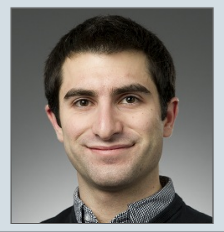
MADALGO CENTER FOR MASSIVE DATA ALGORITHMICS



Bryan T. Wilkinson
Aarhus University

Danmarks Grundforskningsfond
Danish National Research Foundation

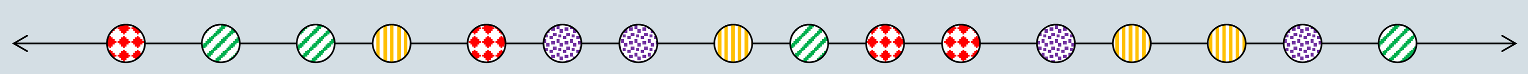
Range Searching in Query-Dependent Categories

Problem

We consider a very natural range searching variant with categorical data in the pointer machine model.

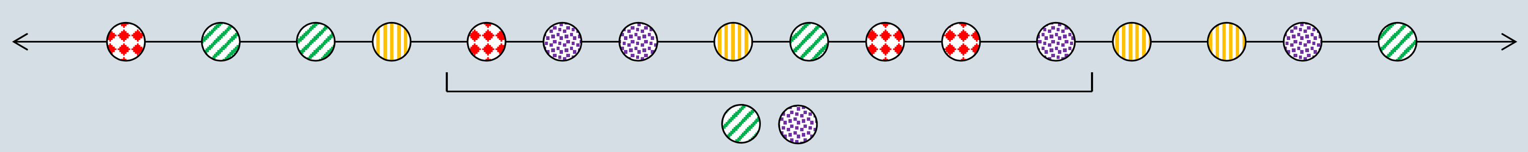
Data structure input:

- n coloured points on the x-axis
- colours taken from $\mathcal{C} = \{1, \dots, t\}$



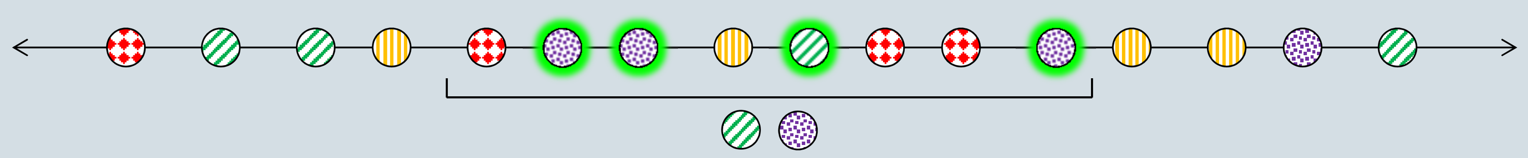
Query input:

- an x-range $[a, b]$
- a set of colours $C \subseteq \mathcal{C}$, such that $|C| = r$



Query output:

- all k points in $[a, b]$ with a colour in C



Motivation

Our problem is motivated by the fact that databases often contain both ordinal (numerical) and nominal (categorical) data.

Consider a company with many employees. Each employee has a salary, which is an integer, and belongs to a department, which is one of marketing, engineering, finance, human resources, etc… Consider the following mapping:

$$\begin{aligned} \text{employee} &\rightarrow \text{point} \\ \text{salary} &\rightarrow x\text{−value} \\ \text{department} &\rightarrow \text{colour} \end{aligned}$$

Now, our data structure can answer natural questions such as:

> *"Which employees in marketing or finance have salaries between 40.000 kr and 50.000 kr ?"*

Previous Results

All previous categorical range searching data structures support a different type of query:

> *"Which colours are represented by at least one point within this range?"*

For example, in one dimension, Janardan and Lopez [1] give a solution in optimal $O(n)$ space and optimal $O(\log n + k)$ query time, where k is the number of colours reported.

Standard range searching is a special case of our problem that arises when there is only one colour. The binary tree solves this special case in optimal $O(n)$ space and optimal $O(\log n + k)$ query time, where k is the number of points reported.

New Results

We give a pointer machine data structure that requires:

- $O(n)$ space
- $O\left(\log n + r \log \frac{t}{r} + k\right)$ query time if the query colours are sorted
- $O(\log n + r \log t + k)$ query time if the query colours are unsorted

We also give matching lower bounds for both query variants, showing that our bounds are optimal for all parameters: n, r, t, and k.

Future Work

As we are tackling a new problem that is quite simple to describe, there are many interesting variants to consider:

- dynamic point set: support insertions and deletions of points
- higher dimensional points: e.g., 2-D points and orthogonal ranges
- multiple categorical attributes: more complex query language

References

[1] R. Janardan, M. Lopez. *Generalized intersection searching problems.* International Journal of Computational Geometry & Applications, 2013.

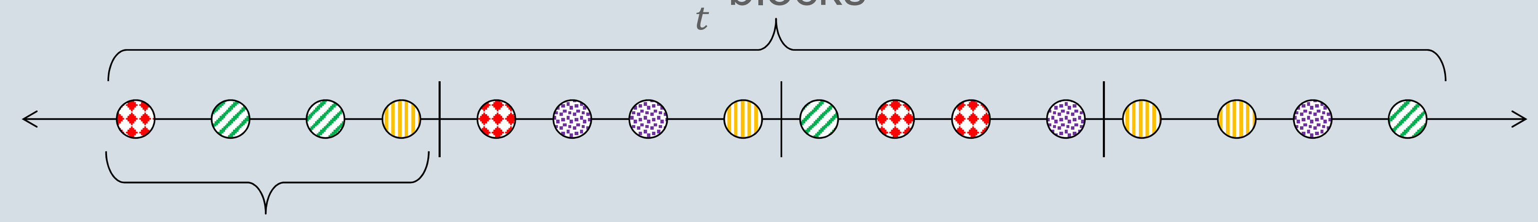
Upper Bound

It is sufficient to find the predecessor of b for each query colour, as we can then find the other points by iterating through linked lists built for each colour in $O(r + k)$ time.

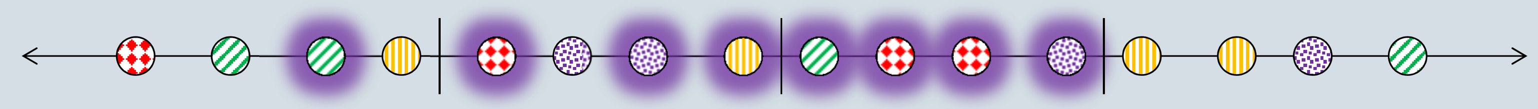
Strategy 1: Separate binary searches.

We simply perform predecessor search at b in separate point sets for each query colour. This requires $O(n)$ space and $O\left(\sum_{i=1}^{r} \log n_i\right) = O\left(r \log \frac{n}{r}\right)$ query time.

Strategy 2: Divide into blocks.

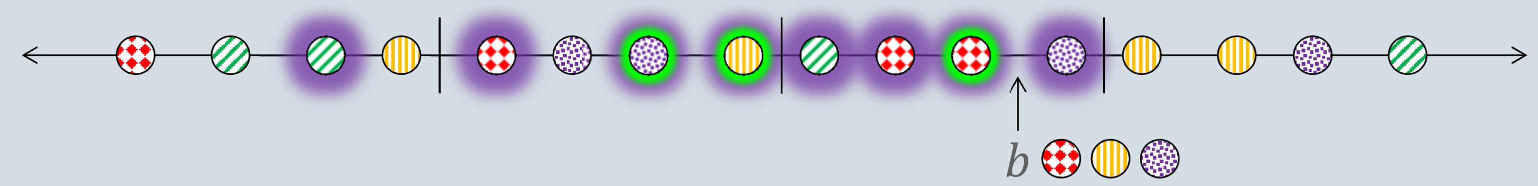


For each block, we store in the data structure of Strategy 1 all points in the block as well as predecessors for all t colours. For example, we store the following points in the data structure for block 3:



Each such data structure requires $O(t)$ space, for a total of $O(n)$ space.

Given a query, we find the block containing b via binary search in $O(\log n)$ time and then we forward the query on to the Strategy 1 data structure for this block, which requires $O\left(r \log \frac{t}{r}\right)$ time.



Lower Bound

We first reduce the problem of searching for r keys in a set of size t to our original problem. Any pointer machine data structure for the former problem must be able to reach at least $\binom{t}{r}$ different combinations of r nodes during a query, since there are $\binom{t}{r}$ different queries which must each locate a different combination of r keys in the data structure. However, we show that if the data structure's query time is $o\left(r \log \frac{t}{r}\right)$, then it cannot reach $\binom{t}{r}$ different combinations of r nodes.