# AoA Programming Assignment 3

Author: Gabriel Hofer

Course: CSC-372

Instructor: Dr. Rebenitsch

Due: November 5, 2020

Computer Science and Engineering
South Dakota School of Mines and Technology

# Question 1

**1. [12pt] Describe, in pseudocode, how to make an algorithm using closest pair of points algorithm to make a dendogram**

The recursive calls of the closest points function form a binary tree because the set of points is divided in half and two recursive calls are made: one for the right half and one for the left half. After those calls have returned, we know the closest pair of points in each half of the set. So, in order to make a dendogram, we simply need to show the structure of the recursive function calls by creating a single node in each function call. Then we add edges connecting each parent call to each of their two children (left and right) recursive calls.

```
CLOSEST-PAIR-OF-POINTS:
1. Sort the points from smallest to largest x coordinate (X)
2. Sort the points from smallest to largest y coordinate (Y)
3. Evenly divide the set of points into "left" and "right" sets (both X and Y!)
4. Repeat division recursively until there are only 2 points
5. To merge, the closest pair of points is either the closest pair
   returned from both sides or is across the split boundary as follows:
   a. find all the Y points within d of the split line where d is
      the distance between the closest points from both halves.
   b. check the nearby points that could be closer (there are only 7 max)
   c. update the minimum pair
// Dendogram Part
6. Create a Node
   a. Add an edge between this Node and the node returned from
      the left-half recursive call.
   b. Add an edge between this Node and the node returned from
      the right-half recursive call.
```

**2. [8pt] What is the run time of your algorithm? Prove, mathematically, what would the run time be for your algorithm?**

The number of nodes in the tree is proportional to the number of function calls or nodes created in the dendogram. If there are $N$ points, then there are $Nlog(N)$ function calls. The runtime of this algorithm is the same as building a binary tree. Creating each node and edge in the dendogram costs $O(1)$. So the runtime of our algorithm is the following:

$$T(N) = O(N \cdot log(N))$$

# Question 2

**a. [5 points] What is the optimal substructure (you must name the specific item being removed)?**

The subproblems in this problem occur when we add some subset of rods to the beam from the total set of rods that we have available. Also, in our algorithm, we always add the smallest unwelded rod to the beam. Thus, $SomeRods$ will contain the first $SomeRods.size$ smallest rods from the set of all rods $AllRods$.

**b. [6 points] What is the greedy property?**

Greedy Choice Property: A global optimum can be attained by selecting a local optimum. For this problem, every time we add a rod to the beam we maintain that the current solution is optimal for the subset of rods that have already been added to the beam AND this locally optimal solution will be used to build more globally optimal solutions. Thus, always adding the smallest rod to the current layer is both locally optimal and globally optimal.

**c. [9 points] Prove correctness of your algorithm. A proof of contradiction is acceptable.**

```
MAKE-BEAM:
  layer:=1
  while remainingRods.size() > 0  do
    beam[ layer ].add(min(remainingRods))    // add shortest rod to beam
    remainingRods.erase(min(remainingRods)) // remove rod from list
    rodCnt[ layer ] += 1                     // count rods in this layer
    if remainingRods.size() <= rodCnt[ layer ] then
      beam[ layer ].add(remainingRods)       // add all rods to the beam
      rodCnt[ layer ] += remainingRods.size()
      remainingRods.clear()                  // remove all rods
      break while loop
    if rodCnt[ layer ] > rodCnt[ layer - 1 ]
      inc layer                              // goto next layer
```

**Proof:** The loop invariants are the two criteria stated in the problem statement.

1. The totals summed length of rods in the $i^{th}$ layer must be less than the total summed length of rods in the $(i+1)^{th}$ layer.
2. The total number of rods in the $i^{th}$ layer must be less than the total number of rods in the $(i+1)^{th}$ layer.

**Initialization:** Before adding any rods, there is only one layer - the criteria require two layers.
**Maintenance:** $L$ is the current layer. $rodCnt[L]$ is the number of rods in layer $L$. We know that layer $L+1$ won't be started if it can't be completed. In other words, layer $L+1$ won't be started unless we know that there are at least $L+1$ remaining/unwelded rods. Also, rods are added to layer $L$ while $rodCnt[L] \leq rodCnt[L-1]$ which maintaines the second criterion. Given that the second criterion is maintained, the first criterion is also met since we also always add the shortest remaining rod to the beam:

$$avgRodLength[L-1] * rodCnt[L-1] < avgRodLength[L] * rodCnt[L]$$

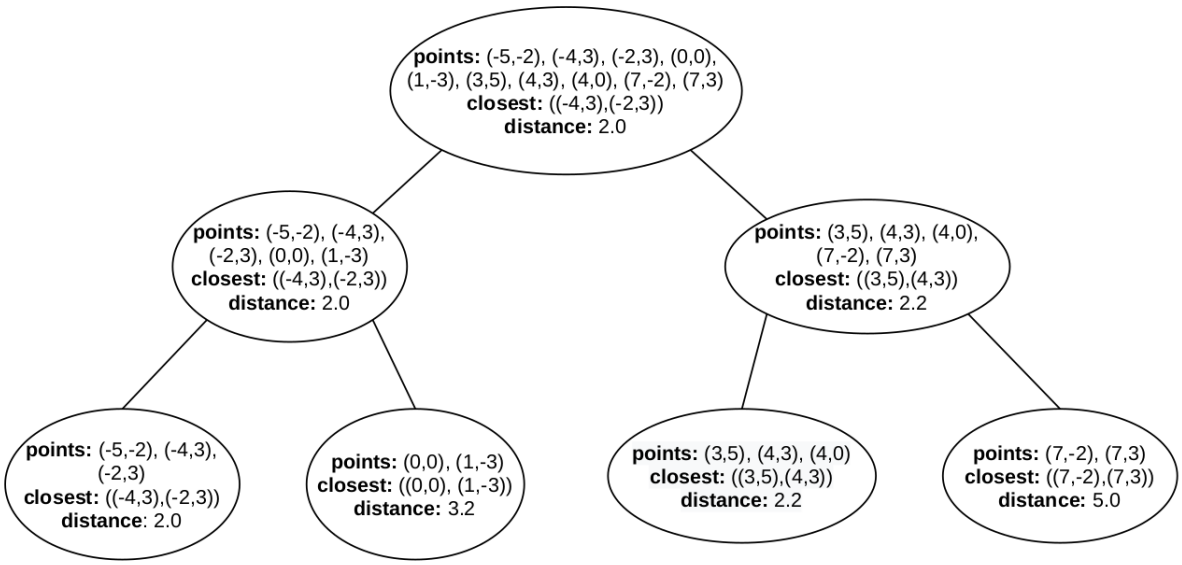
It follows that,

$$summedRods[L-1] < summedRods[L]$$

**Termination:** Also, each layer $L+1$ contains more rods than layer $L$. And, since all rods are welded to the beam in non-decreasing order, the summed length of rods in layer $L+1$ is greater than the summed length of rods in layer $L$.

# Question 3

**"points"** denote all of the points in the set. Each set is essentially divided in half. If the size of the set has an odd parity, then the middle element is put in the left-hand set. **"closest"** represents the closest pair of points in the set of **points**. It is the pair of points that is "returned".




# Question 4

| Step | Add/Remove | Segments | Lines Compared | Intersection? |
|---|---|---|---|---|
| 1 | Add A | A | NA | N |
| 2 | Add B | B, A | A&B | N |
| 3 | Add C | B, A, C | A&B, A&C, B&C | N |
| 4 | Add D | B, D, A, C | A&B, A&C, A&D, B&C, B&D, C&D | N |
| 5 | Remove A | B, D, C | B&C, B&D, C&D | N |
| 6 | Remove B | D, C | C&D | N |
| 7 | Add E | D, C, E | C&D, C&E, D&E | N |
| 8 | Remove C | D, E | D&E | N |
| 9 | Remove D | E | NA | N |
| 10 | Add G | E, G | E&G | N |
| 11 | Add F | E, G, F | E&F, E&G, F&G | N |
| 12 | Add H | E, G, F, H | E&G, E&F, E&H, G&F, G&H, F&H | Y |