Tutorial 4

Disjoint sets

Problem 21.3-3 (572): Give a sequence of m MAKE-SET, FIND-SET and UNION, n of which are MAKE-SET, that takes $\Omega(m \log n)$ time when we use union by rank only.

Solution: First do MAKE-SET of $\{x_1\}, \ldots, \{x_n\}$. Next create a tree of height $\Omega(\log n)$ by $n-1$ UNION: first UNION(x_1, x_2), UNION(x_3, x_4),..., UNION(x_{n-1}, x_n) of size 2; then create sets of size 4 by pairwise UNION of these, etc. This gives a *binomial tree* (page 527), which has $\binom{k}{i}$ of its 2^k nodes at depth i, for $i = 0, .., k$. Hence at least half of the n nodes are at depth $\geq (\log n)/2$, so each FIND-SET on these nodes takes $\Omega(\log n)$ time. Letting $m \geq 3n$, we have more than $m/3$ FIND-SET, so the total cost is $\Omega(m \log n)$.

Alternatively, stop the initial UNION sequence above when we have $\sqrt{n}$ binomial trees, each of size $\sqrt{n}$ and height $(\log n)/2$.

Problem 21.3-5 (572): Show that any sequence of m MAKE-SET, FIND-SET, and UNION, where all UNION appear before any FIND-SET, takes only $O(m)$ time if both path compression and union by rank are used?

What happens in the same situation if only path compression is used?

Solution: $n-1$ UNION take $O(n)$ time and create $n-1$ tree edges. One FIND-SET compresses each edge along the path to the root, so that later FIND-SET on nodes on this path becomes direct children to the root. Thereby cannot m FIND-SET take more time than $O(m + \text{number of edges}) = O(m+n)$.

The result is the same if only path compression is used, since the argument above does not refer to union by rank.

Computational geometry

Problem 33.1-6 (1021): Given a point $p_0 = (x_0, y_0)$ the *right horisontal ray* from p_0 is the point set $\{p_i = (x_i, y_i) : \ x_i \geq x_0, \ y_i = y_0\}$. Show how to determine in $O(1)$ time if a right horisontal ray from p_0 intersects a segment $\overrightarrow{p_1 p_2}$.

Solution: Let $p_1 = (x_1, y_1), \ p_2 = (x_2, y_2)$ and assume $x_1 < x_2$. If $x_2 < x_0$ there is no intersection. Otherwise let $p_0' = (x_2, y_0)$ and determine if $\overrightarrow{p_1 p_2}$ intersects $\overrightarrow{p_0 p_0'}$ by the method from the lecture.

Problem 33.1-7 (1021): Describe an algorithm that in $O(n)$ time decides if a point p_0 is inside a simple n-vertex polygon P.

Solution: First check that p_0 is not on the boundary of P by going through the vertices and the edges in between. This takes $O(n)$ time.

Then count the number of intersections between polygon edges and the right horisontal ray from p_0. The polygon may be represented as a list of consecutive eges, so check for each edge if it is intersected by the ray. This also takes linear time. An even number of intersections implies that p_0 is inside the polygon.

There are special cases to consider when the ray goes through a polygon vertex: if both edges to the vertex are below or above the ray, don't change the count; if one edge is below and the other above, increase the count by 1.

Problem 33.2-2 (1028): Given two segments a and b that are comparable at x, determine in $O(1)$ time if $a >_x b$ or $b >_x a$ holds.

Solution: If the the segments are nonintersecting we can use cross product. Let a's left endpoint be $p_1 = (x_1, y_1)$ and its right endpoint be $p_2 = (x_2, y_2)$. Let b's left endpoint be $p_3 = (x_3, y_3)$ and its right endpoint be $p_4 = (x_4, y_4)$.

Assume without loss of generality that $x_1 < x_3$ and consider $\overrightarrow{p_1 p_2}$ and $\overrightarrow{p_2 p_3}$. If there is a left turn at p_2, then $b >_x a$.

If a and b intersect before x, then their right endpoints determine whether $a >_x b$. If they intersect after x, compare their left endpoints instead.

Problem 33.2-4 (1028): Determine in $O(n \log n)$ time if an n-vertex polygon is simple.

Solution: Use the sweep line algorithm to decide if polygon segments intersect. Ignore intersections at common endpoint, and continue the search.

Problem 33.2-7 (1028): Find all k intersections between n segments in $O((n + k) \log n)$ time.

Solution: Assume segments are not vertical and uppdate a *sweep-line status:* a red-black tree T of segments that the sweep line intersects, and an *event list:* a priority queue ordered by x value of the $2n$ endpoints + nearest intersection points.

Events:

1. If it's a left point of a segment ℓ, then insert ℓ into T.

   If ℓ intersects a neighbor, then insert intersection point in the event list

2. If it's a right point of a segment ℓ, then delete ℓ from T.

   If ℓ's neighbors intersect ahead, then insert intersection point in the event list

3. If it's an intersection between two segments ℓ_1 and ℓ_2, then interchange their order in T.

   If new neighbors intersect ℓ_1 or ℓ_2, then insert intersection point in the event list

Sorting takes $O(n \log n)$ time. Each update of a red-black tree takes $O(\log n)$ time, while updating the priority queue for $k \leq n^2$ intersections takes $O(\log(n + k)) = O(\log n)$ time.

Total time for $2n + k$ events is thereby $O((n + k) \log n)$.

Problem 33.3-5 (1039): Construct the on-line convex hull in $O(n^2)$ time.

Solution: In $O(n)$ time decide if the new point p is outside the current hull H. If that is the case, compute the angles from p to all vertices of H, in $O(n)$ time, and connect p to the two vertices with smallest and largest angle.

Alternatively, compute in $O(n)$ time p's closest vertex q on H. From q step through the vertices of H in both directions, until we have found vertices q_l och q_r where a left and right turn is made relative to a segment from p. Connect p to q_l and q_r to get the new convex hull. Each new point thereby requires linear time, and hence the total time is $O(n^2)$.

Discussion of assignment 4

1. Remember to motivate the time complexities in (b) and (c).
2. Choose a suitable starting point.
3. Participate in the programming contest on Saturday October 2.