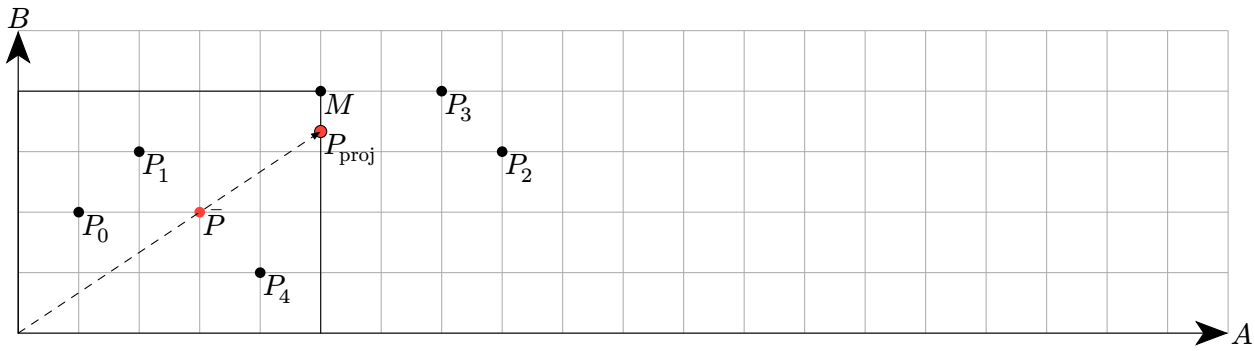
Soft drinks - solution

Atanas Dimitrov, Radoslav Dimitrov

Core view

It proves very useful to look at the problem geometrically. We can think of each drink with its concentration of sugar A_i and acids B_i as a point $P_i = (A_i, B_i)$ – from this point forward drink type and point will be used interchangeably. The order of the points does not affect the queries. Therefore, we can sort them in increasing order of A_i. Additionally, if we have multiple points with the same A_i, we can discard all of them except the one with minimal B_i, since all of those points will appear in the same set of queries and we can mix out those suboptimal points.

Thinking about the general problem statement, we can shift our focus from choosing the final drink amounts $(V_0, V_1, ..., V_{N-1})$ of each drink type to instead choosing the per-liter amount of each drink type, i.e. $(\bar{V}_0, \bar{V}_1, ..., \bar{V}_{N-1})$, s.t. $\sum_{i=0}^{N-1} \bar{V}_i = 1$, and then figuring out the maximum liters of this 1-liter drink we can mix. Geometrically we can visualize this by picking a weighted average $\bar{P}$ of the points P_i and then "projecting" it onto the rectangle with corners $(0, 0)$ and $M = (M_A, M_B)$:



Solution for $N = 2$

We examine 3 different cases:

1. If none of the points fit in the interval (L_A, B_A) the answer is trivially 0.
2. If only one of the points fits in the interval (L_A, B_A), the answer is the amount of liters we can pick from this point P_i (taking special care of the cases $A_i = 0$ and $B_i = 0$): $\min\left(\frac{M_A}{A_i}, \frac{M_B}{B_i}\right)$
3. If both points fit in the interval (L_A, B_A) we need to analyze a bit more careful:

First, notice that if $A_0 < A_1$ and $B_0 \leq B_1$ we can discard point 1 as if there is an optimal solution (V_0, V_1), with $V_1 \neq 0$, we can replace it with $(V_0 + V_1, 0)$ which will either allow us to improve the liters or keep them the same. We are now free to assume $A_0 < A_1$ and $B_0 > B_1$.

Second, since we only have 2 drinks to mix, their per-liter average will lie somewhere on the line (P_0, P_1). Notice that the optimal mixing will consist of either taking only one of the drinks or having a final mix with exactly M_A grams of sugar and M_B grams of acids.

In conclusion, in the case of $N = 2$ we either have to consider the 2 points(when taken alone) or the point which lies on the ray $(0, 0) \to (M_A, M_B)$ on the line $(A_0, B_0) - (A_1, B_1)$, which can be done by ray-line intersection.

Solution $O(QN^2)$

When the interval (L_A, R_A) contains more than 2 points, we can prove the following claim – the optimal solution consists of only taking one drink type or it consists of mixing exactly 2 different drink types. Assume we pick more than 2 drink types to mix – there is some optimal $V_0, V_1, V_2, ..., V_{N-1}$. Again consider the concentration of sugar and acids per liter of the final drink $\bar{V}_0, \bar{V}_1, \bar{V}_2, ..., \bar{V}_{N-1}$, one can view this geometrically as taking a point in the convex hull formed by the set of points $\{P_0, P_1, P_2, ..., P_{N-1}\}$. Thus if we take the intersection of the ray going from $(0, 0)$ to $P_{\text{sum}} = \left(\sum_{i=0}^{N-1} \bar{V}_i A_i, \sum_{i=0}^{N-1} \bar{V}_i B_i\right)$ and the convex hull – we can find another point(achieved by mixing only 2 drink types) with lower concentrations of sugar and acids per liter, which can be then used to improve this original solution $\Rightarrow$ contradiction.

This gives the following $O(N^2)$ algorithm for a single query:

- Consider each point and find the maximum liters of it we can mix.
- Consider each pair of points and find the maximum liters we can mix with exactly M_A grams of sugar and M_B grams of acids.
- Take the maximum of those $O(N^2)$ pairs and $O(N)$ single candidates.

Solution $O(QN)$

Optimizing the above algorithm comes down to the observation that the pairs we need to consider are all consecutive points on the lower-left convex hull(LLCH) of the set of points $\{P_0, P_1, P_2, ..., P_{N-1}\}$. Since we already have the pairs sorted, we can find this LLCH in $O(N)$ time in total.

Solution $O(Q \log(N))$ for $(L_A, R_A) = (0, 10^9)$.

We can further optimize the solution for the case of a static set of points by noticing that mixing exactly M_A sugars and M_B acids is not possible for all line segments of the convex hull which do not cross the ray $(0, 0)$ to (M_A, M_B), so there is only one "interesting" line segment in the LLCH. We can find this line by keeping this LLCH in order of increasing A_i and binary searching for the first point that lies on the right side of the ray. The check can be done exclusively using integer arithmetic by calculating cross-products.

The other part of the candidates consists of taking only a single drink type, but this also means taking one of the constraints to be tight and we can thus consider only the point with minimum A_i or minimum B_i, which also happen to be the first and last points in the LLCH.

The final number of candidates we need to consider comes out to $3 = O(1)$ and we have to search for the line in $O(\log(N))$ time.

Solution $O\left(Q\sqrt{N}\right)$ with $O\left(N\sqrt{N}\right)$ precompute

One idea for solving this problem for full points comes from the $O(\log(N))$ solution for fixed set of points – we do not need to consider a lot of points to find the optimal solution.

Let us split the points in buckets of size $K = \left\lfloor \sqrt{N} \right\rfloor$. The i-th bucket S will contain the set of points $\left\{P_{iK}, P_{iK+1}, ..., P_{\min((i+1)K-1, N-1)}\right\}$. Having these buckets we will solve a query (L_A, R_A) by first finding the interval of indices $[l, r]$, s.t. all A_i, for $i \in [l, r]$ are in $[L_A, R_A]$ and $[s, e]$, s.t. all buckets with indices $\in [s, e]$ are fully contained in $[l, r]$.

First note that if $r - l + 1 \leq 2\sqrt{N}$ we can build the whole LLCH in $O\left(\sqrt{N}\right)$ and use the linear solution. In the case the points are too many to build the whole hull, we will consider the following 5 cases:

- Using the point with minimum A_i – this is the point P_l.
- Using the point with minimum B_i – for each bucket we have precomputed the minimum B_i point in it, we consider those $O\left(\sqrt{N}\right)$ points and the minimum B_i point in the "leftovers" in the 2 ends of the query not fully covered by some bucket.
- Using a pair of points i, j with $l \leq i < sK$ and $eK \leq j \leq r$ – we can solve this case by inserting all $O\left(\sqrt{N}\right)$ points on the 2 ends in a LLCH. and finding the best pair there in $O(\log(N))$.
- Using a pair of points i, j with $sK \leq i, j \leq r$ – this is the hardest case and will be discussed how to be solved in $O\left(\sqrt{N}\right)$ at the end.
- Using a pair of points i, j with $l \leq i, j < eK$ – equivalent to the previous case and will be discussed how to be solved in $O\left(\sqrt{N}\right)$ at the end.

We can thus compute the maximum across each of those cases, resulting in a $O\left(\sqrt{N}\right)$ time per query.

To solve the hardest case, namely $sK \leq i, j \leq r$ (the other follows by symmetry) we will precompute $\text{EmptyTime}_{b_0,b_1}$ for a pair of buckets b_0 and b_1 as follows – starting to fill a LLCH from $P_{b_0 K}$ and iteratively adding to a LLCH points $P_{b_0 K}, ..., P_{N-1}$ what the index of the point from bucket b_1 that leaves last the LLCH is and what the index of the point that removes it is. This allows us to keep a virtual version of the LLCH for the range $[sK, r]$ by giving at least one point per bucket or signalling that the bucket is empty at time r. We will use this information to find the first bucket b_r that has at least one point to the right of the ray from $(0, 0)$ to (M_A, M_B) and insert into a LLCH this bucket, the last bucket before it that is non-empty and the points in $[eK, r]$. This LLCH is built from $\leq 3K = O\left(\sqrt{N}\right)$ points and we can query as usual for $O(\log(N))$ time.

Illustration of idea for the hardest case.

- We have query $(l, r) = (2, 12)$
- $K = 3$ and the bucket cutoffs are dashed
- The full LLCH for (K, r) consists of P_3, P_4, P_9, P_{12}, but the virtual one consists only of P_3, P_9.