Projektovanje algoritama

Priprema za test

Algorithm 1: HierarchicalClustering.

Input: A set P of points whose ith point, p_i, is a pair (x_i, y_i); k, the desired number of clusters.
Output: A set C of k clusters that provides a clustering of the points in P.

```
1 n ← |P|;
2 Initialize n clusters C = {C_1, ..., C_n} such that C_i = {p_i};
3 while |C| > k do
4     (C_i, C_j) ← argmin_{C_i, C_j ∈ C, i≠j} d_{C_i, C_j};
5     C ← C ∪ {C_i ∪ C_j};
6     C ← C \ {C_i, C_j};
7 return C;
```

Algorithm 2: KMeansClustering.

Input: A set P of points whose ith point, p_i, is a pair (x_i, y_i); k, the desired number of clusters; q, a number of iterations.
Output: A set C of k clusters that provides a clustering of the points in P.

```
1  n ← |P|;
2  Initialize k centers μ_1, ..., μ_k to initial values (each μ is a point in the 2D space);
3  for i ← 1 to q do
4      Initialize k empty sets C_1, ..., C_k;
5      for j = 0 to n − 1 do
6          ℓ ← argmin_{1≤f≤k} d_{p_j,μ_f};
7          C_ℓ ← C_ℓ ∪ {p_j};
8      for f = 1 to k do
9          μ_f = center(C_f)
10 return {C_1, C_2, ..., C_k};
```

Algorithm 3: SlowClosestPair.

Input: A set P of (≥ 2) points whose ith point, p_i, is a pair (x_i, y_i).

Output: A tuple (d, i, j) where d is the smallest pairwise distance of points in P, and i, j are the indices of two points whose distance is d.

```
1 (d, i, j) ← (∞, −1, −1);
2 foreach p_u ∈ P do
3     foreach p_v ∈ P (u ≠ v) do
4         (d, i, j) ← min{(d, i, j), (d_{p_u,p_v}, u, v)};   // min compares the first element of each tuple
5 return (d, i, j);
```

Algorithm 4: FastClosestPair.

Input: A set P of (≥ 2) points whose ith point, p_i, is a pair (x_i, y_i), sorted in nondecreasing order of their horizontal (x) coordinates.

Output: A tuple (d, i, j) where d is the smallest pairwise distance of the points in P, and i, j are the indices of two points whose distance is d.

```
1  n ← |P|;
2  if n ≤ 3 then
3      (d, i, j) ← SlowClosestPair(P);
4  else
5      m ← ⌊n/2⌋;
6      P_ℓ ← {p_i : 0 ≤ i ≤ m − 1}; P_r ← {p_i : m ≤ i ≤ n − 1};   // P_ℓ and P_r are also sorted
7      (d_ℓ, i_ℓ, j_ℓ) ← FastClosestPair(P_ℓ);
8      (d_r, i_r, j_r) ← FastClosestPair(P_r);
9      (d, i, j) ← min{(d_ℓ, i_ℓ, j_ℓ), (d_r, i_r + m, j_r + m)};
10     mid ← ½(x_{m−1} + x_m);                                    // center line of strip
11     (d, i, j) ← min{(d, i, j), ClosestPairStrip(P, mid, d)};
12 return (d, i, j);
```

Algorithm 5: ClosestPairStrip.

Input: A set P of points whose ith point, p_i, is a pair (x_i, y_i); mid and w, both of which are real numbers.
Output: A tuple (d, i, j) where d is the smallest pairwise distance of points in P whose horizontal (x) coordinates are within w from mid.

```
1 Let S be a list of the set {i : |x_i − mid| < w};
2 Sort the indices in S in nondecreasing order of the vertical (y) coordinates of their associated points;
3 k ← |S|;
4 (d, i, j) ← (∞, −1, −1);
5 for u ← 0 to k − 2 do
6     for v ← u + 1 to min{u + 3, k − 1} do
7         (d, i, j) ← min{(d, i, j), (d_{p_S[u], p_S[v]}, S[u], S[v])};
8 return (d, i, j);
```

thank you!