

# I/O-Efficient Data Structures for **Colored** Range and Prefix Reporting

Kasper **Green** Larsen, MADALGO, Aarhus University  
Rasmus Pagh, IT University of Copenhagen

Presenter: Yakov Nekrich

# Motivating example

Sanningen finns  
på marken /men  
ingen vågar ta  
den. /Sanningen  
ligger på gatan. /  
Ingen gör den till  
sin.

Jag ligger och ska  
somna, jag ser  
okända bilder /  
och tecken  
klottrande sig  
själva bakom  
ögonlocken / på  
mörkrets vägg. I  
springan mellan  
vakenhet och  
dröm / försöker  
ett stort brev  
tränga sig in  
förgäves.

Jag öppnar dörr  
nummer två. /  
Vänner! Ni drack  
mörkret / och blev  
synliga.

- Store collection of documents (= sets of words).
- Given a string  $p$ , return the IDs of all documents that contain  $p$  as a word.
- Optimal solution: “Inverted index”, precomputing the answer for each  $p$ .

# Motivating example

Sanningen finns  
på marken / men  
ingen vågar ta  
den. / Sanningen  
ligger på gatan. /  
Ingen gör den till  
sin.

Jag ligger och ska  
somna, jag ser  
okända bilder /  
och tecken  
klottrande sig  
själva bakom  
ögonlocken / på  
mörkrets vägg. I  
springan mellan  
vakenhet och  
dröm / försöker  
ett stort brev  
tränga sig in  
förgäves.

Jag öppnar dörr  
nummer två. /  
Vänner! Ni drack  
mörkret / och blev  
synliga.

- Store collection of documents (= sets of words).
- Given a string  $p$ , return the IDs of all documents that contain  $p$  as *prefix of* a word.
- Optimal solution: Topic of this talk.

“colored prefix reporting”

# Simple data structures

Sanningen finns  
på marken / men  
ingen vågar ta  
den. / Sanningen  
ligger på gatan. /  
Ingen gör den till  
sin.

Jag ligger och ska  
somna, jag ser  
okända bilder /  
och tecken  
klottrande sig  
själva bakom  
ögonlocken / på  
mörkrets vägg. I  
springan mellan  
vakenhet och  
dröm / försöker  
ett stort brev  
tränga sig in  
förgäves.

Jag öppnar dörr  
nummer två. /  
Vänner! Ni drack  
mörkret / och blev  
synliga.

- **Inverted list for each prefix:**  
Too much space for fixed alphabet size.
- **Inverted list for each word:**  
Too much time if many documents have many different words with prefix  $p$ .

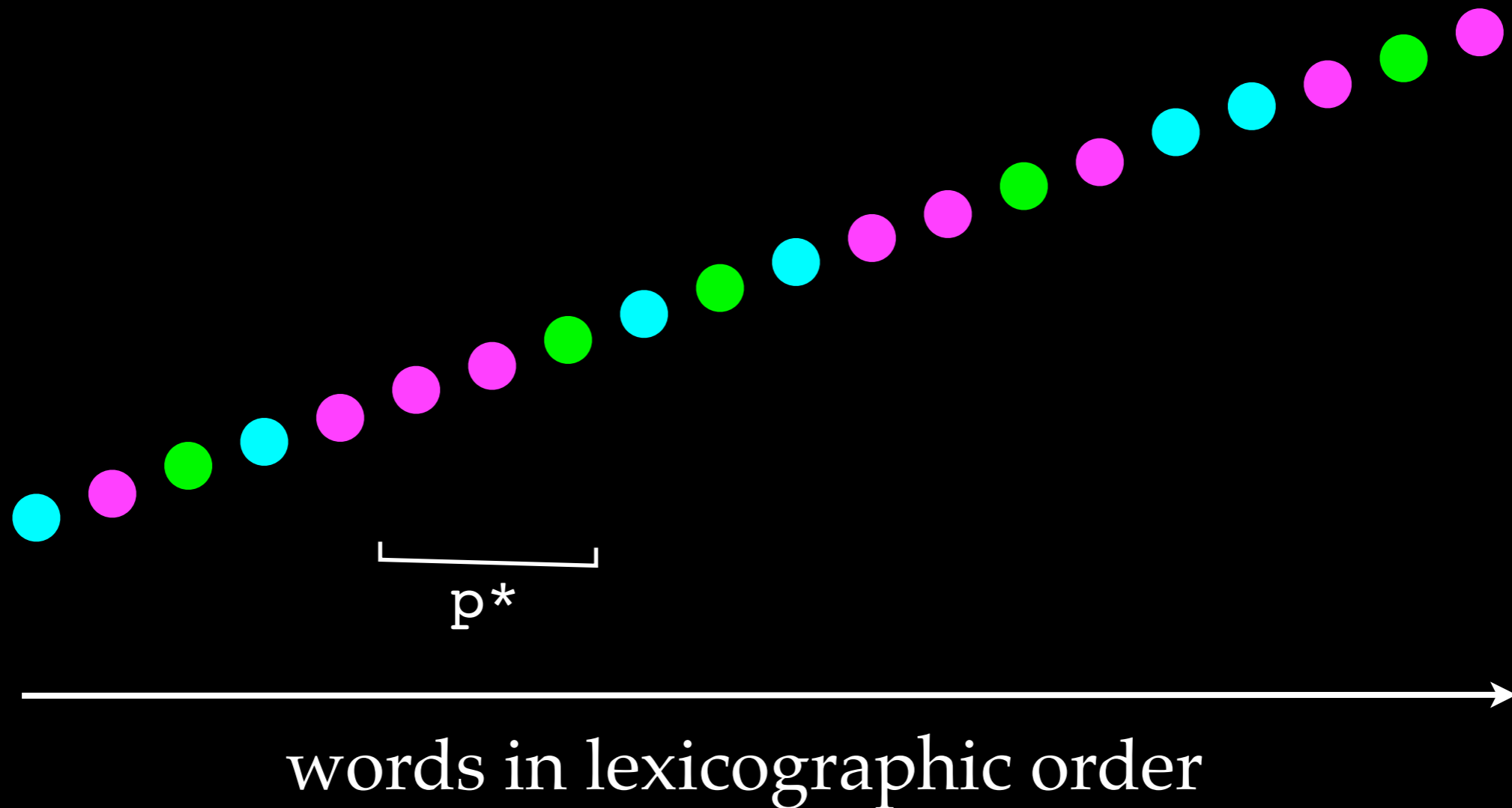
# Relation to range reporting

[GHJS '03]



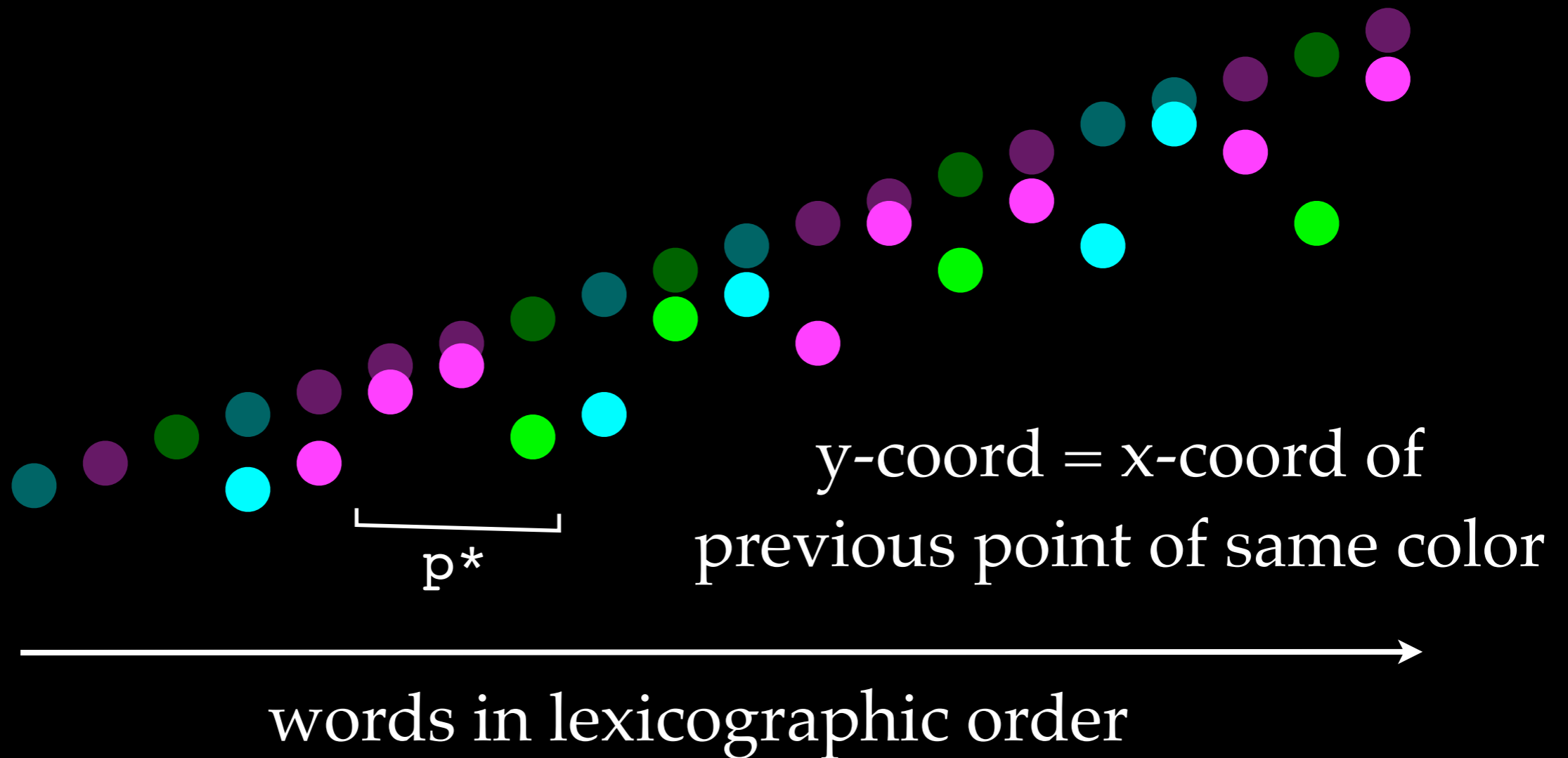
# Relation to range reporting

[GHJS '03]



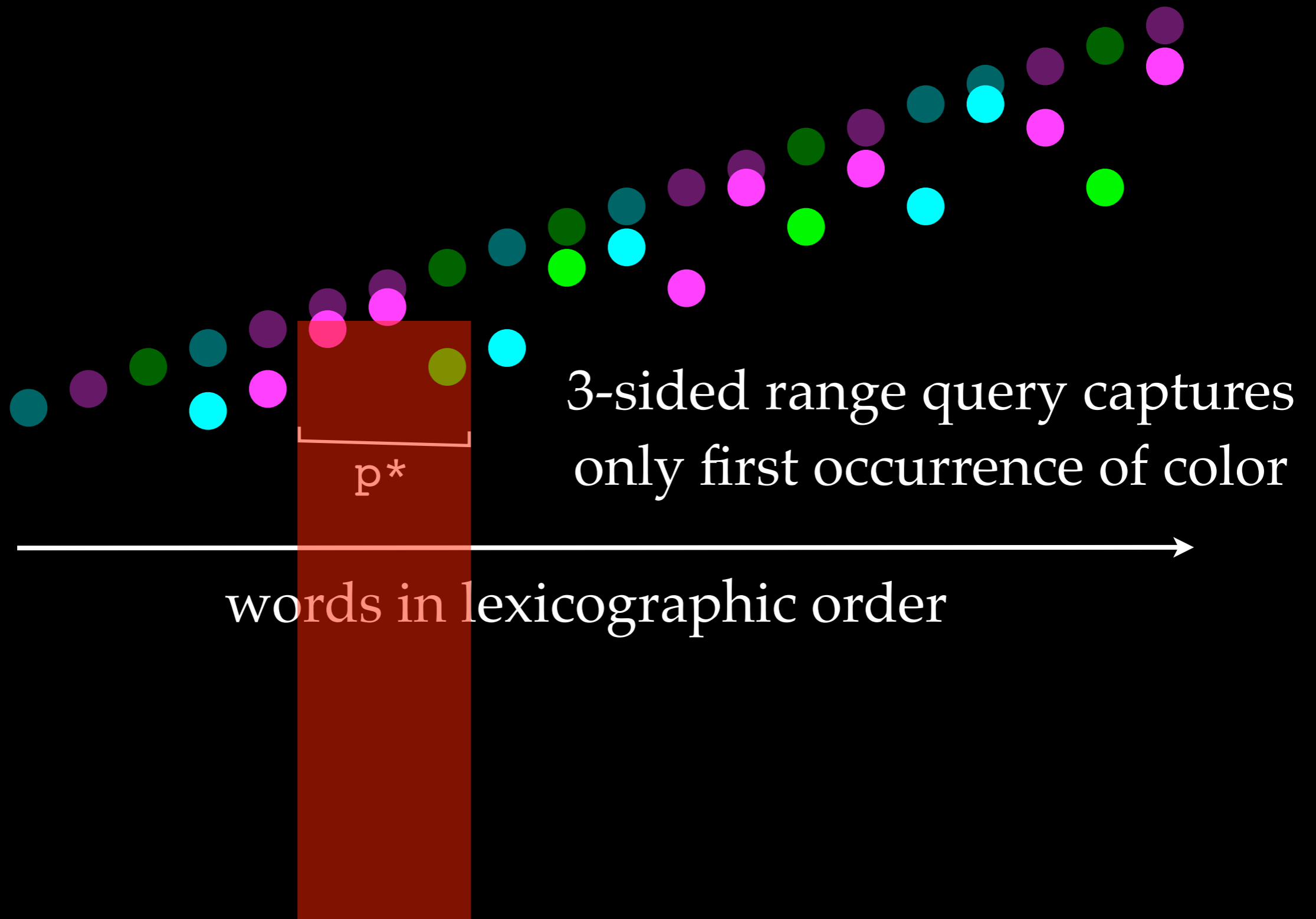
# Relation to range reporting

[GHJS '03]



# Relation to range reporting

[GHJS '03]





# New result

We can store a subset of  $n$  points from  $[n] \times [n]$ , using linear space, such that 3-sided range queries can be answered in  $O(1+k/B)$  I/Os.

- Parameters:
  - $k$  = number of points returned
  - $B$  = number of points in a memory block  
(assume query string fits in one block)

# New result

Optimal time  
and space

We can store a subset of  $n$  points from  $[n] \times [n]$ , using linear space, such that 3-sided range queries can be answered in  $O(1+k/B)$  I/Os.

- Parameters:

$k$  = number of points returned

$B$  = number of points in a memory block  
(assume query string fits in one block)

# New result

Optimal time  
and space

We can store a subset of  $n$  points from  $[n] \times [n]$ , using linear space, such that 3-sided range queries can be answered in  $O(1+k/B)$  I/Os.

- Parameters:

$k$  = number of points returned

$B$  = number of points in a memory block  
(assume query string fits in one block)

Implies optimal  
colored prefix  
reporting

# Model of computation

- I/O model with  $\Theta(B \log n)$  bits per block.
- Memory is a sequence of blocks.  
Cost model: Number of block retrievals (I/Os).
- $O(1)$  blocks can be stored in “cache” and accessed at no cost.

# Model of computation

Many other papers:  
Block stores  $B$  "items".

- I/O model with  $\Theta(B \log n)$  bits per block.
- Memory is a sequence of blocks.  
Cost model: Number of block retrievals (I/Os).
- $O(1)$  blocks can be stored in "cache" and accessed at no cost.

# Selected previous results

| Reference       | Space overhead  | Search time               | Model        |
|-----------------|-----------------|---------------------------|--------------|
| Arge et al. '99 | $O(1)$          | $O(\log_B(n))$            | Comparison   |
| Nekrich '07     | $O(1)$          | $\log_B \log_B \dots (n)$ | Unrestricted |
| Nekrich '07     | $\log_B^*(n)$   | $O(\log_B^*(n))$          | Unrestricted |
| Nekrich '07     | $(\log_B(n))^2$ | $O(1)$                    | Unrestricted |
| Here            | $O(1)$          | $O(1)$                    | Unrestricted |

data structures for 3-sided range reporting

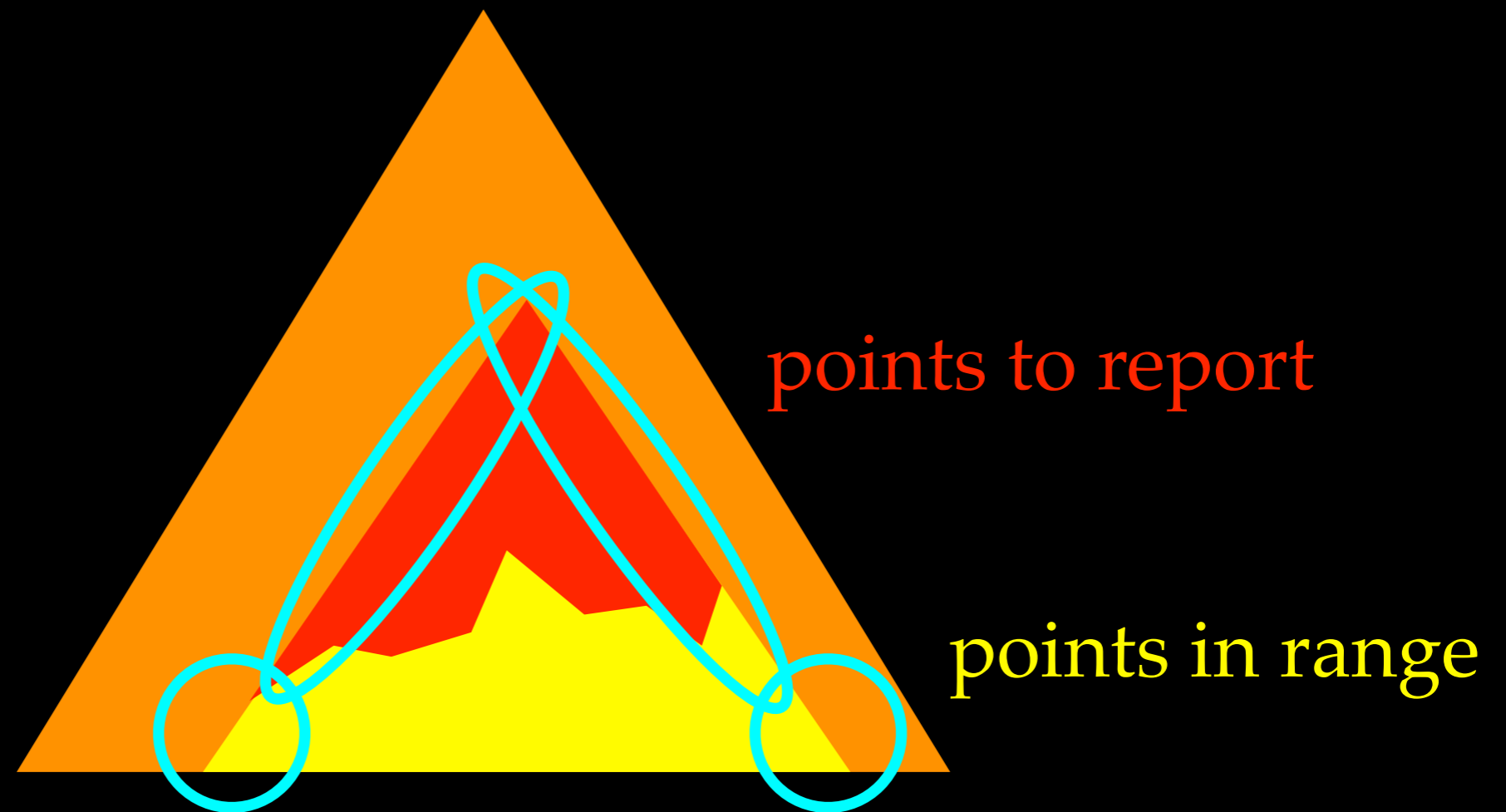
# High-level description

1. Place points in a binary priority search tree



# High-level description

2. Search for points near the “fringe” using  $O(1)$  searches in smaller “base” data structures.





# High-level description

3. Read blocks containing remaining points (easy).

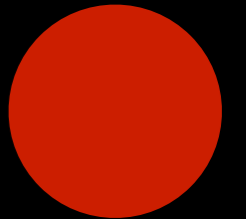


# Base data structure

- Core technical contribution of paper.
- Able to handle point sets of size  $\text{poly}(B \log n)$  optimally.
- **Main technique:** Use of tabulation and fusion trees allow us to make a dynamic data structure partially persistent *free of charge* when the number of updates to it is small.
- Refer to paper for more details.

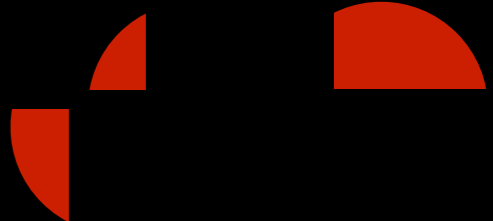
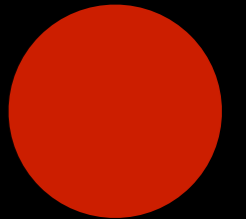
# Indivisibility assumption

- Assumption often used to show lower bounds:  
Each block contains at most  $B$  points/items;  
reading a block is required to report an item.
- Our data structure is among few to break this assumption (with the dictionary of Iacono and Pătraşcu, previous presentation).



# Indivisibility assumption

- Assumption often used to show lower bounds: Each block contains at most  $B$  points/items; reading a block is required to report an item.
- Our data structure is among few to break this assumption (with the dictionary of Iacono and Pătraşcu, previous presentation).
- **Open problem:** Is there a nontrivial lower bound under the indivisibility assumption?



# Memory models

- There may be alternatives to the cache-oriented I/O model:

*Unlike conventional processors that rely on the hardware to automatically bring data and instructions close to the processor with a hierarchy of hardware caches, the Cell Broadband Engine requires the programmer to create a “shopping list” of the data that the program requires.*

from [ibm.com](http://ibm.com)

# Scatter-I/O model

- One I/O operation can read or write *any* set of  $B$  words in memory.
- In this stronger model, we give a much simpler data structure for colored prefix search.

# Scatter-I/O model

- One I/O operation can read or write *any* set of  $B$  words in memory.
- In this stronger model, we give a much simpler data structure for colored prefix search.
- **Open problem:** Can notoriously I/O-difficult graph problems such as BFS and DFS be efficiently solved in this model?

# Conclusion

- Theoretically optimal solutions in the I/O model for **colored** prefix / range reporting.
- In fact, optimal solution to 3-sided range reporting in two dimensions.



# Conclusion

- Theoretically optimal solutions in the I/O model for **colored** prefix / range reporting.
- In fact, optimal solution to 3-sided range reporting in two dimensions.
- **Main open problem:** Efficient extension to top- $k$  searches, where only the  $k$  highest ranked colors are reported.