ASSIGNMENT 4 – COMP 252

Alexandre St-Aubin and Jonathan Campana

February 26, 2024

1. BROWSING THE SMALL ELEMENTS IN A RED-BLACK TREE.

2. GREEDY ALGORITHM. On a flat table, we have placed n disks of radii $r_1, ..., r_n$, numbered from left to right. We push them together without creating overlap, as in the figure below. Give an $O(n)$ time algorithm to compute the size of the smallest axis-aligned rectangle that can hold the disks.

   Solution:

Algorithm 1: Greedy circle packing

Input: An ordered list $\Omega := \{1, 2, 3, ..., n\}$ of n circles.
Output: The minimum width of a rectangle that can hold the disks.

```
// returns the radius of the circle a
Function Radius(circle a):
    return radius of a;

// returns the distance between the centres of a and b if they are pushed
   together.
Function centre_dist(circle a, circle b):
    return sqrt((Radius(a) + Radius(b))^2 - (Radius(a) - Radius(b))^2);

// largest subarray of Ω ending with k, decreasing in radii.
decreasing_radii[] ← new array of circles;

// distance from the left side of the rectangle to circle at index
left_to_circle[] ← new array;
left_to_circle[0] ← 0;
for i ← 1 to n do
    if i = 1 then
        left_to_circle[i] ← Radius(i);
        decreasing_radii[i] ← i;
    else
        max_d ← 0;
        // iterate through array of decreasing radii, starting with the
           smallest, until larger radius is hit
        for j ← length(decreasing_radii) − 1 to j ← 0 do
            // get the distance from the left side of the rectangle to the
               centre of circle i when it is adjacent to circle at
               decreasing_radii [j]
            max_d ← max{max_d, left_to_circle[decreasing_radii[j]] +
             centre_dist(decreasing_radii[j], i)};
            if Radius(i) ≥ Radius(j) then
                // if we encounter a circle of larger radius than i, stop
                   checking distances, as there would clearly be an overlap.
                exit for loop;
        // max_d holds the maximum distance occurring when circle i is
           adjacent to a circle in decreasing_radii.  This
        left_to_circle[i] ← max_d;
        // update array of decreasing radii.
        while Radius(i) ≥ Radius(decreasing_radii[−1]) do
            remove(decreasing_radii[−1]);
        decreasing_radii[length(decreasing_radii)] ← i;
return left_to_circle
```