Exercise sheet 2

Max Wisniewski, Alexander Steen

Problem 1 Convexity

a) Let $\{C_i\}_{i\in I}$ be a set of convex sets. Show that $\bigcap_{i\in I} C_i$ is convex.

Proof: Since each C_i is convex, it holds that the line $\overline{pq}$, for $p, q \in C_i$, is completely contained in C_i. Since for all $p, q \in \bigcap_{i\in I} C_i$, the segment $\overline{pq}$ is contained in each C_i, it holds that $\overline{pq} \in \bigcap_{i\in I} C_i$.
$\Rightarrow \bigcap_{i\in I} C_i$ is convex. □

A similar property can be found for unions of convex sets:
Claim: For any non-decreasing series of convex sets $(C_i)_{i\in I}$ (with respect to set inclusion), the set $\bigcup_{i\in I} C_i$ is convex.
Proof: For all $p, q \in \bigcup_{i\in I} C_i$ there exists a $\tilde{i} \in I$ such that $p, q \in C_{\tilde{i}}$, hence $\overline{pq} \in C_{\tilde{i}}$. Since $(C_i)_{i\in I}$ is non-decreasing, $\overline{pq} \in \bigcup_{i\in I} C_i$. □

b) Let P be a finite point set in the plane. Show that the boundary of the convex hull $CH(P)$ of P is a convex polygon whose vertices are points of P.

Proof:
We first prove, that $CH(P)$ is a polygon and second, the points of the polygon are points of P.

Let C be an arbitrary convex set containing P. Assume C is not a polygon. Fix a point x on that curve [1]. We now take $\varepsilon > 0$ steps on the curve to the point x_ε and look at the set C' where we substituted the line $\overline{xx_\varepsilon}$ for the curve segment that connected them before.
We know if ε is small enough there is no point of P in the cut part. If the curve was bend left, the resulting line will only do left turns at the end. Therefore C' is convex.
Hence C could not be the convex hull.

Next we fix a convex polygon CP that has points of P on the nodes. This can be found by intersecting all halfplanes that contain all points in P as in the Brute-Force algorithm.

Because CP is a convex set containing P only $CH(P) \subset CP$ can hold.

Assume there is a point on the polygon $CH(P)$ $y = p_k \notin P$. Then $p_{k-1}p_kp_{k+1}$ is a triangle pointing to the outside of the polygon. Otherwise we would have taken a left turn in cw order. But this is not possible due to the previous shown fact, that a polygon only of points of P is a convex set containing P. Hence p_k

[1] We assume complete sets. If the set has no boundary we consider a series of points converging to the boundary.

can not be on the hull.

□

c) Show that the segment between two points $p, q \in P$ is an edge of $CH(P)$ if and only if all points of P lie on the same side of the line through p and q.

Proof:
"⇒": By contraposition. Let $\tilde{p}$ a point on the one side of the line through p and q and $\tilde{q}$ a point on the other side. Let $s \in \overline{pq}$. Then, one of $\overline{s\tilde{p}}$ and $\overline{s\tilde{q}}$ is not contained in P. Hence, $\overline{pq}$ cannot be an edge of $CH(P)$.
"⇐":
By the previous part, we know that the $CH(P)$ is a polygon of the points in P. Because $p, q \in P$ we know that $\overline{pq} \in CH(P)$.

Assume $\overline{pq}$ is not on the boundary. Then there exists points $u, v \in P$ on both sides of $\overline{pq}$ by the previous part. But this is not possible due to the premise.

□

Problem 2 Computing the maximal tangent to a subconvex hull in Chens Algorithm in $O(logh^*)$.

Proof:

Let p_{k-1}, p_k be the last computed points on the convex hull. Let $q_1, ..., q_{h^*}$ be the points on the convex hull of the subconvex hull given in ccw or cw order. Then we will compute the point on the hull that maximizes the angle as follows. Assume we have the points given in an array `hull`.

```
pos ← 1
h'← h* / 2
while h' > 0 do
  l ← deg(p_{k-1},p_k,hull[pos-1 mod h^*]))
  c ← deg(p_{k-1},p_k,hull[pos mod h^*]))
  r ← deg(p_{k-1},p_k,hull[pos+1 mod h^*]))
  if l < c && c < r || l == r
    pos ← pos + h'␣mod␣h*
  else if l > c && c > r
    pos ← pos - h' mod h*
  h'← h' / 2
od
return pos
```

Next we have to verify the algorithm.

Claim 1. *The above algorithm maximizes the angle and can be computed in* $O(\log h^*)$ *time.*

⌟

Proof 1:
The running time of the algorithm is obviously $O(\log h^*)$. We start with $\frac{h^*}{2}$. In the while loop we only compute for constant time c. We decrease h' until we hit zero.

$$\Rightarrow T(n) = logh^*.$$

We know that $\sum_{i=1}^{\infty} \frac{h^*}{2} = h^*$ and if our steps are discrete we can reach the point in $logh^*$ time (Binary Search). Let R_i be the set of reachable points on the hull in

step i of the algorithm and q the optimal point. Then

$$\forall i \in \mathbb{N} : q \in R_i, |R_i| = \frac{h^*}{2^i}$$

Induction on i.

I.A. $i = 1$

In the first step we can do at most $\sum_{i=1}^{\infty} \frac{h^*}{2^i} = h^*$ steps. Therefore all points are reachable especially q.

I.S. $i \rightsquigarrow i + 1$

We now in $q \in R_i$. On the last step we were according to the algorithm at one endpoint and we were on the edge of R_i. In the algorithm we divide `h'` by two, therefore we are in the middle of the set according to the given order because it had size `h'` $= |R_i|$. We divide R_i into two subsets R_{i+1}^l and R_{i+1}^r. If the first case is true, we know $q \in R_{i+1}^l$ because the hull was convex therefore we can only decrease the angle again on the other side of the optimum. We take $R_{i+1} = R_{i+1}^l$ as the algorithm says. The other case is similar with R_{i+1}^r. $|R_{i+1}| = |\frac{R_i}{2}|$.

□

Problem 3 Chan's algorithm and superexponential search

a) Show that the first h^* points of the convex hull can be computed in $O(n)$ time given the r subconvex hulls.

Solution:

Given an two dimensional array `hulls` where `hull[i]` is the array of points of the i-th hull and `hull[i][j]` is the j-th point of the i-th hull in ccw order.

Let p_0, p_1 be the first points as given in the algorithm. The array `act` contains for each hull the index of the point that maximizes the angle in the current step. Then we compute the algorithm as follows.

```
int[n] act
compute act[i] with algorithm in ex. 2
for k = 2 to h* - 1
    for i = 1 to n / h*
        while angle(p_{k-1}, p_k, hull[i][act[i]]) < angle((
            p_{k-1}, p_k, hull[i][act[i+1 mod h*]])
              act[i] = (act[i] + 1) mod h*
    i' = argmax_{0≤i<h*/n}  angle((p_{k-1}, p_k, hull[i][act[i+1 mod h*]]
    p_{k+1} = hull[i'][act[i']]
```

After the initial searching of the max we will just proceed in the ccw order of the hull.

Claim 2. *The above algorithm computes the first h^* points of the convex hull and runs in $O(n)$ time.*

⌟

Proof 2. Given the correctness of chan's algorithm we only changed the computation of the next maximum angle in each step. We do not do a binary search in each step, but go through the list linear. In this iteration we will find of course the maximum angle, due to the termination criterion of the while loop. This maximum exists and will therefor be found.

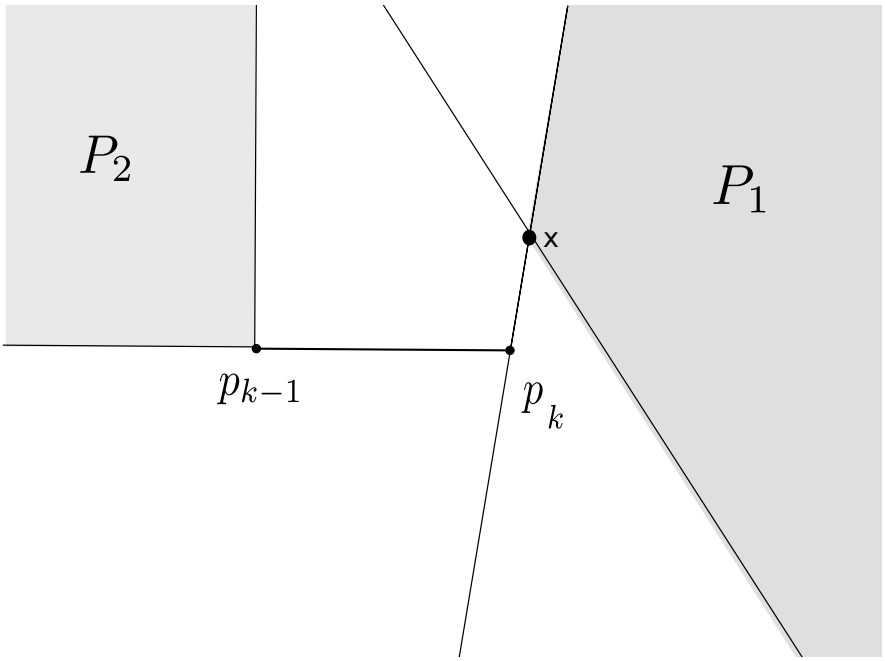

Figure 1: Spaces able to consider x as a point.

Next we show, that each point on the small convex hulls will only be considered twice through all iterations.

Let $0 \leq i < n/h^*$ be arbitrary but fixed the i-th hull and $1 \leq x \leq h^*$ be the index of the x-point of the i-th hull.

If we started with `hull[i][x]` it is on the hull. Therefor if we get the index again we will eventually after finite steps take this point into the hull. Hence we consider x to be a maximal point in round $k > 1$. Let p_{k-1}, p_k the last computed points on the hull. If we look at the picture in figure 1 if we assume the convex polygon has a border after x at most at the line $\overline{p_k x}$ and assumed somewhere on the crossing line. The next point from which x is no longer the maximizing point is element of P_1 potentially x itself.
If we want to leave x behind us a second time we have to cross p_{k-1} again. But to come by p_{k-1} we have to be up left in P_2 with some point p_{k+t} such that it is convex. But there is no way we can cross p_{k-1}[2] We can either connect to p_{k-1} such that the polygon is closed in the algorithm finishes. Or we come above or below p_{k-1}. This is not possible because all lines of the algorithm are borders of the polygon. Now either $\overline{p_{k+t}p_{k+t+1}}$ is on the left of $\overline{p_{k-1}p_k}$ in which case $\overline{p_{k+t}p_{k+t+1}}$ is no borderline or it is the other way around.

We conclude, that it is not possible to enter P_1 a second time, such that we will consider x only twice.

This shows us, that the innermost loop can only skip each element once.
The running time consists of

1) Computing the n/h^* starting points on the hulls. For each hull we can compute the point in $O(\log h^*)$.

[2]By crossing we mean passing the line given by the normal on $\overline{p_{k-1}p_k}$ and and the point p_{k-1} itself.

$\Rightarrow O(n)$ total running time.

2) The internal for and while loops consider all points at most twice as proven above. This gives a runtime of $O(n)$ in total.
3) The maximum can be found in $O(n/h^*)$ and we compute it h^* times. The running time therefor is $O(n)$ in total.

The algorithm has a running time of $T(n) = 3 \cdot O(n) = O(n)$. □

b) Show that by 2^{2^i} as the iteration for h^* we obtain a total running time of $O(n \log h)$.

Proof:

From the lecture we know, that the running time for one iteration of the algorithm with h^* has the running time $O(n \log h^*)$. Iteration our h^* gives us that

$$h^*_{i+1} = 2^{2^{i+1}} = \left(2^{2^i}\right)^2 = (h^*_i)^2$$

hence if we terminate in step n it holds that $h^*_n \leq h^2$. Or we should have terminated before.

$$h^*_n = 2^{2^n} \leq h^2 \Rightarrow n \leq (\log \log h) + 1 \tag{1}$$

Therefor the total running time of all iterations is

$$\begin{aligned} T(n,h) &= \sum_{n=1}^{\log\log h+1} n \log h^*_n = n \sum_{n=1}^{\log\log h+1} \log h^*_n \\ &= n \sum_{n=1}^{\log\log h+1} \log 2^{2^n} = n \sum_{n=1}^{\log\log h+1} 2^n \\ &= n \frac{2^{\log\log h+2} - 1}{2-1} \\ &= 4 \cdot n \cdot 2^{\log\log h} = 4n \log h \\ &= O(n \log h) \end{aligned} \tag{2}$$

□