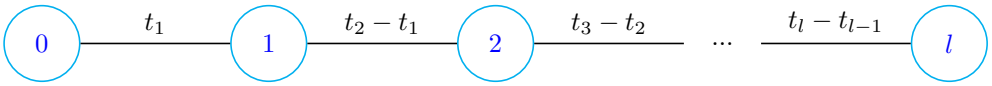
So we can estimate $INF_{\alpha,S}(U)$ as in (d), and with a similar arguments we get an unbiased estimator with $CV \leq O(\frac{1}{\sqrt{k}})$. Again, if we choose $k = O(\varepsilon^{-2})$, we get $CV \leq \varepsilon$.

3 Time-decay

(a) For any current time t we define the weighted undirected graph G_t as follows: $G_t = (V_t, E_t)$, where

$$V_t = \{0\} \cup \{i \mid t_i < t\}$$
$$E_t = \{(i-1, i) \mid t_i < t\}$$

and the distance (weight) of the edge $(i-1, i)$ will be $t_i - t_{i-1}$ (or just t_1 for the edge $(0, 1)$).



For any time t, the centrality of 0 with respect to the decay function α is exactly

$$\sum_{i \in V_t} \alpha(d_{0,i}) = 1 + \sum_{i \mid t_i < t} \alpha(t_i - t_{i-1} + t_{i-1} - t_{i-2} + ... + t_2 - t_1 + t_1) = 1 + \sum_{i \mid t_i < t} \alpha(t_i) = 1 + T_\alpha(t)$$

Given $\varepsilon > 0$, we estimate $T_\alpha(t) = C_{t,\alpha}(0) - 1$ (where $C_{t,\alpha}(0)$ is the centrality of 0 in the graph G_t) in the same way we saw in class:

- Draw a **hash function** $h : \{1, ..., M\} \to [0, 1]$ (assuming M is a large bound on the number of the stream element).
- Maintain **the sketch** $ADS(0)$ while t grows, with parameter $k = \lceil \varepsilon^{-2} \rceil$:
  For the graph G_t, $i \in ADS(0)$ iff $h(i)$ is one of the lowest k values among $\{h(0), h(1), ..., h(i)\}$. Therefore we start with $ADS(0) = \{(0, 0)\}$ and when t_i comes we do:

  $$\text{IF } |ADS(0)| < k \text{ OR } h(i) < k\text{th lowest in } \{h(j) \mid j \in ADS(0)\},$$
  $$ADS(0) \leftarrow ADS(0) \cup \{(i, t_i)\}$$

  For checking the IF statement, we keep a variable s for $|ADS(0)|$, and a balanced binary search tree T for the bottom-k values in $\{h(j) \mid j \in ADS(0)\}$. In the beginning, $s = 0$ and T is empty. When we get a new element with hash value $h(i)$, if $s < k$ we add 1 to s and insert $h(i)$ to T. If $s \geq k$, and $h(i)$ is smaller than the maximum of T, we insert $h(i)$ to T and delete the maximum.
  Sketch (expected) size: $O(k \ln(|V_t|)) = O(\varepsilon^{-2} \ln(|V_t|))$.
  Processing time for each element: $O(\log(k))$.
- **Estimate the indicator** $I_{0 \rightsquigarrow i}$ (equals to 1 iff $0 \rightsquigarrow i$, i.e. always):

  $$\widehat{I_{0 \rightsquigarrow i}} = \begin{cases} \frac{1}{\min\{\frac{k}{i+1}, 1\}} & i \in ADS(0) \\ 0 & \text{otherwise} \end{cases}$$

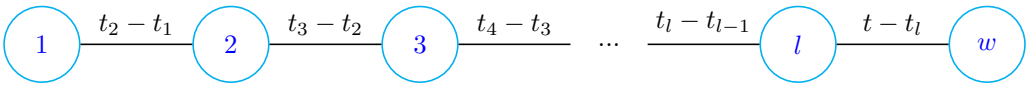
  Note that the probability that $i \in ADS(0)$ is the probability that $h(i)$ is one of the lowest k values among $\{h(0), h(1), ..., h(i)\}$, which is exactly $\min\{\frac{k}{i+1}, 1\}$. So the estimator is unbiased: $E[\widehat{I_{0 \rightsquigarrow i}}] = \min\{\frac{k}{i+1}, 1\} \cdot \frac{1}{\min\{\frac{k}{i+1}, 1\}} + 0 = 1$.
- The **estimator for** $T_\alpha(t) = C_{t,\alpha}(0) - 1$ is then:

  $$\widehat{T_\alpha(t)} = \widehat{C_{t,\alpha}(0)} - 1 = \sum_{i \mid t_i < t} \widehat{I_{0 \rightsquigarrow i}} \alpha(t_i) - 1 = \sum_{i \in ADS(0)} \frac{\alpha(t_i)}{\min\{\frac{k}{i+1}, 1\}} - 1 =$$
  $$= \sum_{i \in ADS(0)} \alpha(t_i) \max\{\frac{i+1}{k}, 1\} - 1 = \sum_{i \in ADS(0) \setminus \{0\}} \alpha(t_i) \max\{\frac{i+1}{k}, 1\}$$
- We know that the **CV** of this estimator is $\leq \frac{1}{\sqrt{2k-2}} \leq \frac{1}{\sqrt{k}} \leq \frac{1}{\sqrt{\varepsilon^{-2}}} = \varepsilon$.

(b) Here we define, for a current time t, a weighted undirected graph $G'_t = (V'_t, E'_t)$ as follows:

$$V'_t = V_t \setminus \{0\} \cup \{w\}$$
$$E'_t = E_t \setminus \{(0,1)\} \cup \{(l,w)\}$$

where w is a special node that represents the current time and l is the maximal i such that $t_i < t$. The distances of the edges of E_t are as in (a), and for (l,w), the distance is $t - t_l$.



Notice that the centrality of w in G'_t with respect to a function α is

$$C_{t,\alpha}(w) = \alpha(d_{w,w}) + \sum_{i \in V_t \setminus \{0\}} \alpha(d_{w,i}) = 1 + \sum_{i \mid t_i < t} \alpha(t - t_l + t_l - t_{l-1} + ... + t_{i+1} - t_i) =$$
$$= 1 + \sum_{i \mid t_i < t} \alpha(t - t_i) = 1 + T_\alpha(t)$$

Again, given $\varepsilon > 0$, we use the same estimation we saw in class for this centrality:

- Draw a **hash function** $h : \{1, ..., M\} \to [0,1]$ (assuming M is a large bound on the number of the stream element).
- Maintain **the sketch** $ADS(w)$ while t grows, with parameter $k = \lceil \varepsilon^{-2} \rceil$:
  We will keep $ADS(w)$ in a balanced binary search tree T that contains the nodes of G'_t, sorted by their hash values $h(i)$. For each element i of T we keep two variables:
  - $d_{w,i}$ = the distance from i to w
  - $rank(i)$ = the number of nodes $j \in \{i, i+1, ..., l, w\}$ s.t. $h(j) \leq h(i)$

  (we can keep these variables in pointers from the tree nodes, for example). Note that $i \in ADS(w)$ iff $rank(i) \leq k$.
  In the beginning, $T = \{w\}$, $d_{w,w} = 0$ and $rank(w) = 1$. When we get a new stream element t_i, we insert i to T (according to its hash value $h(i)$), do $d_{i,w} \leftarrow t - t_i$ and $rank(w) \leftarrow 1$. Then we traverse the tree nodes that has a larger hash value than $h(i)$ and increment their $rank$. If some node rank became $> k$, we delete it from the tree.
  Sketch (expected) size: $O(k \ln(|V_t|)) = O(\varepsilon^{-2} \ln(|V_t|))$.
  Processing time for each element: $O(k \ln(|V_t|)) = O(\varepsilon^{-2} \ln(|V_t|))$ (for updating the $rank$s, in the worst case).
- **Estimate the indicator** $I_{i \rightsquigarrow w}$ (equals to 1 iff $i \rightsquigarrow w$, i.e. always):

$$\widehat{I_{i \rightsquigarrow w}} = \begin{cases} \frac{1}{\min\{\frac{k}{l-i+2}, 1\}} & i \in ADS(w) \\ 0 & \text{otherwise} \end{cases}$$

  Note that the probability that $i \in ADS(w)$ is the probability that $h(i)$ is one of the lowest k values among $\{h(i), h(i+1), ..., h(l), h(w)\}$, which is exactly $\min\{\frac{k}{l-i+2}, 1\}$. So the estimator is unbiased: $E[\widehat{I_{i \rightsquigarrow w}}] = \min\{\frac{k}{l-i+2}, 1\} \cdot \frac{1}{\min\{\frac{k}{l-i+2}, 1\}} + 0 = 1$.
- The **estimator for** $T_\alpha(t) = C_{t,\alpha}(w) - 1$ is then:

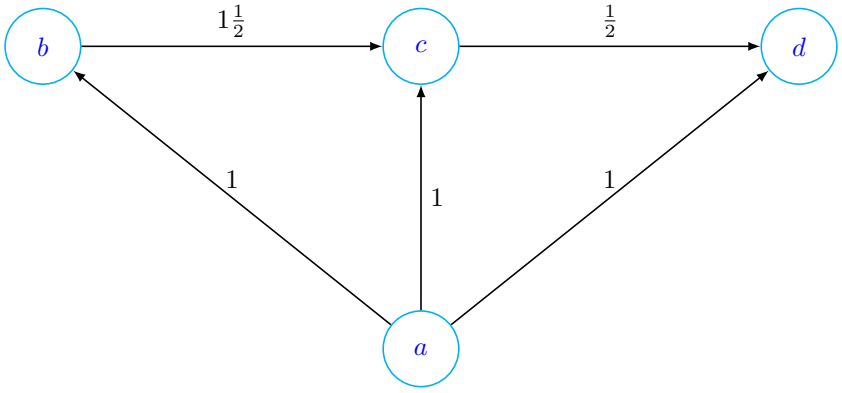
$$\widehat{T_\alpha(t)} = \widehat{C_{t,\alpha}(w)} - 1 = \sum_{i \mid t_i < t} \widehat{I_{i \rightsquigarrow w}} \alpha(t - t_i) - 1 = \sum_{i \in ADS(w)} \frac{\alpha(t - t_i)}{\min\{\frac{k}{l-i+2}, 1\}} - 1 =$$
$$= \sum_{i \in ADS(w)} \alpha(t - t_i) \max\{\frac{l-i+2}{k}, 1\} - 1 = \sum_{i \in ADS(w) \setminus \{w\}} \alpha(t - t_i) \max\{\frac{l-i+2}{k}, 1\}$$

- We know that the **CV** of this estimator is $\leq \frac{1}{\sqrt{2k-2}} \leq \frac{1}{\sqrt{k}} \leq \frac{1}{\sqrt{\varepsilon^{-2}}} = \varepsilon$.

4 Submodularity

(a) **f is not monotone**

Example:



Consider the following subsets of nodes:

$$S_1 = \{a\}, \quad S_2 = \{a, b\}, \quad S_3 = \{a, b, c\}$$

Computing f over the subsets, we get:

$$f(S_1) = 1 + 1 + 1 = 3, \quad f(S_2) = 1 + 1 + 1\frac{1}{2} = 3\frac{1}{2}, \quad f(S_3) = 1 + \frac{1}{2} = 1\frac{1}{2}$$

So we have $S_1 \subseteq S_2 \subseteq S_3$, but $f(S_1) < f(S_2) > f(S_3)$, therefore f is not monotone increasing and not monotone decreasing.

f is submodular

Proof: First notice that if we define $w_{u,v} = 0$ for every $u, v \in V$ such that $(u, v) \notin E$, we can write f as follows:

$$f(S) = \sum_{u \in S,\ v \notin S} w_{u,v}$$

(In the original definition, and in the new one, edges that doesn't exist add 0 to the sum).

Let $S \subseteq T \subseteq V$ be subsets of nodes and let $i \in V \setminus T$ be another node. We show the "diminishing return" property:

$$f(T \cup \{i\}) - f(T) = \sum_{u \in T \cup \{i\},\ v \notin T \cup \{i\}} w_{u,v} - \sum_{u \in T,\ v \notin T} w_{u,v} =$$

$$= \left(\sum_{u \in T,\ v \notin T \cup \{i\}} w_{u,v} + \sum_{v \notin T \cup \{i\}} w_{i,v} \right) - \left(\sum_{u \in T,\ v \notin T \cup \{i\}} w_{u,v} + \sum_{u \in T} w_{u,i} \right) =$$

$$= \sum_{v \notin T \cup \{i\}} w_{i,v} - \sum_{u \in T} w_{u,i}$$

For S we similarly get:

$$f(S \cup \{i\}) - f(S) = \sum_{v \notin S \cup \{i\}} w_{i,v} - \sum_{u \in S} w_{u,i}$$

Since $S \subseteq T$ and $w_{u,v} \geq 0$ for every u, v, we get that $\sum_{u \in S} w_{u,i} \leq \sum_{u \in T} w_{u,i}$ and also $\sum_{v \notin T \cup \{i\}} w_{i,v} \leq \sum_{v \notin S \cup \{i\}} w_{i,v}$, so finally we get:

$$f(T \cup \{i\}) - f(T) = \sum_{v \notin T \cup \{i\}} w_{i,v} - \sum_{u \in T} w_{u,i} \leq \sum_{v \notin S \cup \{i\}} w_{i,v} - \sum_{u \in S} w_{u,i} = f(S \cup \{i\}) - f(S)$$

(b) f **is both monotone and submodular**

Proof: We saw in class that sum of submodular function is submodular, and we can simply prove the same for monotonicity: if $f = \sum f_i$ and f_i are all monotone (increasing) functions, then

$$S \subseteq T \implies f(S) = \sum f_i(S) \leq \sum f_i(T) = f(T)$$

Now, notice that in our case, $f = \sum_{u \in V_2} f_u$, where $f_u(S)$ is the sum of the weights of the heaviest two edges from S to u ($S \subseteq V_1$).

Clearly, f_u is monotone increasing for every u, since if $S \subseteq T$, then the edges from S to u are included in the edges from T to u. Therefore the heaviest 2 edges can only be heavier in T than in S. By the comment above we get that f **is monotone increasing**.

Similarly to (a), we can define $w_{v_1,v_2} = 0$ for every $v_1 \in V_1$ and $v_2 \in V_2$ such that there is no edge (v_1, v_2). For every $U \subseteq V_1$ and $i \in V_1 \setminus U$, let $(a,u),(b,u)$ be the heaviest 2 edges from U to u ($w_{a,u} \geq w_{b,u}$). Then:

$$f_u(U \cup \{i\}) - f_u(U) = \begin{cases} w_{i,u} + w_{a,u} - f_u(U) & i \text{ is one of the heaviest 2 edges from } U \cup \{i\} \text{ to } u \\ w_{a,u} + w_{b,u} - f_u(U) & \text{otherwise} \end{cases} =$$

$$= \begin{cases} w_{i,u} + w_{a,u} - f_u(U) & w_{b,u} \leq w_{i,u} \\ w_{a,u} + w_{b,u} - f_u(U) & \text{otherwise} \end{cases} = \begin{cases} w_{i,u} - w_{b,u} & w_{b,u} \leq w_{i,u} \\ 0 & \text{otherwise} \end{cases} = \max\{0, w_{i,u} - w_{b,u}\}$$

Now let $S \subseteq T \subseteq V_1$ be subsets of nodes and let $i \in V_1 \setminus T$ be another node. We show the "diminishing return" property for f_u. Since T contains S, the second heaviest edge from T to u, (t,u), can only be heavier than the second heaviest edge from S to u, (s,u). Therefore:

$$f_u(T \cup \{i\}) - f_u(T) = \max\{0, w_{i,u} - w_{t,u}\} \leq \max\{0, w_{i,u} - w_{s,u}\} = f_u(S \cup \{i\}) - f_u(S)$$

That is, f_u is submodular for each $u \in V_2$, hence f **is submodular**.

5 Greedy Bicriteria

(a) We repeat a part of the proof we saw in class, for the fact that $f(G_k) \geq (1-\frac{1}{e})f(Opt_k)$:

$$f(Opt_k) - f(G_i) \leq f(Opt_k \cup G_i) - f(G_i) \leq \sum_{u \in Opt_k} f_{G_i}(u) \leq k \cdot \max_u f_{G_i}(u) \implies$$

$$\max_u f_{G_i}(u) \geq \frac{f(Opt_k) - f(G_i)}{k} \implies$$

$$f(Opt_k) - f(G_{i+1}) = f(Opt_k) - (f(G_i) + \max_u f_{G_i}(u)) \leq$$

$$\leq f(Opt_k) - (f(G_i) + \frac{f(Opt_k) - f(G_i)}{k}) = (1 - \frac{1}{k})(f(Opt_k) - f(G_i))$$

By using the last inequality again and again we get:

$$f(Opt_k) - f(G_{i+1}) \leq (1-\frac{1}{k})(f(Opt_k) - f(G_i)) \leq (1-\frac{1}{k})^2(f(Opt_k) - f(G_{i-1})) \leq ...$$

$$... \leq (1-\frac{1}{k})^{i+1}(f(Opt_k) - f(G_0)) = (1-\frac{1}{k})^{i+1}(f(Opt_k) - f(\emptyset)) = (1-\frac{1}{k})^{i+1} f(Opt_k)$$

Therefore:

$$f(G_{i+1}) \geq (1 - (1-\frac{1}{k})^{i+1}) f(Opt_k)$$

Now, notice that this is true for every i, even if $i > k$, so we can substitute $i = ck - 1$ (and use the fact that for every x, $1 - x \leq e^{-x}$):

$$f(G_{ck}) \geq (1 - (1-\frac{1}{k})^{ck}) f(Opt_k) \geq (1 - \frac{1}{e^c})(Opt_k)$$

which is a better approximation, for every integer $c \geq 1$.

(b) In this case, we know that for each i: $f(G_{i+1}) \geq f(G_i) + (1-\varepsilon)\max_u f_{G_i}(u)$. Substituting this in the proof in (a), we get:

$$f(Opt_k) - f(G_{i+1}) \leq f(Opt_k) - (f(G_i) + (1-\varepsilon)\max_u f_{G_i}(u)) \leq$$

$$\leq f(Opt_k) - (f(G_i) + (1-\varepsilon)\frac{f(Opt_k) - f(G_i)}{k}) = (1 - \frac{1-\varepsilon}{k})(f(Opt_k) - f(G_i)) \leq ...$$

$$... \leq (1 - \frac{1-\varepsilon}{k})^{i+1}(f(Opt_k) - f(G_0)) = (1 - \frac{1-\varepsilon}{k})^{i+1} f(Opt_k)$$

For $i = ck - 1$:

$$f(Opt_k) - f(G_{ck}) \leq (1 - \frac{1-\varepsilon}{k})^{ck} f(Opt_k) \leq e^{-c(1-\varepsilon)} f(Opt_k) \implies$$

$$f(G_{ck}) \geq (1 - e^{-c(1-\varepsilon)}) f(Opt_k)$$

We can continue estimating by using the inequality: $e^{\varepsilon} = 1 + \varepsilon + \frac{\varepsilon^2}{2} + \frac{\varepsilon^3}{6} + ... <$
$< 1 + \varepsilon(1 + \frac{1}{2} + \frac{1}{6} + ...) < 1 + e\varepsilon$ $(0 < \varepsilon < 1)$.

$$f(G_{ck}) \geq (1 - e^{-c(1-\varepsilon)}) f(Opt_k) = (1 - e^{-c}e^{\varepsilon}) f(Opt_k) \geq (1 - e^{-c}(1 + e\varepsilon)) f(Opt_k) =$$

$$= (1 - \frac{1}{e^c} - \frac{\varepsilon}{e^{c-1}})) f(Opt_k)$$

For $c > 1$, this is a better approximation than the one we saw in class: $f(G_k) \geq$
$\geq (1 - \frac{1}{e} - \frac{\varepsilon}{2}) f(Opt_k)$