Problem G: Tree

Alexej Onken

February 29, 2020

Run-time analysis

- Computing the convex hull is done in $\mathcal{O}(n \log n)$ time
- Once that is being accomplished, we have to assign an evaluation number to every point in the point set P, with the help of the evaluation function $\boxed{E(k) = 2 \cdot k - 1 - \sum_{i=1}^{k} d_k}$.
- This takes us additional n time steps per iteration $\Rightarrow \mathcal{O}(n \log n) + n$.
- In addition, certain points p_k assigned an evaluation number $E(k)$ need to be searched for, which takes additional n steps $\Rightarrow \mathcal{O}(n \log n) + n + n$.
- This whole procedure will be repeated n times, since one point and one point only will be deleted from the upper hull, after a connection (p_i, p_j) is established $\Rightarrow n \cdot (\mathcal{O}(n \log n) + n + n) = \boxed{\mathcal{O}(n^2 \log n)}$.

Algorithm 1 Treesolver Algorithm

```
1: procedure TREESOLVER(points)
2:     uhp ← upperhull(points)                                  ▷ Calculating UH(P)
3:     outputlist ← emptylist          ▷ Outputlist with all connecctions (p_i, p_j)
4:     notuhp ← difference(points, uhp)
5:     if any(deg(i) == 1 and deg(j) > 1) for i, j in zip(uhp, uhp[1 :]) then
6:         outputlist ← outputlist.append((i, j))                          ▷ Case 1
7:         poins.remove(i)
8:         deg(j) ← deg(j) − 1
9:         Treesolver(points)
10:        return outputlist
11:    if all(deg(i) == 1) for i in uhp then                               ▷ Case 2
12:        if length(points) == 2 then                                  ▷ Subcase (i)
13:            outputlist ← outputlist.append((i, j))
14:            return outputlist
15:        if leftmost(notuhp) > 1 then                                 ▷ Subcase (ii)
16:            i ← leftmost(uhp)
17:            j ← leftmost(notuhp)
18:            outputlist ← outputlist.append((i, j))
19:            poins.remove(i)
20:            deg(j) ← deg(j) − 1
21:            Treesolver(points)
22:            return outputlist
23:        if leftmost(notuhp) == 1 then                                ▷ Subcase (iii)
24:            if eval(k − 1) > 0 and eval(k) <= 0 for k in points then
25:                olddeg ← deg(k)
26:                deg(k) ← eval(k − 1) + 1
27:                pointsleft ← points[0 : k]
28:                Treesolver(pointsleft)
29:                deg(k) ← olddeg − deg(k)
30:                pointsright ← points[k :]
31:                Treesolver(pointsright)
32:                return outputlist
33:    else                                                                ▷ Case 3
34:        if eval(k) == 0 for k in points then
35:            highestyleft ← maxy(points[0 : k])
36:            highestyright ← maxy(points[k :])
37:            outputlist ← outputlist.append((highestyleft, highestyright))
38:            deg(highestyleft) ← deg(highestyleft) − 1
39:            deg(highestyright) ← deg(highestyright) − 1
40:            pointsleft ← points[0 : k]
41:            pointsright ← points[k :]
42:            Treesolver(pointsleft)
43:            Treesolver(pointsright)
44:            return outputlist
```