# Homework 2

Mihai Codoban  Arpit Christi  Caius Brindescu
Morgan Shirley

October 25, 2014

Colleagues we collaborated with:

- a
- b

**Problem 1**

**Problem 2** Given a set $C$ of $n$ circles in the plane, each specified by its radius $r$ and the coordinates of its center $(x, y)$, we set out to find a minimum set of rays from the origin $(0, 0)$ that will intersect every circle.

(a) We define a circle's *angular range* to be the angles at which a ray fired from the origin will intersect that circle. We define a circle's *highest angle* as the highest angle in its angular range (with respect to a ray that we define as having angle 0, with angles increasing in a counterclockwise direction).

Given the condition that there exists a ray from the origin which does not intersect any circles in $C$, we can define the following algorithm.

**Algorithm 1**

```
Set a ray from (0, 0) which does not intersect any circles in C to be angle 0.
L ← C sorted by highest angle
m ← 0
while L is not empty do
    Create ray R_m at highest angle of first circle in L
    Delete from L all circles intersected by R_m
    m ← m + 1
end while
return m
```

The fact that this algorithm defines a set of rays that intersect all circles in $C$ is clear, as we only remove circles that intersect with a ray and we remove all circles.

To show that this is an optimal solution, assume that there is an optimal solution that does not fire a ray at the highest angle of the first circle in our sorted list $L$. Obviously, the first ray in the solution will have to be in the first circle's angular range, because otherwise that circle will not be intersected by any ray. The correctness of the solution will not be changed by moving the first ray to the highest angle of the first circle. This is because such an alteration will not move the ray out of the angular range of any circle, as the first circle has the lowest highest angle of any circle in $L$.

After the removal of any circles intersected by the first ray from $L$, the situation is remains the same - the optimality of a solution would not be changed by moving the next ray to the highest angle of the first circle in $L$. Therefore, by induction, Algorithm 1 is optimal.

Algorithm 1 starts with identifying a ray that does not intersect any circles in $C$. This is $O(n \log n)$ as we may have to sort the circles by angular range to identify an angle that is not in any circle's angular range. Then, there is an $O(n \log n)$ sort of $C$ (we may be able to skip this step if we remember the sorting from the previous step). Finally, we create rays and delete circles; this will be $O(n)$ because we need to consider every circle in $C$.

Therefore, overall Algorithm 1 is $O(n \log n) + O(n) = O(n \log n)$.

(b) If we remove the requirement that there exists a ray that does not intersect any circles in $C$, we can use a modification of Algorithm 1 to find a greedy algorithm that returns a solution within 1 of optimal.

**Algorithm 2**

```
Randomly create a ray R_0 and define that ray to be angle 0.
L ← C sorted by highest angle
Delete from L all circles intersected by R_0
m ← 1
while L is not empty do
    Create ray R_m at highest angle of first circle in L
    Delete from L all circles intersected by R_m
    m ← m + 1
end while
return m
```

If the random ray is part of an optimal solution, then by the analysis in part (a) Algorithm 2 will return an optimal solution. If the random ray is not part of an optimal solution, then (again by the analysis in part (a)) Algorithm 2 will return an optimal solution over the remaining circles. This optimal solution of a reduced problem will not have a size greater than an optimal solution of the overall problem, so Algorithm 2 will return a solution with size at worst 1 ray more than the optimal solution.

The time-complexity analysis of Algorithm 2 is the same as that of Algorithm 1, except that we need not calculate a suitable angle 0 ray, as any

will do. This will not affect the overall running time, because the sorting step will still take $O(n \log n)$ time. Therefore, Algorithm 2 is $O(n \log n)$.

**Problem 3**