Tutorium #13

10/02/2023

Priority Queues

Procedure build($\{e_1, \ldots, e_n\}$) $M := \{e_1, \ldots, e_n\}$
Function size **return** $|M|$
Procedure insert(e) $M := M \cup \{e\}$
Function min **return** $\min M$
Function deleteMin $e := \min M$; $M := M \setminus \{e\}$; **return** e
Function remove(h : Handle) $e := h$; $M := M \setminus \{e\}$; **return** e
Procedure decreaseKey(h : Handle, k : Key) **assert** key(h) $\geq k$; key(h) $:= k$
Procedure merge(M') $M := M \cup M'$

address.

- Pairing Heaps
  $\hat{=}$ Forests + min Ptr
- Fibonacci Heaps
  FAST UNION BY RANK

- Binary Heap

- Monotone PQ (e.g. Bucket Queue)

advanced graph algorithms

- shortest paths (Dijkstra, A^*, Bellman Ford, BFS)

  only $\mathbb{R}^+$, bidirectional ; potential ; negative cycle Detect.

  ↳ also all pair shortest paths

- Strongly connected comp.

  ↳ BFS / DFS scheme

Max flow

- Definition
- ILP/LP
- Ford Fulkerson (augmenting path, residual graph)
- Dinic
- Preflow push

$\boxed{\text{MIN CUT}}$

techniques to solve problems to optimality

- Dynamic programming (when to use, run time, ...)
- branch & bound (fixed parameter tractable $O(f(k) \cdot p(n))$)
- branch & reduce — Independent set reduction rules
- linear programming

ILP = NP hard

LP = P

Duality, examples

Greedy Algo.

choose best local solution , examples(!)

Approximation Algo

- $\dfrac{f(x(I))}{f(x^*(I))} \leq \rho$ ← approx factor

- non-approx. (Traveling Salesman ; easy to see:

$HC \leq_p \alpha\text{-approx TSP}$)

Geometric Algos

- Graham Scan
- Smallest enclosing ball (Idea of probabilistic proof)

Parallel Algo

Why parallel?

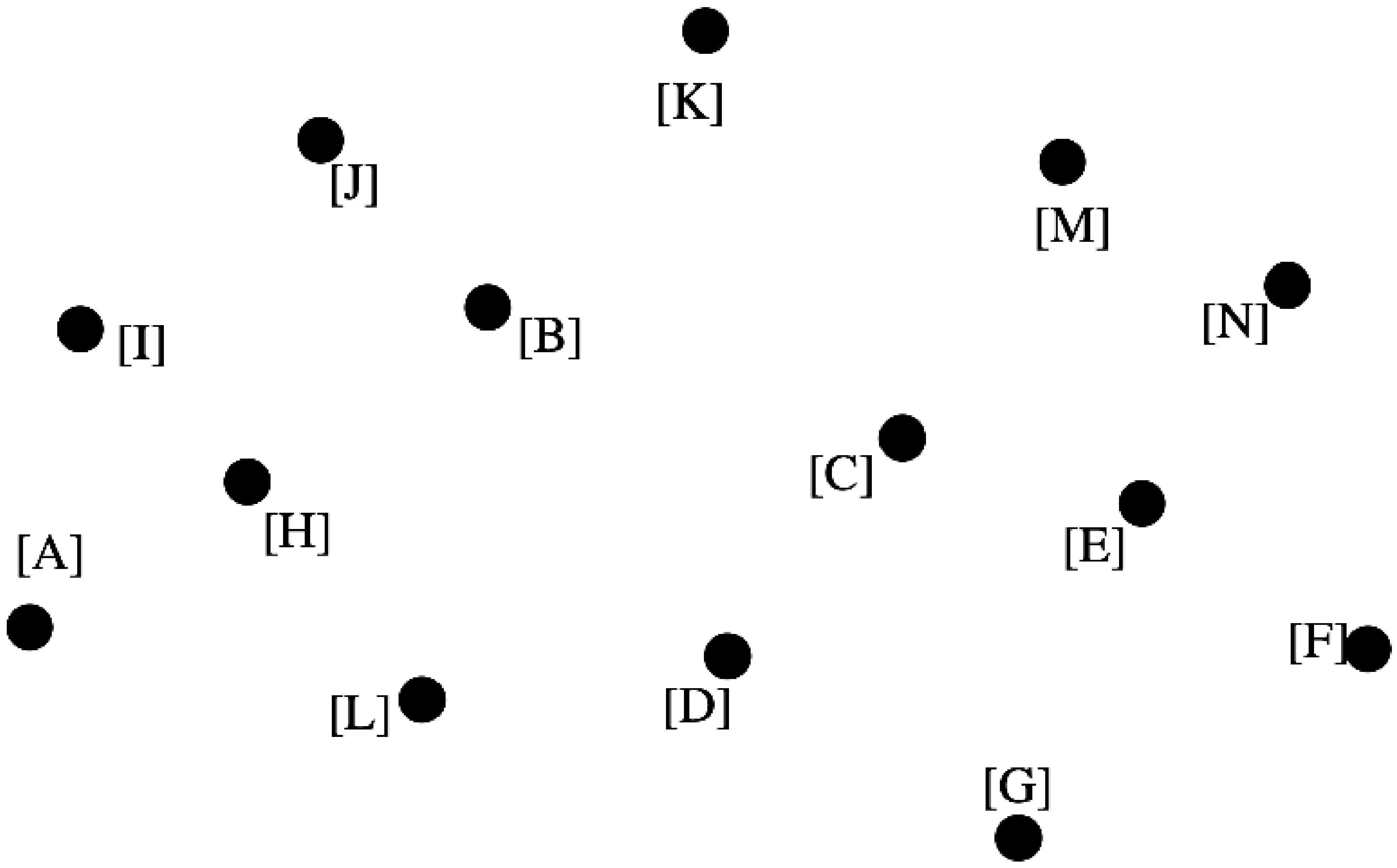
Analyse (Speedup, ...)

Sorting

Extern. mem. algos

Sorting

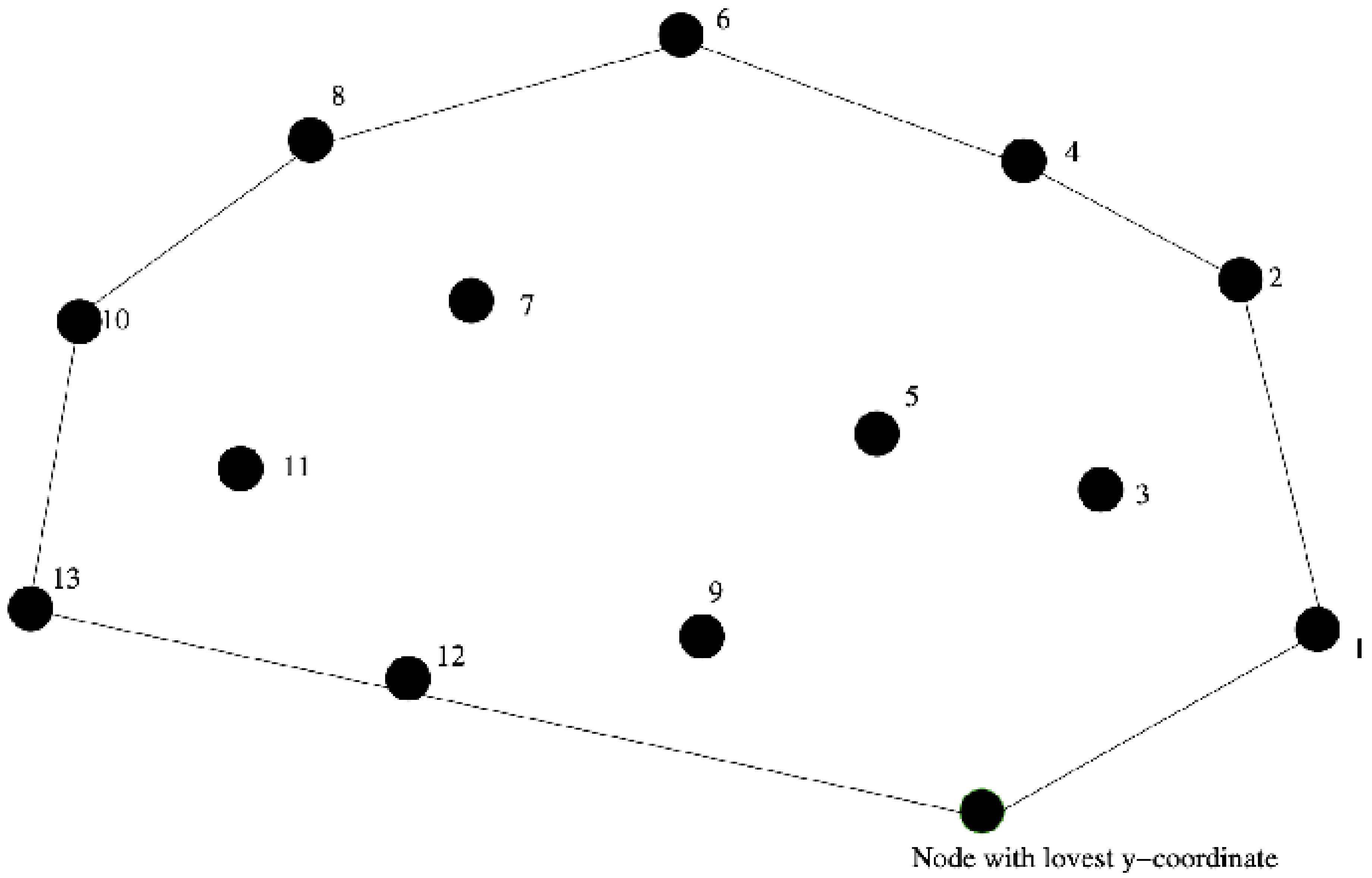
Problem 1 (2 points) Given is the following point set. Draw the border of the convex hull of the point set, mark the start node of the Graham-Scan algorithm and give the order in which the Graham-Scan algorithm will consider the points by writing numbers next to the points.



A more detailed algorithm

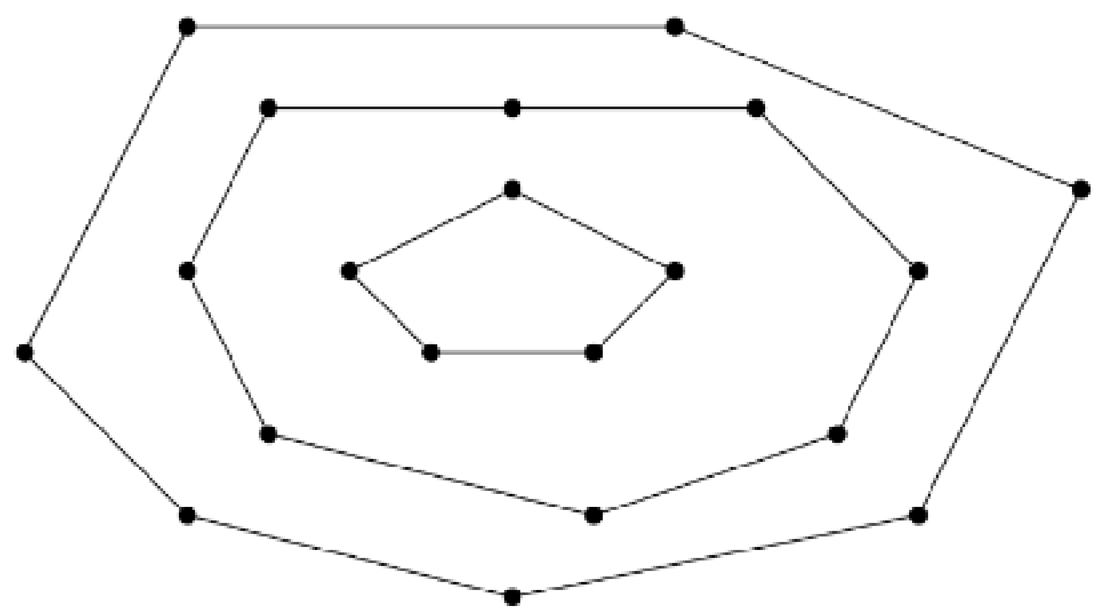
```
GRAHAM-SCAN(Q)
let p0 be the point in Q with the minimum y-coordinate,
        or the leftmost such point in case of a tie
let ⟨p1, p2, ..., pm⟩ be the remaining points in Q,
        sorted by polar angle in counterclockwise order around p0
        (if more than one point has the same angle, remove all but
        the one that is farthest from p0)
PUSH(p0, S)
PUSH(p1, S)
PUSH(p2, S)
for i ← 3 to m
    do while the angle formed by points NEXT-TO-TOP(S), TOP(S),
                and pi makes a nonleft turn
            do POP(S)
        PUSH(pi, S)
return S
```

Solution:



Problem 2 (6 points) Given is a finite, non-empty set of points $P \subset \mathbf{R}^2$. Let $H(S)$ be the *boundary* of the convex hull of a subset $S \subseteq P$. The layers of P are defined recursively:
Base case: $S_1 = H(P)$
General case: $S_i = H(P \backslash (S_1 \cup \ldots \cup S_{i-1}))$ for $i > 1$

We now consider the following problem. We want to find the layers $S_1, \ldots, S_k$ of P such that $S_k \neq \emptyset$ and $S_{k+1} = \emptyset$. The following figure shows a point set with $k = 3$ non-empty sets.

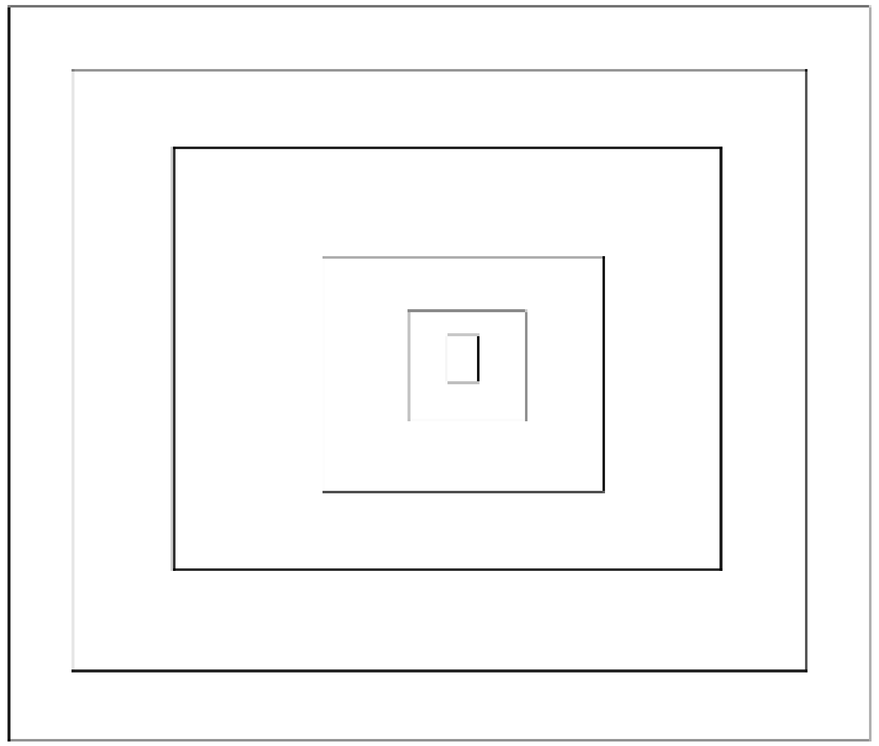


1. (3 points) Develop an algorithm that given a set of points outputs the layers of P in time $O(n^2 \log n)$. In particular, prove the running time of your algorithm.

2. (3 points) Give a family of point sets $P_1, P_2, \ldots,$ with $|P_n| \in O(n)$ such that your algorithm achieves the worst-case for those instances, i.e. a family of instances that the running time of your algorithm is $\Omega(n^2 \log n)$.

1. Our algorithm works iteratively as follows: compute the convex hull of the point set. This is the set S_1. Now remove the points and repeat to compute S_2, and so on and so forth until the point set is empty. This solves the problem. The running time of one iteration of convex hull computations is $O(n \log n)$ (e.g. using Graham scan). In each iteration, we remove at least three nodes (for non trivial instances, otherwise it may be two). Hence, after $O(n)$ iterations the remaining set is empty. Overall, this yields time $O(n^2 \log n)$ running time.

2. Consider a sequence of nestes points arrange on squares as shown below:

   Given an instance P_i, one generates the next instance P_{i+1} by adding four points outside of the convex hull of P_i in the same manner (on a square). In this instance, our algorithm removes four points in each iteration. Hence, we get to a running time of $\Omega(n^2 \log n)$.

Problem 3 (8 points)

Prove that in the procedure $GRAHAM - SCAN$, points p_1 and p_m must be vertices of $CH(Q)$.

Solution:

Assume p_1 is not in $CH(Q)$. Therefore there must be a vertex p_x where the sequence $< p_0, p_1, p_x >$ forms a right turn. This only happens when the polar angle of p_x (with p_0 as the source) is lower than the polar angle of p_1. This is not possible as p_1 is by definition the vertex with the lowest polar angle. The proof for p_m is equivalent.