Solution to HW 3

CS 473: Algorithms, Fall 2018

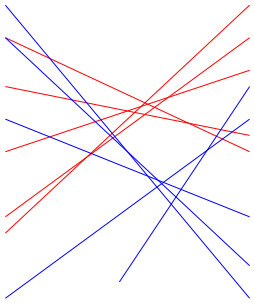
(due Wednesday, at noon, September 19, 2018)

Version: **1.00**

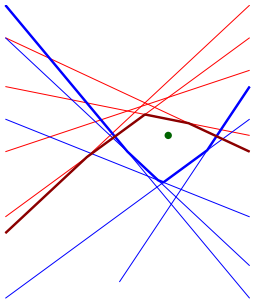
---

1 (100 PTS.) Sandwich point.

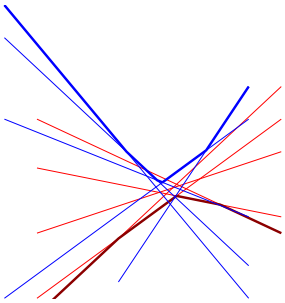
Let T and B be two sets of lines in the plane, in general position, such that $|T| + |B| = n$. The lines of T are top, and the lines of B are bottom. Here, general position implies that no two lines of $T \cup B$ are parallel, no line of $T \cup B$ is vertical or horizontal, no three lines of $T \cup B$ meets in a point, and no two vertices have the same x coordinate (a vertex is a point of intersection of two lines). Here, we are interested in computing in linear time a point that is above all the bottom lines of B and below all the top lines of T. Such a point is ***feasible***. (Here, a point that is on a line, can be considered to be either above or below the line, as convenient.) Here are a few examples:



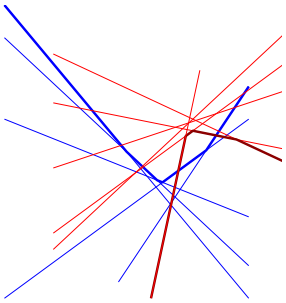
Input



feasible point



Infeasible input



Another feasible input.

We interpret a line $\ell \equiv y = ax + b \in T \cup B$ as a function from $\mathbb{R}$ to $\mathbb{R}$. In particular, for $z \in \mathbb{R}$, let $\ell(z)$ be the y coordinate of $\ell \cap (x = z)$ – that is, $\ell(z) = az + b$.

1.A. (10 PTS.) For $x \in \mathbb{R}$, let $U_B(x) = \max_{\ell \in B} \ell(x)$ be the ***upper envelope*** of B. Similarly, let $L_T(x) = \min_{\ell \in T} \ell(B)$ be the ***lower envelope*** of T. Prove that $U_B(x)$ is a convex function, and $L_T(x)$ is a concave function.

Solution: Consider any two values $x_1, x_2 \in \mathbb{R}$. Observe that $p_1 = (x_1, U_B(x_1))$ and $p_2 = (x_2, U_B(x_2))$ are two points that are above all the bottom lines. As such, the segment p_1p_2 is above each one of these lines, and as such, the segment p_1p_2 is above the image of $U_r(x)$, which implies convexity.

The argument for $L_T(x)$ is symmetric.

1.B. (10 PTS.) Consider the function $Z(x) = L_T(x) - U_B(x)$. Argue that $Z(x)$ is concave, and show how to compute its value at point x in linear time.

Solution:

The negation of a convex function is concave, and the sum of two concave function is concave.

Computing $Z(x)$ require computing the values of $\ell(x)$, for all $\ell \in R \cup B$, which takes linear time. We then compute the maximum of all the top values, and the minimum of all the bottom ones, and take the difference. Clearly this takes linear time.

1.C. (10 PTS.) Prove that $Z(x) \geq 0$ for some $x \iff$ there a point above all the top lines, and below all the bottom lines.

Solution:

$Z(x) \geq 0 \implies Z(x) = L_T(x) - U_B(x) \geq 0 \implies L_T(x) \geq U_T(x)$. In particular, the point $p = \Big(x, \big(L_T(x) + U_B(x)\big)/2\Big)$ is above all the bottom lines, and below all the top lines.

Similarly, given a feasible point $p = (x, y)$, we have that $L_T(x) \geq y \geq U_B(x)$, which implies that $Z(x) = L_T(x) - U_B(x) \geq 0$.

1.D. (20 PTS.) Let x^* be the value where $Z(\cdot)$ achieves its maximum. Let $X = \{x_1, \ldots, x_m\}$ be a set of m numbers, and let $z_1 = \mathit{Select}(X, \lfloor m/2 \rfloor - 1)$, $z_2 = \mathit{Select}(X, \lfloor m/2 \rfloor)$, and $z_3 = \mathit{Select}(X, \lfloor m/2 \rfloor + 1)$, where $\mathit{Select}(X, i)$ is the ith rank element in X.

Show, how in linear time, one can decide if $x^* \in [z_1, \infty]$ or $x^* \in [-\infty, z_3]$,

Note, that this implies that for roughly half the values of X, this decides that the maximum of $Z(\cdot)$ is either bigger or smaller than them.

Solution:

Compute $v_1 = Z(z_1), v_2 = Z(z_2), v_3 = Z(z_3)$.

If $v_1 \leq v_2 \leq v_3$, then by the concavity of Z, the maximum can not be in the interval $[z_1, z_2]$, and similarly it can not be in $[-\infty, z_2$, and as such $x^* \in [z_1, \infty]$ in this case.

If $v_1 \geq v_2 \geq v_3$, then by the concavity of Z, the maximum can not be in the interval $[z_2, z_3]$, and similarly it can not be in $[z_3, \infty$, and as such $x^* \in [-\infty, z_3]$ in this case.

If $v_1 \leq v_2$ and $v_2 \geq v_3$, then by the concavity of Z, the maximum must be somewhere in the interval $[z_1, z_3]$, and as such $x^* \in [-\infty, z_3]$ in this case.

Note, that by the concavity of $Z(\cdot)$ there are no other options.

1.E. (20 PTS.) For the case that $|T| \geq |B|$, show how, in linear time, either

(i) compute a feasible point of T and B, or

(ii) compute a set $T' \subseteq T$, such that the maximum of $Z_{T,B}(\cdot)$ and $Z_{T',B}(\cdot)$ is the same, where $Z_{T',B}$ is the function Z described above for the sets T' and B.

(Hint: Follow the algorithm seen in class.)

Solution:

Match the lines of T in pairs ℓ_i, ℓ'_i, for $i = 1, \ldots, t = |T|/2$. Let $p_i = \ell_i \cap \ell'_i$, and let $x_i = x(p_i)$, for all i. Let $X = \{x_1, \ldots, x_t\}$. Compute z_1, z_2, z_3 as the median and its two immediate neighbors in X, as suggested above. Compute $v_1 = Z(z_1)$, $v_2 = Z(z_2)$ and $v_3 = Z(z_3)$. If any of these values is non-negative, then the algorithm found a feasible point.

Otherwise, use the algorithm of **(1.D.)**, to decide:

(I) If $x^* \in [-\infty, z_3]$, then $p_{t/2+1}, p_{t/2+2}, \ldots, p_t$ are all to the right of x^*. Consider a point p_i in this set – it is the intersection of ℓ_i and ℓ'_i, and assume that ℓ_i has a bigger slope than ℓ'_i. But then ℓ_i is below ℓ'_i to the left of p_i, which implies that ℓ'_i can not contribute to the lower envelope of the top lines. As such, we can throw it away from T. Let T' be the remaining lines. We have that $|T'| \leq (7/8)|T|$, Observe that $U_{T,B}(x) = U_{T',B}(x)$ are the same for $x < z_3$, which implies that $Z_{T,B}(x^*) = Z_{T',B}(x^*)$, and X^* is the point of maximum for both functions.

(II) The case $x^* \in [z_1, \infty]$ is handled symmetrically.

1.F. (30 PTS.) Given T and B, describe a linear time algorithm for computing a feasible point if it exists. If it does not, the algorithm should output that there is no such point.

Solution:

If $n = |T| + |B| = O(1)$ then we can solve the problem by brute force – compute the all the $\binom{n}{2}$ vertices of the lines of $T \cup B$. If any of them is feasible, then we are done. Otherwise, the input is not feasible.

Otherwise, if $|T| \geq |B|$, then we use the above algorithm, that in linear time reduces T size by a factor of (7/8)th. A symmetric algorithm can be applied to B if $|B| \geq |T|$. As such, in each iteration, either we find a feasible solution, or we reduce the problem size by a constant factor. As such, this algorithm solve the problem and runs in linear time.

2 (100 PTS.) Absolutely not subset sum.

Let $B = \{b_1, \ldots, b_m\} \subseteq [\![U]\!] = \{1, 2, \ldots, U\}$. A number $t \leq U$ is ***n-representable*** by B, if there exists integer numbers $\alpha_i \geq 0$, for $i = 1, \ldots, m$, such that

(i) $\sum_{i=1}^{m} \alpha_i = n$, and

(ii) $\sum_{i=1}^{m} \alpha_i b_i = t$.

Show how to compute, as fast as possible, if t is n-representable by B by an algorithm with running time close to linear in m and U (the dependency of the running time on n should be polylogarithmic in n).

[Hint: Use FFT.]

To make life easy for you, I broke it into three steps:

2.A. (30 PTS.) Show how to solve the case $n = 2$.

> **Solution:** Set $p(x) = \sum_{i=1}^{m} x^{b_i}$ and compute the polynomial $q(x) = p(x) \cdot p(x)$ using FFT. Clearly, the coefficient of x^i is non-zero if and only if there are two numbers $\gamma_1, \gamma_2 \in B$, such that $\gamma_1 + \gamma_2 = i$. As such, the desired set is powers of all the non-zero coefficients of $q(x)$.
> Let
> $$[q(x)] = B \oplus B = \left\{ i \in [\![U]\!] \;\middle|\; x^i \text{ has non-zero coefficient of } q(x) \right\}$$
> be the output set.
> The running time is $O(U \log U)$.

2.B. (20 PTS.) Show how to solve the case that n is a power of 2.

> **Solution:** Assume $n = 2^{\Delta}$. Just apply part (A) repeatedly Δ times. That is, let $B_0 = B$, and let $B_i = B_{i-1} \oplus B_{i-1}$. Clearly, this takes $O(U\Delta \log U) = O(U \log U \log n)$.

2.C. (50 PTS.) Show how to solve the general problem.

> **Solution:** Let $\Delta = \lfloor log_2 n \rfloor$. Compute $B_0, \ldots, B_{\Delta}$ as suggested above. Let $n = \sum_{i=0}^{\Delta} 2^i \alpha_i$ be the binary representation of n, and observe that the desired quantity is just the product of all the B_is for which α_i is non zero. This can be computed using Δ applications of FFT, as done in (A) and (B). This gives us the desired result in $O(U \log U \log n)$ time overall.

(As usual, the solutions to (A) and (B) are much simpler than (C), and are useful in solving (C).)

3 (100 PTS.) Computing Polynomials Quickly

In the following, assume that given two polynomials $p(x), q(x)$ of degree at most n, one can compute the polynomial remainder of $p(x) \bmod q(x)$ in $O(n \log n)$ time. The ***remainder*** of $r(x) = p(x) \bmod q(x)$ is the unique polynomial of degree smaller than this of $q(x)$, such that $p(x) = q(x) * d(x) + r(x)$, where $d(x)$ is a polynomial.

Let $p(x) = \sum_{i=0}^{n-1} a_i x^i$ be a given polynomial.

3.A. (25 PTS.) Prove that $p(x) \bmod (x - z) = p(z)$, for all z.

> **Solution:** By the definition of mod, we can write $p(x) = q(x) \cdot (x - z) + r(x)$ where $r(x)$ is a polynomial of degree one less then $x - z$ (i.e., its degree is zero – its a constant). In particular, $p(z) = q(z)(z - z) + r(z) = r(z)$. As such $p(x) \bmod (x - z) = p(z)$.

3.B. (25 PTS.) We want to evaluate $p(\cdot)$ on the points $x_0, x_1, \ldots, x_{n-1}$. Let

$$P_{ij}(x) = \prod_{k=i}^{j} (x - x_k)$$

and

$$Q_{ij}(x) = p(x) \bmod P_{ij}(x).$$

Observe that the degree of Q_{ij} is at most $j - i$.
Prove that, for all x, $Q_{kk}(x) = p(x_k)$ and $Q_{0,n-1}(x) = p(x)$.

Solution:

By definition $P_{kk}(x) = x - x_k$ and by (A), we know that

$$Q_{kk}(x) = p(x) \bmod P_{kk}(x) = p(x) \bmod (x - x_k) = p(x_k).$$

Lemma 0.1. *Let $p_1(x)$ and $q_1(x)$ be two arbitrary polynomials, such that the degree of $p_1(x)$ is strictly smaller than the degree of $q_1(x)$. Then $p_1(x) \bmod q_1(x) = p_1(x)$.*

Proof: Indeed, we have $p_1(x) = 0 * q_1(x) + p_1(x)$ and the degree of $p_1(x)$ is smaller than the degree $q_1(x)$. Thus, by the definition of the reminder we have $p_1(x) = p_1(x) \bmod q_1(x)$. ■

Now, $p(x)$ is of degree $n - 1$ and $P_{0,n-1}(x)$ is of degree n. As such, by the above lemma, we have

$$Q_{0,n-1}(x) = p(x) \bmod P_{0,n-1}(x) = p(x).$$

3.C. (25 PTS.) Prove that for $i \leq k \leq j$, we have

$$\forall x \quad Q_{ik}(x) = Q_{ij}(x) \bmod P_{ik}(x)$$

and

$$\forall x \quad Q_{kj}(x) = Q_{ij}(x) \bmod P_{kj}(x).$$

Solution:

We need the following easy lemma.

Lemma 0.2. *For any three polynomials $a(x), b(x)$ and $c(x)$ we have*

$$a(x) \bmod b(x) = (a(x) \bmod b(x)c(x)) \bmod b(x).$$

Proof: Let $r(x) = a(x) \bmod b(x)c(x)$ and $q(x)$ be the unique polynomial such that $a(x) = q(x)b(x)c(x) + r(x)$. Similarly, let

$$r'(x) = r(x) \bmod b(x) = (a(x) \bmod b(x)c(x)) \bmod b(x)$$

and $q'(x)$ be the unique polynomial such that $r(x) = q'(x)b(x) + r'(x)$. Now, we have

$$\begin{aligned} \forall x \qquad a(x) &= q(x)b(x)c(x) + r(x) = q(x)b(x)c(x) + q'(x)b(x) + r'(x) \\ &= \big(q(x)c(x) + q'(x)\big)b(x) + r'(x). \end{aligned}$$

Now, by the definition of $r'(x)$ its degree is strictly smaller than the degree of $b(x)$, and we conclude that $r'(x)$ is the reminder of $p(x) \bmod b(x)$, establishing the claim. ■

Now, the claim that $Q_{ik}(x) = Q_{ij}(x) \bmod P_{ik}(x)$ is equivalent (by definition), to the claim that

$$p(x) \bmod P_{ik}(x) = (p(x) \bmod P_{ij}(x)) \bmod P_{ik}(x).$$

But $P_{ij}(x) = P_{ik}(x)P_{(k+1)j}(x)$. As such, the claim is that

$$p(x) \bmod P_{ik}(x) = \big(p(x) \bmod P_{ik}(x)P_{(k+1)j}(x)\big) \bmod P_{ik}(x).$$

But this is implied immediately by the lemma, by setting $a(x) = p(x)$, $b(x) = P_{ik}(x)$ and $c(x) = P_{(k+1)j}(x)$.
The second claim follows in a similar fashion.

3.D. (25 PTS.) Given an $O(n \log^2 n)$ time algorithm to evaluate $p(x_0), \ldots, p(x_{n-1})$. Here $x_0, \ldots, x_{n-1}$ are n given real numbers.

Solution:

By the above, we need to compute, for all k, the quantity $Q_{kk}(x) = p(x_k)$. Now, we do this recursively by computing (top down) the polynomials $Q_{ij}(x)$ (for some values of i and j). Indeed, given $Q_{ij}(x)$ we can compute $Q_{il}(x)$ and $Q_{(l+1)k}(x)$ by using the two identities from (c), where $l = \lfloor (i+j)/2 \rfloor$. This takes $O(d \log d)$ time, where $d = j - i + 1$ is the upper bound on the degrees of these polynomials. Continuing this recursively, would yield all the polynomials $Q_{ii}(x), \ldots, q_{jj}(x)$.

As such, starting with $Q_{0(n-1)}(x) = p(x)$, we can perform the above recursive algorithm to get the desired polynomials $Q_{00}(x), \ldots, Q_{(n-1)(n-1)}(x)$, which are (luckily for us) exactly the numbers $p(x_0), \ldots, p(x_{n-1})$.

As for the running time, the recurrence is $T(n) = O(n \log n) + 2T(n/2) = O(n \log^2 n)$.