CS 164 & CS 266: Computational Geometry

Week 6

Lecture 6b: half-plane range search and fractional cascading

David Eppstein
University of California, Irvine

Spring Quarter, 2022
Half-plane range search
Warm-up: Range emptiness

Query: Is a given half-plane (the set above or below a line) empty?

Store upper and lower convex hull edges in arrays, sorted by slope.

For a half-plane above line, binary search for line’s slope in upper hull.
For half-plane below, search lower hull.

Edges with slopes closest to query slope are the endpoints of the extreme point in a direction perpendicular to the given line.

It is the deepest point in the range (if there is any point in the range) and the closest point to the range otherwise.
Listing all convex hull vertices in the range

Find extreme point

Walk left and right from it listing points in range until finding a point that is not in range (or looping back to start)

Time $O(\log n + k)$

But only finds hull vertices!
Nested hulls

Find hull of given points, remove its vertices, and repeat

Result: A sequence of nested convex polygons
Can be constructed in time $O(n \log n)$ (but complicated)
Using nested convex hulls for range reporting

To list all points in a query halfplane:

Start at the outermost hull
While we have not yet found a hull that is separate from the query range:
  ▶ Do a binary search to find the extreme point
  ▶ If it is inside the range, walk around the hull to find all other hull vertices in range, and move on to the next hull
  ▶ Otherwise, stop: this level has no results and none of the hulls nested inside it can have any results
Analysis of the nested binary search algorithm

Each binary search except for the last one finds a point to report
Each walk step except for the last in a walk finds a point to report
Worst case: one point in range in each hull, $k + 1$ binary searches
Time: $O(k \log n)$

Can we do better?
Insight: all binary searches use the same key (the slope)
Fractional cascading
What is fractional cascading?

Goal: Speed up related binary searches without too much space

A simplified version:

Data: \( k \) sorted lists of numbers \( S_0, S_1, \ldots, S_{k-1} \)

Total length: \( n = |S_0| + |S_1| + \cdots + |S_{k-1}| \)

No repeated values, even in different lists

Query: find the successors of a given number \( q \) in each list

\( s_i = \text{successor of } q \text{ in list } S_i \)
Example

Data:

- $S_0 = [0, 10, 20, 30, 40, 50, 60, 70]$
- $S_1 = [1, 2, 13, 25, 27, 51, 57]$
- $S_2 = [21, 22, 31, 32, 33, 41, 99]$
- $S_3 = [67, 68, 69]$

Total length $n = 8 + 7 + 7 + 3 = 25$

Query for $q = 24$ would find

$s_0 = 30$  $s_1 = 25$  $s_2 = 31$  $s_3 = 67$
Naïve solutions

Do the binary searches separately

Space = \( O(n) \) for storing each \( S_i \) as a sorted list
Query time = \( O(k \log n) \) for \( k \) binary searches

Merge into one list

For each value \( x \), store \( k \)-tuple of successors for queries that return \( x \) as their smallest value

\[
0: (0,1,21,67), 1: (10,1,21,67), 2: (10,2,21,67), 10: (10,13,21,67), 13: (20,13,21,67), 20: (20,25,21,67), 21: (30,25,21,67), \ldots
\]

Binary search in merged sorted array + look up \( k \)-tuple

Space \( O(kn) \), query time \( O(k + \log n) \)
Fractional cascading

Working backwards through the sequence of lists $S_i$, construct $T_i$: merged structure for $(S_i + \text{half the elements of } T_{i+1})$

Choosing the half of the elements that are in odd-numbered positions e.g. if $T = 1, 2, 3, 5, 7, 11, 20$ then $\frac{1}{2} T = 2, 5, 11$

So $T_i$ consists of:

- A sorted array of the merged items from $S_i + \frac{1}{2} T_{i+1}$
- A dictionary mapping each merged item $x$ to a pair $(a, b)$ where one of $a$ or $b$ is $x$, and the other one is the successor of $x$ in the other merged list
- When there is no successor in the other list, use $+\infty$
Example

- $S_3 = 67, 68, 69$  \( T_3 = S_3 \) (nothing to merge)  \( \text{Half elements: 68} \)
- $S_2 = 21, 22, 31, 32, 33, 41, 99$
- $T_2 = 21:(21,68), 22:(22,68), 31:(31,68), 32:(32,68), 33:(33,68), 41:(41,68), 68:(99,68), 99:(99, +\infty)$
- Half the elements of $T_2$: 22, 32, 41, 99
- $S_1 = 1, 2, 13, 25, 27, 51, 57$
- $T_1 = 1:(1,22), 2:(2,22), 13:(13,22), 22:(25,22), 25:(25,32), 27:(27,32), 32:(51,32), 41:(51,41), 51:(51,99), 57:(57,99), 99:(+\infty, 99)$
- Half the elements of $T_1$: 2, 22, 27, 41, 57
- $S_0 = 0, 10, 20, 30, 40, 50, 60, 70$
- $T_0 = 0:(0,2), 2:(10,2), 10:(10,22), 20:(20,22), 22:(30,22), 27:(30,27), 30:(30,41), 40:(40,41), 41:(50,41), 50:(50,57), 57:(60,57), 60:(60, +\infty), 70:(70, +\infty)$
Searching fractionally cascaded lists

To find the successors of $q$:

- Binary search for successor $t_0$ in merged list $T_0$
- Set $i = 0$
- Then, repeat:
  - Use dictionary for $T_i$ to find the pair $(a, b)$ where $a = s_i = $ successor in $S_i$ and $b$ is successor in $\frac{1}{2} T_{i+1}$
  - Output $s_i$
  - Let $c$ be the (skipped) element of $T_{i+1}$ just before $b$
  - If $q < c$ then $t_{i+1} = c$ else $t_{i+1} = b$
  - Set $i = i + 1$
Example (continued)

To search for the successor of $q = 24$:

- Binary search in $T_0$ finds successor $t_0$: $27:(30, 27)$
- Output $s_0 = 30$, successor in $S_0$
- Successor in $T_1$ might be either 27 or previous item, 25
- Because $q < 25$, successor in $T_1$ is $25:(25, 32)$
- Output $s_1 = 25$, successor in $S_1$
- Successor in $T_2$ might be either 32 or previous item, 31
- Because $q < 31$, successor in $T_2$ is $31:(31, 68)$
- Output $s_2 = 31$, successor in $S_2$
- Successor in $T_3$ might be either 68 or previous item, 67
- Because $q < 67$, successor in $T_3$ is 67
- Output $s_3 = 67$, successor in $S_3$
Fractional cascading analysis

Query time

One binary search + $O(1)$ for each list after the first

Total $O(k + \log n)$

Space and set-up time

Each element of $S_i$ contributes 1 to the length of $T_i$, $\frac{1}{2}$ to the length of $T_{i-1}$, $\frac{1}{4}$ to the length of $T_{i-2}$, ... 

So the total space and total set-up time is $O(n)$

Best combination of time and space from naïve solutions
Half-plane range reporting, revisited

Structure:
- Nested hulls
- Sorted lists of slopes of upper and lower hull edges, ordered from outer hull to inner
- Fractional cascading

Query: One binary search to start fractional cascading, after which we can find extreme point in each level in constant time.

List vertices on each hull, starting from its extreme point, stopping when we reach a hull that is entirely outside the range.

Total $O(\log n + k)$