SEEM5020 2023 Fall: Homework 5 Solution

- **Q1.** Event $E_1$: For any $r$-near neighbor $x$, it does not collide with query point $q$. Event $E_2$: There are too many (more than $10L$) far points colliding with $q$ in these $L$ hash functions.

  $P[E_1] = P[g_1(x) \neq g_1(q) \cap g_2(x) \neq g_2(q) \cdots g_L(x) \neq g_L(q)]$

  $=P[g_1(x) \neq g_1(q)]P[g_2(x) \neq g_2(q)] \cdots P[g_L(x) \neq g_L(q)] = (1 - \frac{1}{n^p})^{2n^p} \leq \frac{1}{e^2}$

  $P[E_2] = P[Y \geq 10L] \leq \frac{E[Y]}{10L} \leq \frac{L}{10L} = \frac{1}{10}$

  The LSH query algorithm correctly returns a $c$-approximate $r$-near neighbor with probability at least $\frac{9}{10} - \frac{1}{e^2}$.

- **Q2.**

  **Solution 1.** The total fail probability is at most $\frac{1}{5}$. For each iteration, the fail probability we require is at most $P[E] = \frac{1}{5 \log n}$. For event $E_1$, we have

  $P[E_1] = (1 - \frac{1}{n^p})^L \leq (1 - \frac{1}{n^p})^{n^p(L/n^p)} = \frac{1}{e^{L/n^p}}$

  For event $E_2$, we have

  $P[E_2] = P[Y \geq kL] \leq \frac{E[Y]}{kL} \leq \frac{L}{kL} = \frac{1}{k}$

  Require

  $P[E] = \frac{1}{e^{L/n^p}} + \frac{1}{k} = \frac{1}{5 \log n}$

  Set $L = n^p \ln(10 \log n)$ and $k = 10 \log n$ to achieve the above probability.

  To answer a query for the $c$-approximate nearest neighbor search, it proceeds as follows: Adopt the above setting of the LSH algorithm to find $c$-approximate $r$-near neighbor with an arbitrary $r$. If we can find such a $c$-approximate $r$-near neighbor $x$, we continue to update $r$ as $\text{dist}(q,x)/2$ similar to the binary search and terminate if the current LSH algorithm fails, where we return the last successful point $x'$ for the $c$-approximate nearest neighbor search. In this case, there is no point satisfying $\text{dist}(q,x) \leq \frac{c}{2}\text{dist}(q,x')$ with a high probability, indicating $\text{dist}(q,x') > \frac{c}{2}\text{dist}(q,x')$. Thus $\text{dist}(q,x') < \frac{c}{2}\text{dist}(q,x') < \text{dist}(q,x^*) < c\text{dist}(q,x^*)$ and $x'$ satisfies the definition of the $c$-approximate nearest neighbor search for $c \geq 2$.

  Once the LSH algorithm fails for the initial LSH algorithm, we update $r$ as $2r$ and terminate until we find the first successful point $x'$ satisfying the $c$-approximate $r$-near neighbor search, where we return $x'$ for the $c$-approximate nearest neighbor search. In this case, $\text{dist}(q,x') \leq cr$ but $\text{dist}(q,x^*) > \frac{c}{2}r$, where $\text{dist}(q,x') \leq cr \leq 2\text{dist}(q,x^*)$ satisfying the definition of the $c$-approximate nearest neighbor search for $c \geq 2$.

  **Solution 2.** Apply the binary search on radius $r$ with the $(c,r)$-ANNS query to answer the $c$-ANNS query.
At first, apply \((\sqrt{c}, r)\)-ANNS for \(r\):

\[
r = \left\{ \frac{d_{\min} \cdot \sqrt{c}^{-1}}{2}, \frac{d_{\min} \cdot \sqrt{c}^{1}}{2}, \frac{d_{\min} \cdot \sqrt{c}^{2}}{2}, \ldots \frac{d_{\max}}{\sqrt{c} - 1} \right\}
\]

i.e., \(r = \frac{d_{\min}}{2} \cdot \sqrt{c}^{j}\), where \(j = -1, 0, 1 \ldots \lfloor \log_{\sqrt{c}}\left(\frac{2 \cdot d_{\max}}{(\sqrt{c} - 1) \cdot d_{\min}}\right) \rfloor\).

We use binary search to find the minimum \(r\) that \((\sqrt{c}, r)\)-ANNS returns a point \(p\) but \((\sqrt{c}, \frac{r}{\sqrt{c}})\)-ANNS fails and we have \(d(q, p) \leq \sqrt{c} \cdot r\). Thus, \(\forall p' \in P\), \(d(q, p') > \frac{r}{\sqrt{c}}\) for \((\sqrt{c}, \frac{r}{\sqrt{c}})\)-ANNS fails, which is equivalent to the statement \(r < \sqrt{c} \cdot \min_{p' \in P} d(q, p')\). Hence, we have

\[
d(q, p) \leq \sqrt{c} \cdot r < c \cdot \min_{p' \in P} d(q, p')
\]

which means point \(p\) corresponds to the answer of \(c\)-ANNS query.

Next consider the two cases where \(r = \frac{d_{\min} \cdot \sqrt{c}^{-1}}{2}\) or \(r = \frac{d_{\max}}{\sqrt{c} - 1}\).

When \(r = \frac{d_{\min} \cdot \sqrt{c}^{-1}}{2}\), if \((\sqrt{c}, \frac{d_{\min} \cdot \sqrt{c}^{-1}}{2})\)-ANNS returns a point \(p\) which has \(d(q, p) \leq \frac{d_{\min}}{2}\), then there is no other point \(p' \in P\) that \(d(q, p') < \frac{d_{\min}}{2}\), otherwise, by triangle inequality, we will have \(d(p, p') \leq d(q, p) + d(q, p') < d_{\min}\), which contradicts the fact. Hence, the point \(p\) returned is the nearest neighbor of \(q\).

When \(r = \frac{d_{\max}}{\sqrt{c} - 1}\), if \((\sqrt{c}, \frac{d_{\max}}{\sqrt{c} - 1})\)-ANNS fails to return a point, which means \(D = \min_{x \in P} d(q, x) = d(q, p') > \frac{d_{\max}}{\sqrt{c} - 1}\). By triangle inequality, we have:

\[
\forall p \in P, d(q, p) \leq d(p, p') + d(q, p') \leq d_{\max} + D < (\sqrt{c} - 1) \cdot D + D < \sqrt{c} \cdot D
\]

Hence, any point \(p \in P\) is the answer of \(\sqrt{c}\)-ANNS query, and also the answer of \(c\)-ANNS query.

Now we need \(k = \log_{\sqrt{c}}\left(\lceil \log_{\sqrt{c}}\left(\frac{2 \cdot d_{\max}}{(\sqrt{c} - 1) \cdot d_{\min}}\right) \rceil \right)\) \((c, r)\)-ANNS queries in total to obtain the answer of \(c\)-ANNS query. The failure probability of \(c\)-ANNS query is at most \(\frac{1}{5}\), so the failure probability of each \((c, r)\)-ANNS query should be at most \(\frac{1}{5k}\). According to Q1, set \(L = k_{1} \cdot n^{\rho}\) and the stopping condition to retrieve at most \(k_{2} \cdot L\) data points to make \(\frac{1}{k_{1}^{2}} + \frac{1}{k_{2}} < \frac{1}{5\kappa}\). Now, we can achieve a \(c\)-ANNS query.

**Q3.**

**Solution 1.** Increase the number of buckets of the same size from 2 to \(b > 2\). This comes at the increased space.

**Solution 2.** If we want to keep the same sketch size, we may consider the following batch update-based solution. If \(X\) is come in a short time with high frequency, according to the original operation, it will take \(O(X \cdot \log N)\) time complexity to update, degrading the throughput.

Actually, we can handle \(X\) update operations together in \(O(X + \log(X + N))\) time.
Define \( b_i \) where \( b_i = \{1, 2\} \) as the number of buckets with the size of \( 2^i \) before the update. Let \( T \) denote the number of 1s in all buckets, i.e.,
\[
T = \sum 2^i \cdot b_i.
\]
Our goal is to obtain \( b'_i \) (\( b'_i = \{1, 2\} \)) to denote the number of buckets with size of \( 2^i \) after update, then we have:
\[
T + X = \sum 2^i \cdot b'_i.
\]
It is not difficult to obtain \( b'_i \). We can first determine the maximum bucket size as \( 2^k \). Then we have
\[
T + X = \sum_{i=0}^{k} 2^i \cdot b'_i
\]
\[
T + X - (2^{k+1} - 1) = \sum_{i=0}^{k} 2^i \cdot (b'_i - 1)
\]
We can see that \( \{b'_i - 1\} \) is binary representation of \( T + X - (2^{k+1} - 1) \).
So, we know all bucket sizes after the update and only need to attach the correct timestamps to each new bucket. We achieve this by scanning the newly coming 1s and all the old buckets before updating from the latest timestamp to the oldest timestamp.
Before scanning, set \( y = 0 \) and \( t = 0 \). During the scanning process, we add 1 to \( y \) if we go through a 1 or add \( 2^i \) to \( y \) when we go through an old bucket with the size of \( 2^i \). Update \( t \) to \( \max(t, t_c) \) where \( t_c \) is the timestamp of the 1 or the old bucket.
Once \( y \) equals the size of the current new bucket, we attach the timestamp \( t \) to the corresponding bucket, let \( y = 0 \) and \( t = 0 \), and focus on the next new bucket.
Thus, we update \( X \) 1s only in time complexity \( O(X + \log(X + N)) \), which is much faster than \( O(X \cdot \log N) \). In this approach, the sketch size remains unchanged.