1. **Open Addressing**  
Let table size be $m = 15$ (with items indexed from 0...14).  
Use the hash function $h(x) = (x \mod 15)$ and linear hashing to hash the items 19, 6, 18, 34, 25, 34 in that order.  
Draw the resulting table.  

*Solution: See external PDF*

2. **Universal Hashing**  
Recall the universal hash function family defined by  
\[ h_{a,b}(x) = \left( (ax + b) \mod p \right) \mod m \]

where $a \in \mathbb{Z}_p^*$, $b \in \mathbb{Z}_p$ and $p$ is a prime with $p \geq U$. Let $p = 17$, $m = 5$. For all $x = 0, 1, \ldots, 16$ write the values for $h_{1,0}(x)$. Now write all the values for $h_{2,2}(x)$.  

*Solution:*

<table>
<thead>
<tr>
<th>$x$</th>
<th>$h_{1,0}(x)$</th>
<th>$2x + 2 \mod 17$</th>
<th>$h_{2,2}(x)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>12</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>16</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>9</td>
<td>4</td>
</tr>
<tr>
<td>13</td>
<td>3</td>
<td>11</td>
<td>1</td>
</tr>
<tr>
<td>14</td>
<td>4</td>
<td>13</td>
<td>3</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
3. Divide and Conquer for closest pair

Let \( P = \{p_1, p_2, \ldots, p_n\} \) be \( n \) two-dimensional points and define

\[
\delta(P) = \min_{p, p' \in P : p \neq p'} d(p, p')
\]

to be the closest pair distance of \( P \).

Let \( X \) be a real value and split \( P \) on the line \( x = X \) so that

\[
P_L = \{p \in P : p.x \leq X\}, \quad P_R = \{p \in P : p.x > X\}.
\]

Suppose you are given the closest pair distance of the two sets:

\[
\delta_L = \delta(P_L) \quad \text{and} \quad \delta_R = \delta(P_R).
\]

Set \( \delta' = \min(\delta_L, \delta_R) \) and define the points contained by the \( \delta' \) strips to the left and right of the line \( x = X \) by

\[
S_L = \{p \in P_L : X - p.x \leq \delta'\}, \quad S_R = \{p \in P_R : p.x - X \leq \delta'\}
\]

(a) Prove that

\[
\delta(P) = \min(\delta_L, \delta_R, d(S_L, S_R))
\]

where \( d(P_1, P_2) = \min\{d(p_i, p_2) : p_1 \in P_1, p_2 \in P_2\} \).

**Solution:** By definition

\[
\delta(P) = \min(\delta_L, \delta_R, d(P_L, P_R)).
\]

If \( \delta < \delta' \), then \( \delta = d(P_L, P_R) \), i.e., \( \delta = d(p, p') \) where \( p \in P_L \) and \( p' \in P_R \). But, if \( p \notin S_L \) or \( p' \notin S_R \) then

\[
\delta = d(p, p') \geq |p'.x - p.x| \geq \delta'
\]

leading to a contradiction.

(b) Suppose that you are given the values \( \delta_L \) and \( \delta_R \) and each of the sets \( P_L \) and \( P_R \) sorted by \( y \)-coordinate. Show how to calculate \( \delta(P) = \min(\delta_L, \delta_R, d(S_L, S_R)) \) in \( O(n) \) time.

**Hint.** In \( O(n) \) time first find \( S_L \) and \( P_L \), each sorted by \( y \) coordinate. Then show how, in \( O(|S_L| + |S_R|) \) time, you can find \( d(S_L, S_R) \) by using the ideas from the gridding lemma.

**Solution:**

In \( O(n) \) time walk through each of \( P_L \) and \( P_R \), pulling out the items in \( S_L \) and \( S_R \) sorted in \( y \) increasing order. Then, in another \( O(n) \) step, merge the two lists so that you have \( S_L \cup S_R \) in increasing \( y \) order. Put these values in an array sorted by increasing \( y \) order so that you can access an item’s predecessor and successor in \( O(1) \) time.

Using a variant of the gridding lemma taught in class we can now see that if \( p \in S_L, p' \in S_R \) and \( d(p, p') < \delta \) then \( p' \) must be at most 11 points above \( p \) or 11 points below \( p \) in the sorted list. This immediately gives the algorithm: Walk through the sorted list from smallest to largest \( y \) coordinate. If current point \( p \) is in \( S_L \), find the 11
points above it and the 11 points below it. This can be done in \(O(1)\) time. Throw away the points that are in \(S_L\), leaving only the points in \(S_R\). Calculate the distance between \(p\) and all of these \(O(1)\) points and keep the minimum value. After doing this for all the points in \(S_L\), return the smallest distance found. This will be \(d(S_L, S_R)\) if \(d(S_L, S_R) \leq \delta'\).

(c) Now construct a divide and conquer algorithm for finding \(\delta(P)\) that works by

(i) Finding the median by \(x\)-coordinate of \(P\). Set this \(x\) coordinate to be \(X\).
(ii) Split \(P\) on \(X\) into \(P_L\) and \(P_R\).
(iii) Recursively find \(\delta(P_L)\) and \(\delta(P_R)\)
(iv) Use the ideas above to find \(\delta(P)\) using \(O(n \log n)\) extra time

Note that the recursion will terminate when \(P = \{p\}\) or \(P = \{p, p'\}\). In those cases \(\delta(P) = \infty\) or \(\delta(P) = d(p, p')\) can be found in \(O(1)\) time.

The correctness of the algorithm follows from (a) and (b).

Show how to implement the algorithm in \(O(n \log^2 n)\) time.  

Solution:

(i) and (ii) take \(O(n)\) time. (iii) requires \(2T(n/2)\).
(iv) requires sorting \(P_L\) and \(P_R\) and then performing the \(O(n)\) algorithm from the previous part.

Sorting \(P_L\) and \(P_R\) requires \(O(n \log n)\) time so the running time recurrence is

\[
T(n) \leq 2T(n/2) + O(n \log n) + O(n) = 2T(n/2) + On(\log n)
\]

which gives \(T(n) = O(n \log^2 n)\).

(d) Can you improve this to \(O(n \log n)\) time?

Solution:

Let \(CP(P)\) be the result of the algorithm run on point set \(P\). We modify the algorithm so that, instead of just returning \(CP(P)\), it also returns \(P\) sorted by \(y\) coordinate; That is, after finding the closest pair distance \(\delta\) in (iv), it then uses \(O(n)\) time to merge \(P_L\) and \(P_R\) (which had been recursively returned in sorted order) so that \(P\) is now sorted (by \(y\) coordinate) as well. The algorithm works by

(i) Finding the median by \(x\)-coordinate of \(P\). Set this \(x\) coordinate to be \(X\).
(ii) Split \(P\) on \(X\) into \(P_L\) and \(P_R\).
(iii) Recursively find \(\delta(P_L)\) and \(\delta(P_R)\).
Recursion also returns \(P_L\) and \(P_R\) is sorted \(y\)-order
(iv) Use the ideas above to find \(\delta(P)\) using \(O(n)\) extra time
(v) Merge \(P_L\) and \(P_R\) to get \(P\) in sorted \(y\) order.

the running time recurrence is now

\[
T(n) \leq 2T(n/2) + O(n) = 2T(n/2) + On(\log n) = O(n \log n).
\]