New CS 473 Spring 2015
Homework 4 Problem 1

Arash Khatibi (khatibi2)
Liang Tao (ltao3)
Chen Zhang (czhang49)

Solution (Problem 1): Define the following variables and values.
$S[1,2,\ldots,n]$:the unsorted input value list.
$B[k]$: the $k-th$ box in the histogram.
$\widetilde{S}[0,1,2,\ldots,n]$: the sorted input value list with $\widetilde{S}[0]=0$
$Area(i,m)$: $(\widetilde{S}[m]-\widetilde{S}[i])*(m-i+1)$ the area of a box, spanning from input value i to input value m, in which $2\le i\le n, i\le m\le n$
$BestHisto(i,j)$: The best histogram starting at $\widetilde{S}[i]$ with j box left, $j=k,\ldots,1$.

The first thing we do after given the unsorted input value is to sort the list, which will take $O(n\log n)$ time. It will give us the sorted list $\widetilde{S}[1,2,\ldots,n]$. Then start the algorithm based on the sorted list $\widetilde{S}$.

$$BestHisto(i,j)=\begin{cases}((n-i+1)*(1-\widetilde{S}[i]))^2 & j=1\\ \min_{i\le m\le n}((m*\widetilde{S}[m])^2+BestHisto(m,j-1)) & j=k\\ \infty & i<n, j=0 \quad or \quad n=i, j>1\\ \min_{i\le m\le n}(Area(i,m)^2+BestHisto(m,j-1)) & else\end{cases}$$

The first case means that if there is only one box left, so the best histogram with only one box left is simply the box from the starting $\widetilde{S}[i]$ to 1, including all the input values in between .
The second case means that if we are putting down the first box, we calculate the area of the box by the number of values inside the box and the range of the box, since the starting point is always 0.
The third case means that if we are running out of boxes and there are still input values left, then there is no way we could move on, thus this case is marked by assigning the bandwidth as infinity. Similarly, if all the input values are assigned to a box and there are still boxes left, we can neither move forward from here
The fourth case is that for a certain box that needs to be determined, it starts from the previous end of a box and visit all the rest of the sorted input array, since the box always starts and stops at a input array value. Then it examines what the best histogram is if the box ends at that location.

A 2D array can be used for memoization of this problem. Define the 2D array in the following way:

$$LP(i,j)=BestHisto(i,j)$$

in which $1\le i\le n$ and $1\le j\le k$. Since we can see that the LP matrix is filled from $j=1$ column to $j=k$ column, so the memoization process also follows this pattern. Thus the algorithm goes like below:

Initialize the matrix LP.
fix $j=1$, for $i=1:n$
  calculate $BestHisto(i,j)$ based on $j=1$
for $j=2:k-1$, $i=1:n$
  calculate $BestHisto(i,j)=\min_{i\le m\le n}(Area(i,m)^2+BestHisto(m,j-1))$
fix $i=1, j=k$
  calculate $BestHisto(1,k)=\min_{i\le m\le n}((m*\widetilde{S}[m])^2+BestHisto(m,j-1))$

Note for the last case we only need to fix $i = 1, j = k$ since only this value is needed. With this algorithm, since for each entry in $BestHisto(i, j)$, we need to compare the previous column, the algorithm goes $O(n^2 \times k)$. Plus at the very beginning the sorting algorithm would take $O(n \log n)$ time, the overall running time is $O(n^2 \times k + n \log n)$, which makes it $O(n^2 \times k)$.

■