Problem 6.1 [Cache-oblivious Maximal Points in 3D].

Describe a cache-oblivious algorithm which takes $N$ distinct points in 3D space and returns a list of all maximal points. A point $(x, y, z)$ is maximal if there is no other point $(x', y', z')$ such that $x' \geq x$, $y' \geq y$, and $z' \geq z$; in other words, $(x, y, z)$ is not dominated by any other point. Your algorithm should run in $O(sort(N, M, B)) = O(NB \log_{M/B} N)$ memory transfers, under the tall-cache assumption.

Solution: We'll use distribution sweeping on the $x$ and $y$ coordinates with a lazy-funnelsort divide and conquer, similar to the batch orthogonal range-searching algorithm.

First, we can transform the points so all $x$-coordinates are distinct, and likewise for $y$- and $z$-coordinates: set $x' = x + \varepsilon_1 y + \varepsilon_2 z$ for very small $\varepsilon_1, \varepsilon_2 > 0$, which preserves all dominance relationships. Equivalently, we can set $x'$ is the tuple $(x, y, z)$ ordered lexicographically, $y'$ is the tuple $(y, z, x)$ ordered lexicographically, and $z'$ is the tuple $(z, y, x)$ ordered lexicographically, which also works equivalently. This makes our analysis easier, as we no longer have to deal with ties.

Now, sort all points by the $x$ coordinate. Then, do a modified mergesort on the $y$-coordinate using the lazy-funnelsort structure. At each stage of the merge, we merge two adjacent vertical strips, and output the only the maximal points of the merged strip, sorted by decreasing $y$. Note that keeping only maximal points is enough: domination is transitive, so any other point dominated by a discarded point would be dominated itself by a maximal point.

It remains to show the efficient streaming merge/filter step. Given two sequences $L$ and $R$ of points sorted in decreasing $y$, where all points in $L$ have smaller $x$-coordinate than all points in $R$, we want to discard all dominated points. Because the points of $R$ are maximal in $R$ and likewise with $L$, we’ll only ever discard points in $L$ which are dominated by a point in $R$. Thus, we sweep downwards by decreasing $y$, and keep a running maximum $z$-value from $R$. We output all values from $R$, and discard a value from $L$ if its $z$-value is less than the current maximum $z$ in $R$ (which must be from a point above and to the right, by construction), so we’re done.

Thus, we successfully merged two adjacent vertical strips and filtered for only maximal points, so by the lazy-funnelsort analysis, our algorithm is complete and takes $O(sort(N)) = O(NB \log_{M/B} N)$ memory transfers.