CSC 212—Algorithms and Complexity
Implement divide-and-conquer algorithms for an 8-way mergesort and closest pairs of points in 1-D and 2-D.
Practical 5 Term 3 22 August 2016

Submit today or before 17h20 Friday 26 August 2016.

Do all your work inside your new `53practical` directory. Please do not store any data in your working directory. Use the data provide in the `ds/notes` directory. Marks will be deducted if there is any data in your answer files.

1. *Put this work in the subdirectory* `1answer`. Using a divide-and-conquer algorithm, implement a merge sort that splits the data up into eight (8) fairly equal parts at each recursive step and then combine the eight sorted lists by merging them. There is a list of 8^6 integers called `shuffledints.text` in the `ds/notes` directory. Your program must print the smallest and largest elements only.

2. *Put this work in the subdirectory* `2answer`. Using a divide-and-conquer algorithm, implement 1-D closest pairs.
   1-D closest pairs—Identify, and determine the distance between the closest pair of distinct points lying on the X-axis using a divide-and-conquer method. Note first sorting with merge sort and then determining the closest pair is not a pure divide-and-conquer method for doing this—it is a mixed method. There is a list of doubles in the file `Xlinepoints.text` in the `ds/notes` directory. Your program must print the values of the closest pair of points.

3. *Put this work in the subdirectory* `3answer`. Using a divide-and-conquer algorithm, implement 2-D closest pairs.
   2-D closest pairs—Determine the distance between the closest pair of distinct points lying in a 2-D plane using a divide-and-conquer method. We have provided a list of `double` point pairs in the XY-plane in the `XYplanepoints.text` directory. Your program must print the values of the closest pair of points.

Use the Internet if the description of an algorithm is not clear enough.

Eight-way merge sort An array `int X[]` containing `N` elements can be sorted using the void method `void mergesort(X, from, to)`, which works as follows.

MERGESORT

1. For only one element the method does nothing.
2. For less than eight elements, arrange them in the correct order.
3. For more that eight elements split the list into eight parts and sort each part `mergesort(X, from1, m1)`, `mergesort(X, from2, m2)`,..., `mergesort(X, from8, m8)`. These eight sorted parts are then merged together yielding a single sorted subarray `X[from..to]`. Note that `from1 = from`, `from2 = m1 + 1`, etc.

1-D closest pairs Identify and return the distance of the closest pair of points in the real interval $[left, right]$.

CLOSESTPAIR1D

1. For only one point the method does nothing.
2. For exactly two points the method returns their distance and indices.
3. For more that two points calculate `m = (from + to)/2` and determine the closest pair in the first half of `X` with `leftD = closestpair1D(X, from, m)` and its second half with `rightD = closestpair1D(X, m+1, to)`. The closest pair must then be determined in the straddle area as follows. The smallest of `leftD` and `rightD` is called δ. The straddle area is now the interval that starts at $m - \delta$ and ends at $m + \delta$. The closest pair must be found that lies in this interval.
4. The closest pair is `min(leftD, rightD, delta)`.

2-D closest pairs Identify and return the distance of the closest pair of points in the real interval $[left, right]$.

CLOSESTPAIR2D

1. For only one point the method does nothing.
2. For exactly two points the method returns their distance and indices.
3. For more that two points calculate `m = (from + to)/2` and determine the closest pair in the first half of `X` with `leftD = closestpair2D(X, from, m)` and its second half with `rightD = closestpair2D(X, m+1, to)`. The closest pair must then be determined in the straddle area as follows. The smallest of `leftD` and `rightD` is called δ. The straddle area is now the the area bounded by the interval that starts at $m - \delta$ and ends at $m+\delta$ and runs along the necessary part of the Y-axis. The closest pair must be found that lies in this rectangle. The straddle area now has an added dimension but is otherwise treated similarly to that in the 1-D case.
4. The closest pair is `min(leftD, rightD, delta)`.