Code Jam 2012 - Round 2

Analysis: Aerobics

The key to this problem lies in realizing that there really is a lot of space on the mat to take advantage of, and a number of different approaches will work.

For the small input, putting the circles along one of the longer edge, and - if it runs out - along the opposite edge will work. The precise analysis of why they fit is somewhat tedious, so we will skip it in favor of describing two solutions that can also deal with the large input.

A randomized solution

The large input is a more interesting case. We will describe two solutions that allow one to solve this problem. The first one will be randomized. (If you solved the Equal Sums problem in Round 1B, you should already realize how helpful randomized algorithms can be!).

We will order the circles by decreasing radius. We will then take the circles one by one, and for each circle try to place it on the mat at a random point (that is, put the center of the circle at a random point on the mat). We then check whether it collides with any circle we have placed previously (by directly checking all of them). If it does collide, we try a different random point and repeat until we find one that works. When we succeed in placing all the circles, we're done.

Of course, if we manage to place all the circles, we have found a correct solution. The tricky part is why we will always find a good place to put the new circle in a reasonable time. To see this, let's consider the set of "bad points" - the points where we cannot put the center of a new circle because it would cause a collision. If we are now placing circle **j** with radius $\mathbf{r_j}$, then it will collide with a previously placed circle **i** if and only if the distance from the new center to the center of circle **i** is less than $\mathbf{r_i} + \mathbf{r_j}$. This means that the "bad points" are simply a set of circles with radii $\mathbf{r_i} + \mathbf{r_j}$.

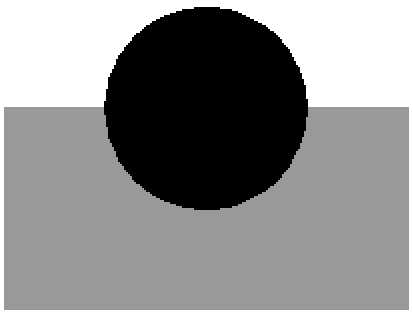
What is the total area of the set of bad points? Well, since the set is a group of circles, the area is at most the sum of the areas of the circles. (It can be less because the circles can overlap, but it cannot be more). As we are placing the circles ordered by decreasing radius, we know $\mathbf{r_j} \leq \mathbf{r_i}$, so the area of the $\mathbf{i}^{th}$ "bad" circle is at most $\pi * (2\mathbf{r_i})^2 = 4\pi \mathbf{r_i}^2$. Here is where we will use that the mat is so large - we see that the total bad area is always at most 80 percent of the mat. Therefore, we have at least a 1 in 5 chance of choosing a good center on every attempt. In particular, it is always possible to find a good center, and it will take us only a few tries.

For each attempt, we have to make O(**N**) simple checks, and we expect to make at most 5**N** attempts, so the expected time complexity of this algorithm is $O(\mathbf{N}^2)$ - easily fast enough.

A deterministic solution

As usual, if we are willing to deal with a bit more complexity in the code, we can get rid of randomness in the solution, taking advantage of all the extra space we have in a different way. One sample way to do this follows.

To each circle of radius **R** we attach a 4**R** * 2**R** rectangle as illustrated below:



The top edge of the rectangle passes through the center of the circle. The bottom edge, the left edge, and the right edge are all at a distance of 2**R** from the center of the circle.

We now place circles one at a time, starting from the ones with the larger radius. We always place each circle in the topmost (and if we have a choice, leftmost) point not covered by any of the rectangles we have already drawn.

An argument similar to the one used in the randomized solution proves that if we place a circle like that, it does not collide with any of the previously placed circles. As we place each point in the topmost available point, the centers of the previously placed circles are necessarily above the last one placed, and so if our new circle would collide with one of the previous ones, it would have to be in the rectangle we associated with it.

Now notice that the areas of all the rectangles we place is the sum of $8\mathbf{R}^2 = 8 / \pi$ times the total area of the circles - which is easily less than the area of the mat. This means we always can place the next circle within the mat.

This solution is somewhat more tricky to code than the previous one (because we have to find the topmost free point, instead of just choosing a random one), but still easier than if we tried to actually stack circles (instead of replacing them by rectangles).