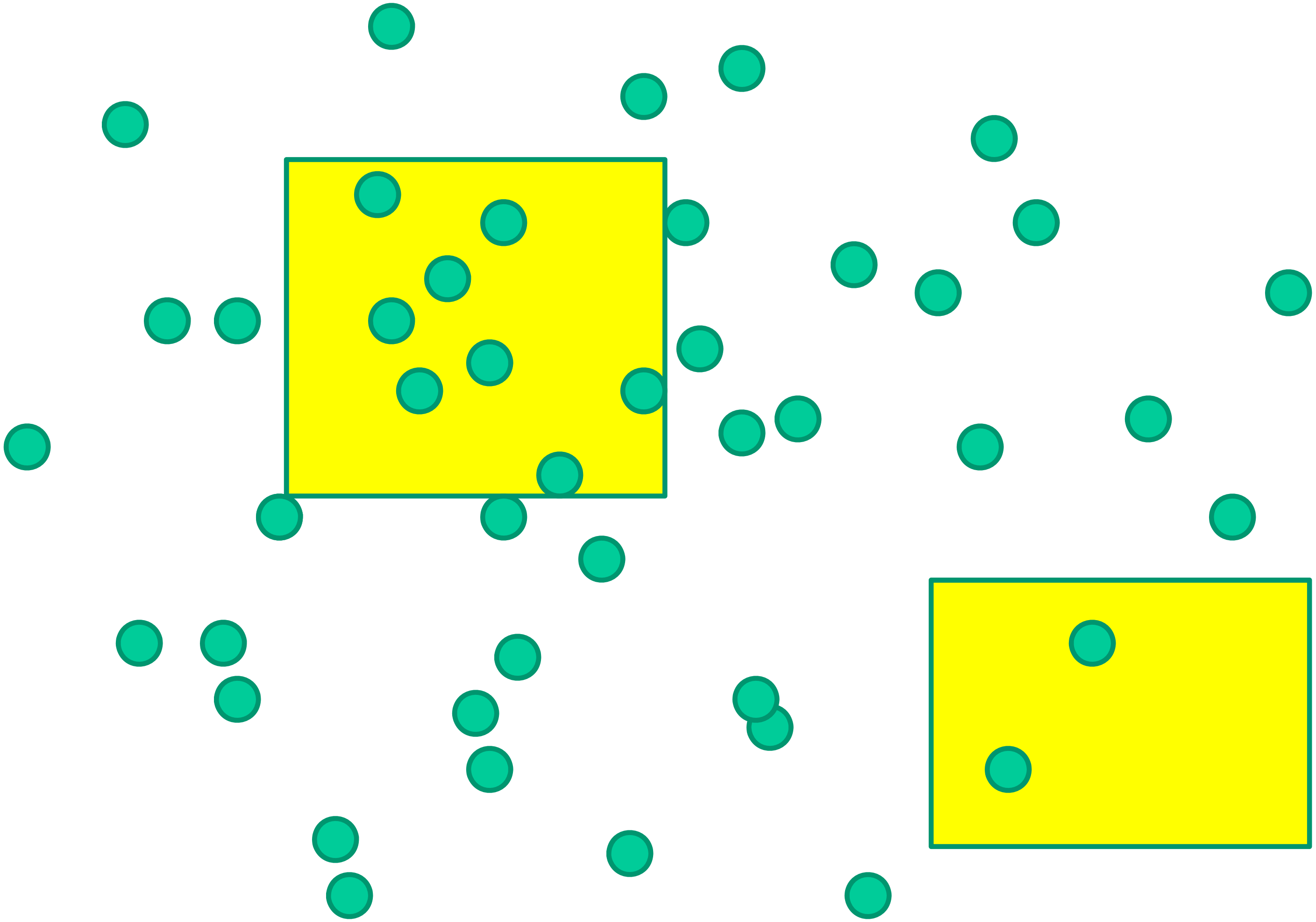
Range searching

- Given a faimly F of regions and a set S of n points in the d-dimensional space, construct a data structure D(S), such that we can do the following query efficiently.
- Reporting range query: Given any region R in F, report the set of points of S in R.
- Counting range query: answer the number of points of S in R

Rectangular range searching(d=2)

- Given a set S of n points in the plane, construct a data structure D(S), such that we can do the following query efficiently.
- Rectangle range query (counting): Given an axis parallel rectangle R, report the set of points of S in R.
- Answer the number of points of S in R efficiently

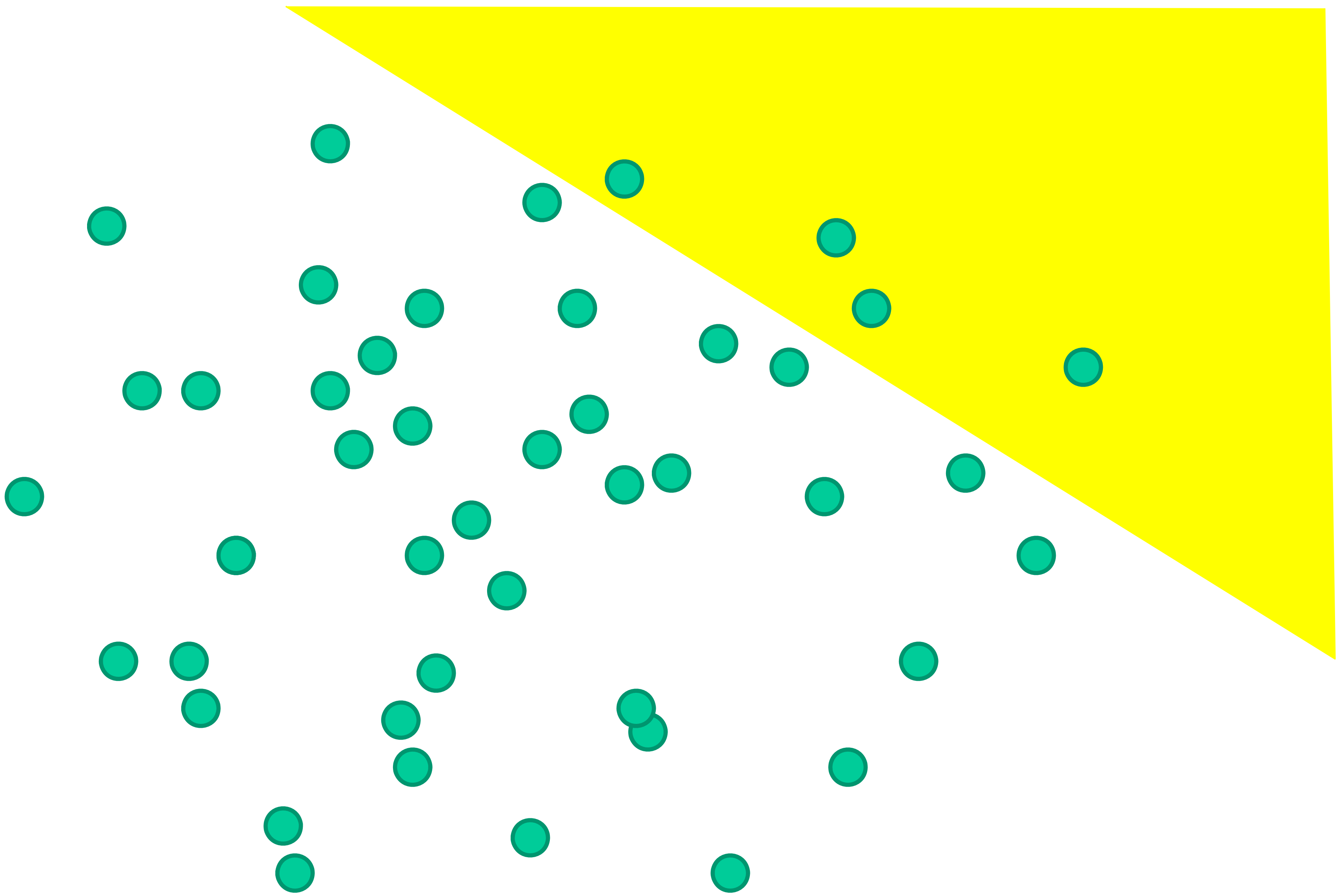


Example of application

- Given a database of customers with (income, sales), report the set of customers such that
  - $2M < income < 2.5M$
  - $100K < sales < 300K$

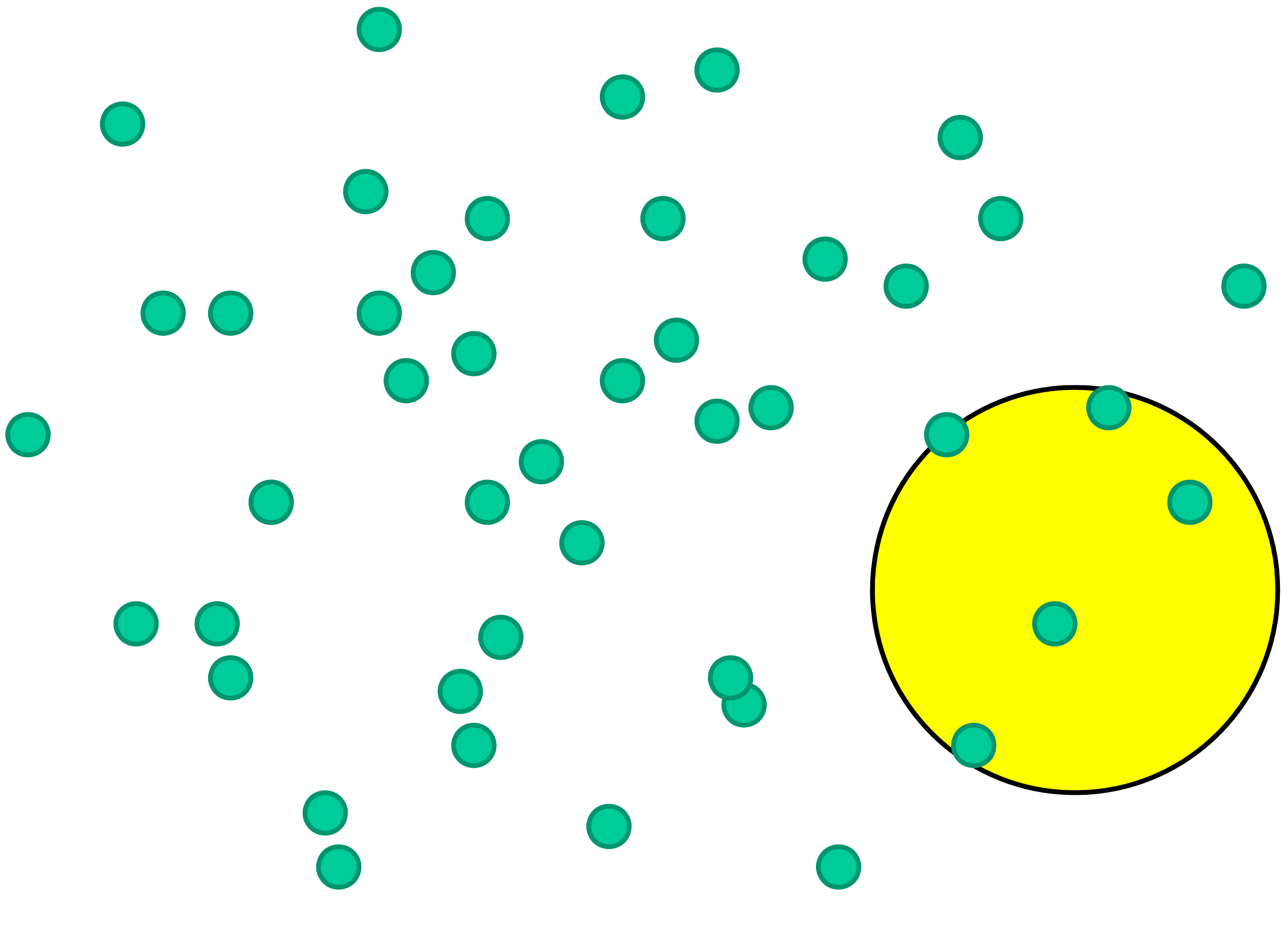
Halfplane range searching(d=2)

- Given a set S of n points in the plane, construct a data structure D(S), such that we can do the following query efficiently.
- Halfplane range query: Given a halfplane H, report the set of points of S in H efficiently
- Answer the number of points of S in H.



Circle range searching(d=2)

- Given a set S of n points in the plane, construct a data structure D(S), such that we can do the following query efficiently.
- Circle range query: Given a circle C, report the set of points of S inside C.
- Answer the number of points of S in C.

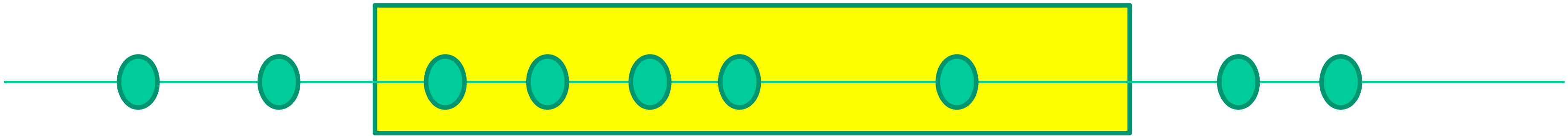


Example

- Answer the set of Italian restaurants within distance of 300 M from Sendai station.

Interval Range searching (d= 1)

- Given a set S of n data with real key values, construct a data structure D(S), such that we can do the following query efficiently.
- Interval range query (reporting): Given an interval I, report the set of data of S such that the key values are in I.
- Interval range query (counting): Answer the set of data of S whose key values are in I.



Theorem 1

There is an O(n) size data structure D(S) to answer the reporting interval range query in O(k+log n) time, where k is the number of reported elements. Also, the counting interval query can be compted in O(log n) time using O(n) size data structure.

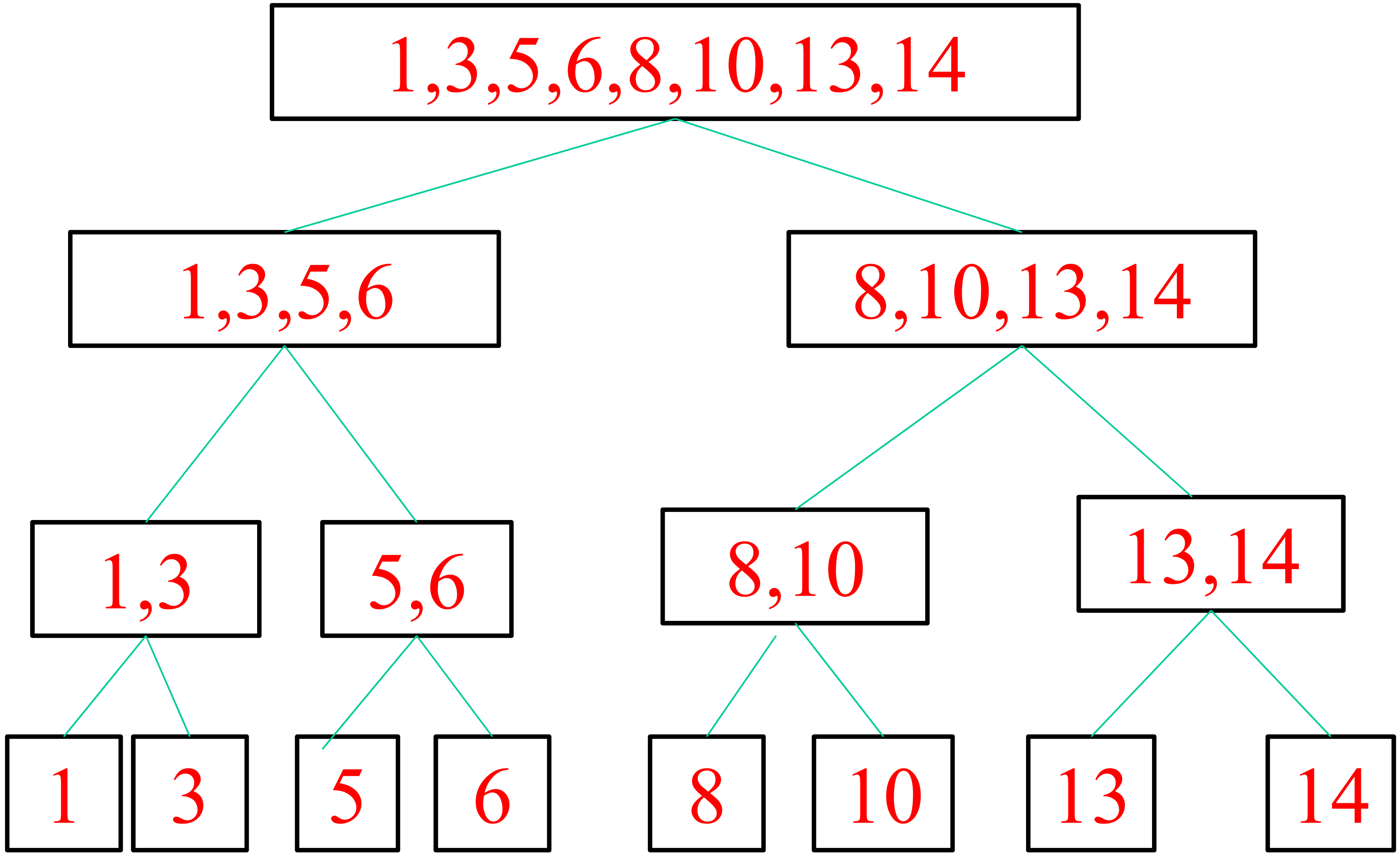
Interval Range searching (d= 1)

- Given a set S of n data with real key values key(x) and real data value data(x) for each x of S, construct a data structure D(S), such that we can do the following query efficiently.
- Range minimum query : Given an interval I, report x of S such that the key(x) values are in I and minimizing data(x).
  - Very important in data compression and data mining.

Interval tree

- Store n data into a binary tree $T(S)$
- The root of the tree contains S
- The left child is $T(S_1)$, where S_1 is the set of the n/2 data with small key values
- The right child is $T(S_2)$, where $S_2 = S - S_1$

A set stored at a vertex of the interval tree is called a primary set.


1,3,5,6,8,10,13,14
1,3,5,6
8,10,13,14
1,3
5,6
8,10
13,14
1
3
5
6
8
10
13
14

Interval query using T(S)

- Lemma 1.

For any interval I, $S \cap I$ is represented as a union of O(log n) primary sets.

- Lemma 2.

The primary sets considered in Theorem 1 can be computed in O(log n) time.

- Theorem 2. The range minimum query can be computed in O(log n) time

Rectangular range searching(d=2)

- Given a set S of n points in the plane, construct a data structure D(S), such that we can do the following query efficiently.
- Rectangle range query (counting): Given an axis parallel rectangle R, report the set of points of S in R.
- Answer the number of points of S in R efficiently

Rectangular range searching

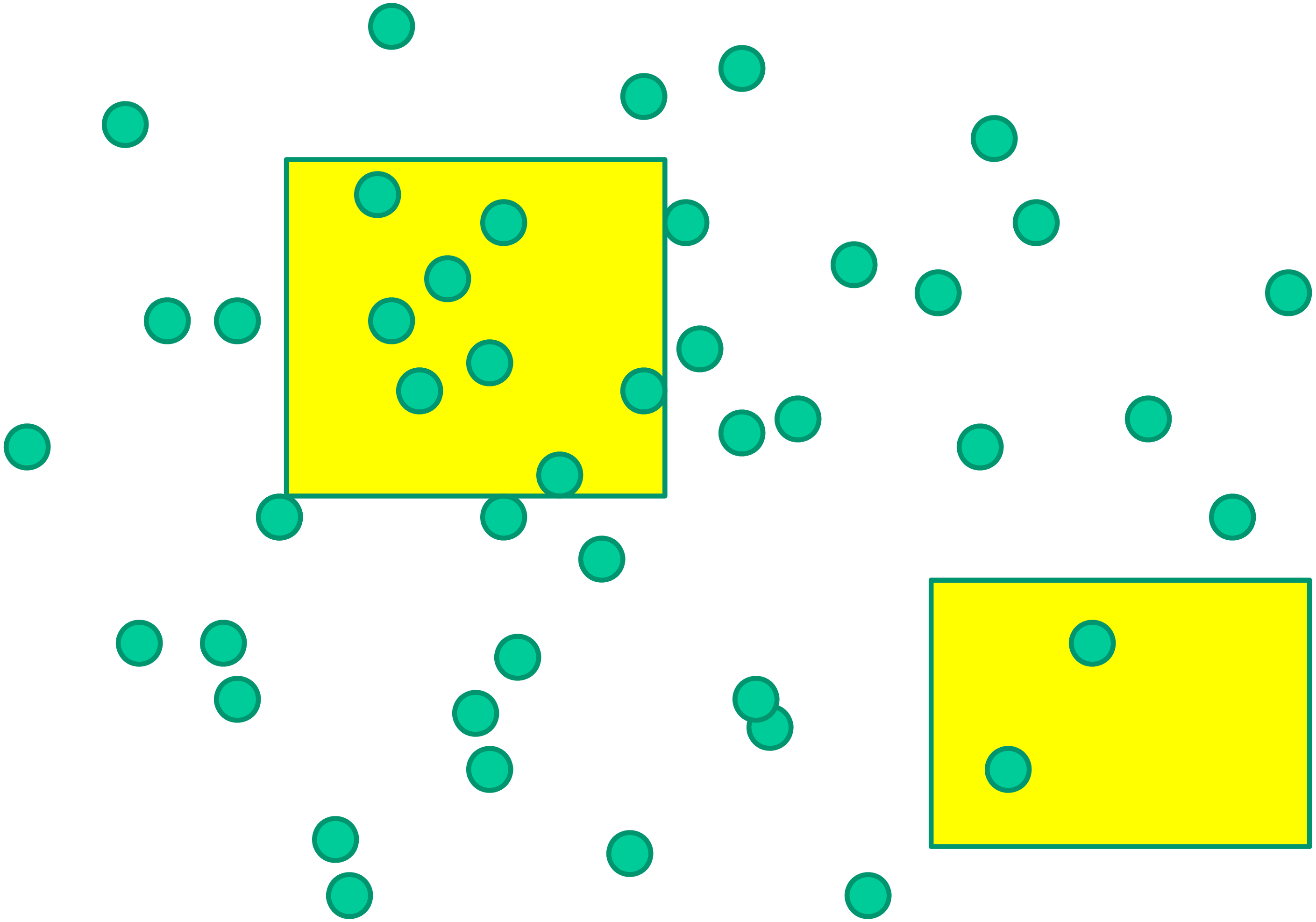
- Theorem 3

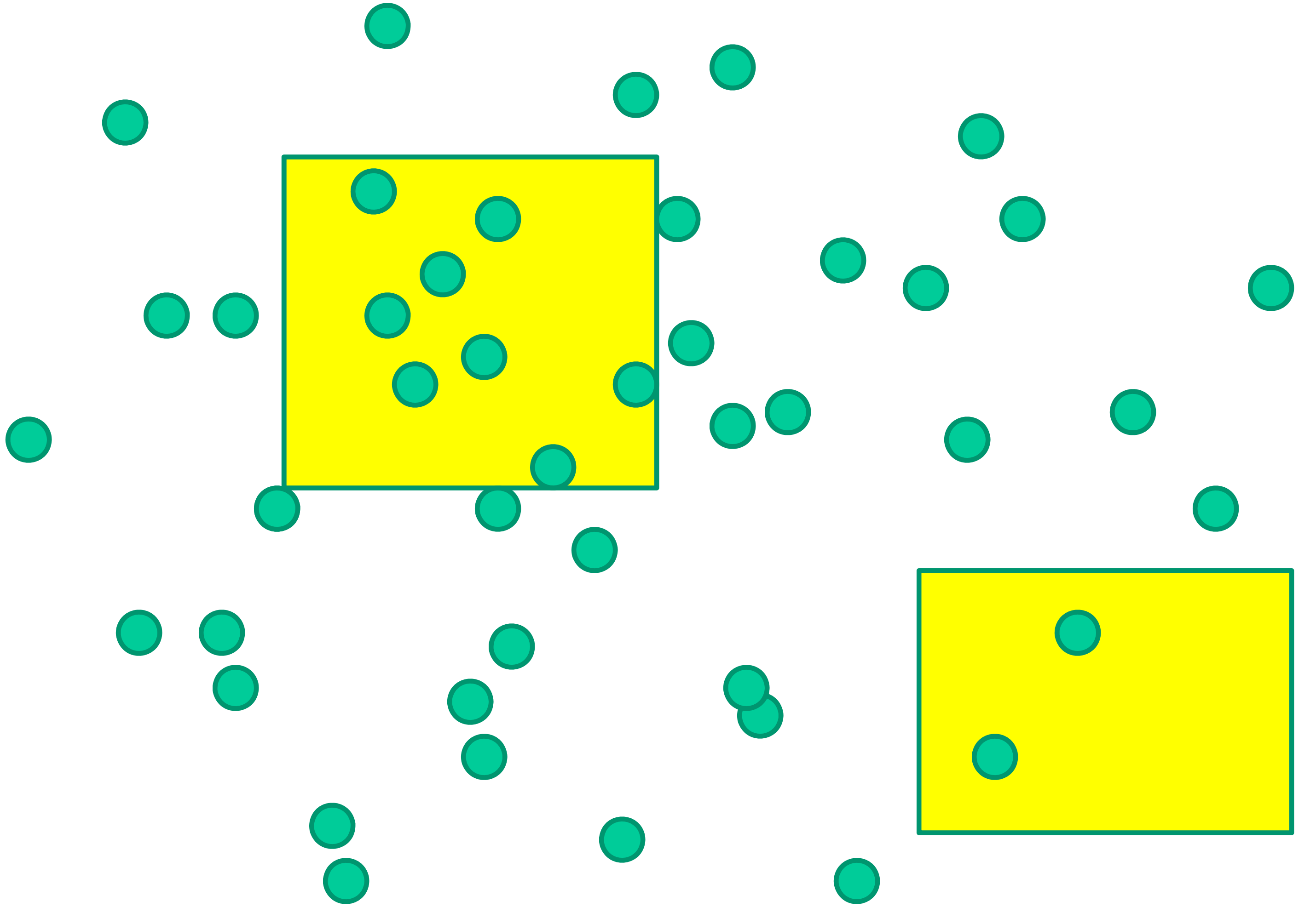
There is a data structure of size O(n log n) to answer the reporting range query in O(k + log n) time , where k is the number of reported elements.

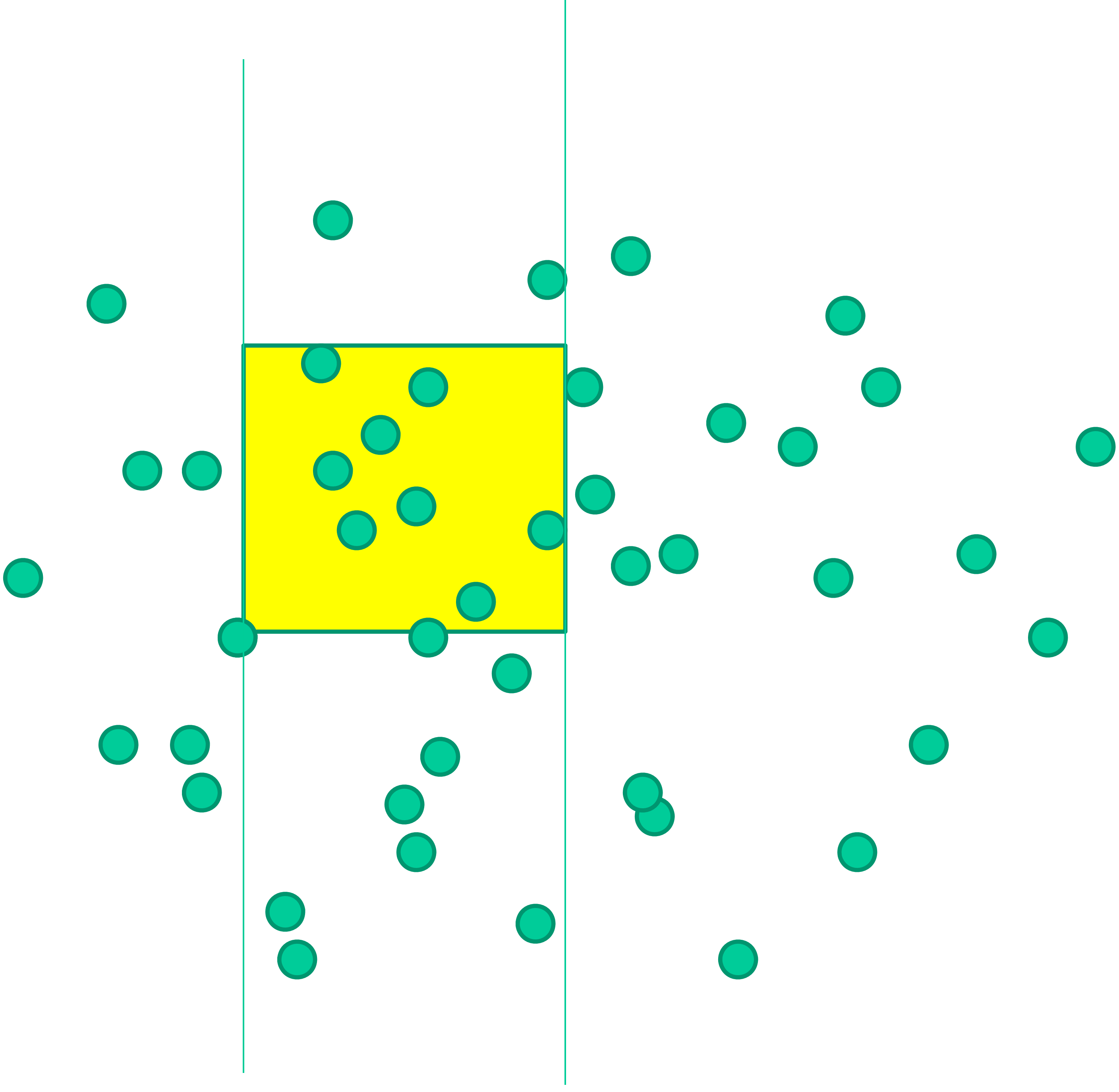
- Theorem 4.

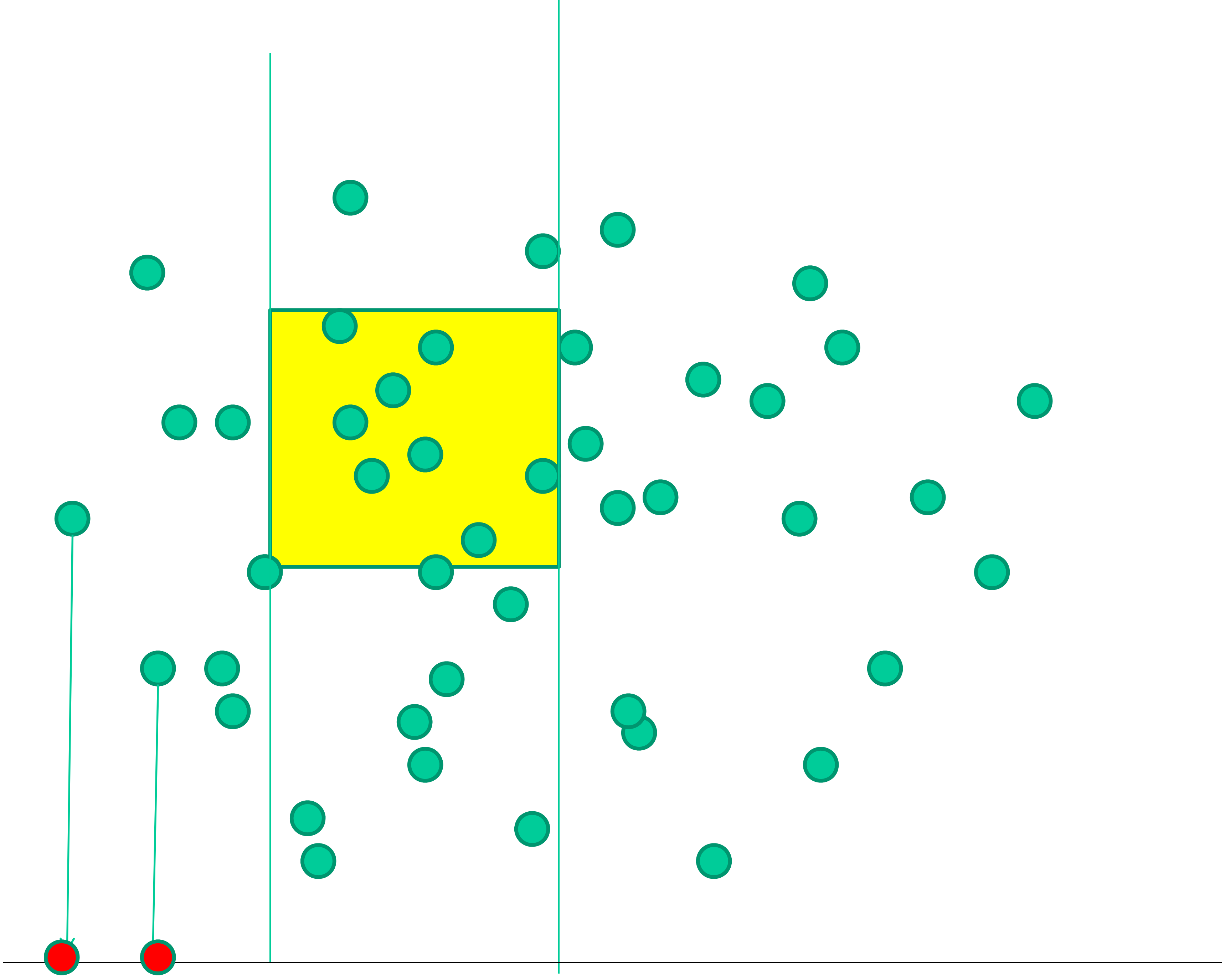
There is a data structure of size O(n log n) to answer the counting range query in O($\log^2 n$) time.

- Theorem 5. The counting range query can be done in O(log n) time using an O(n log n) size data structure.



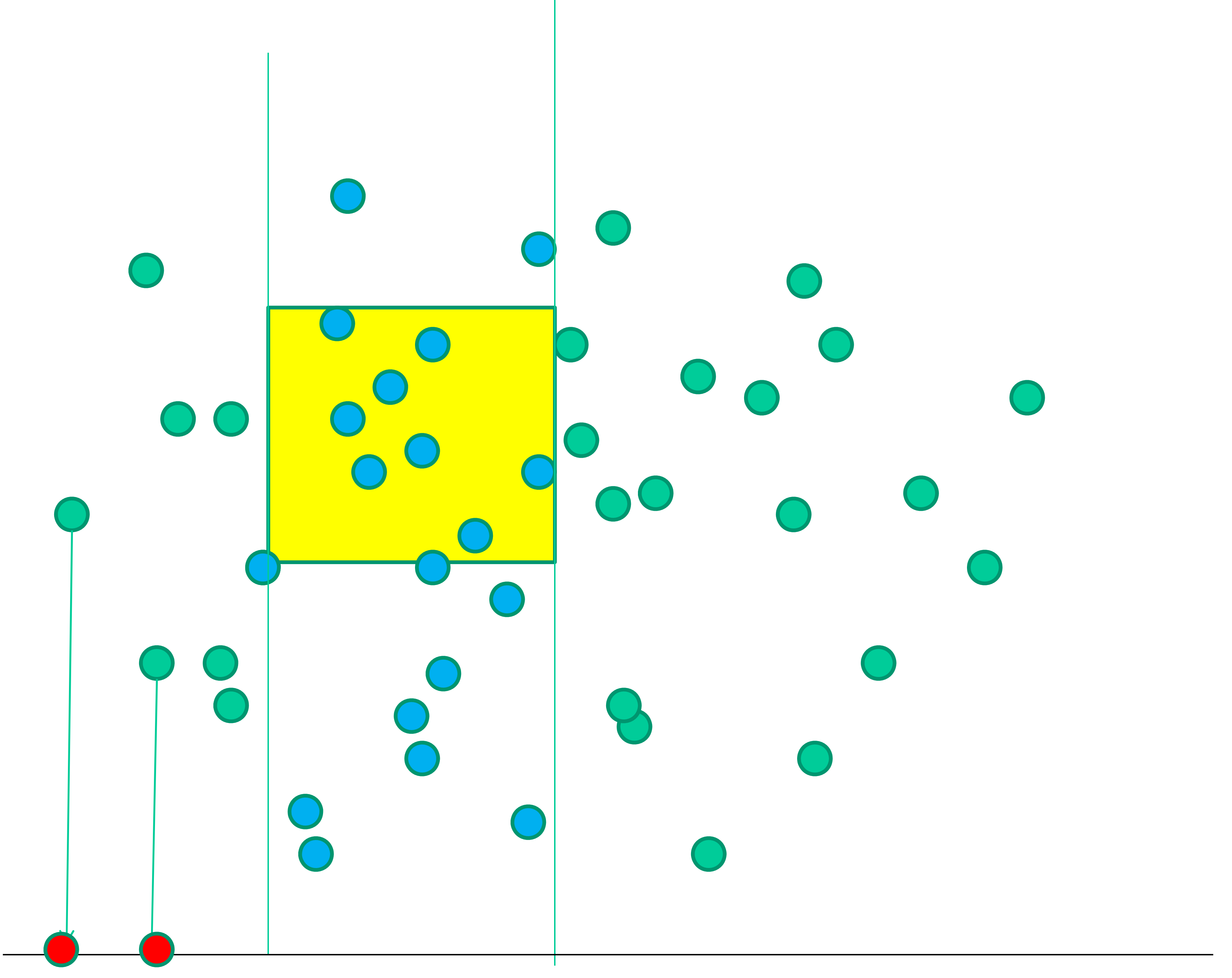


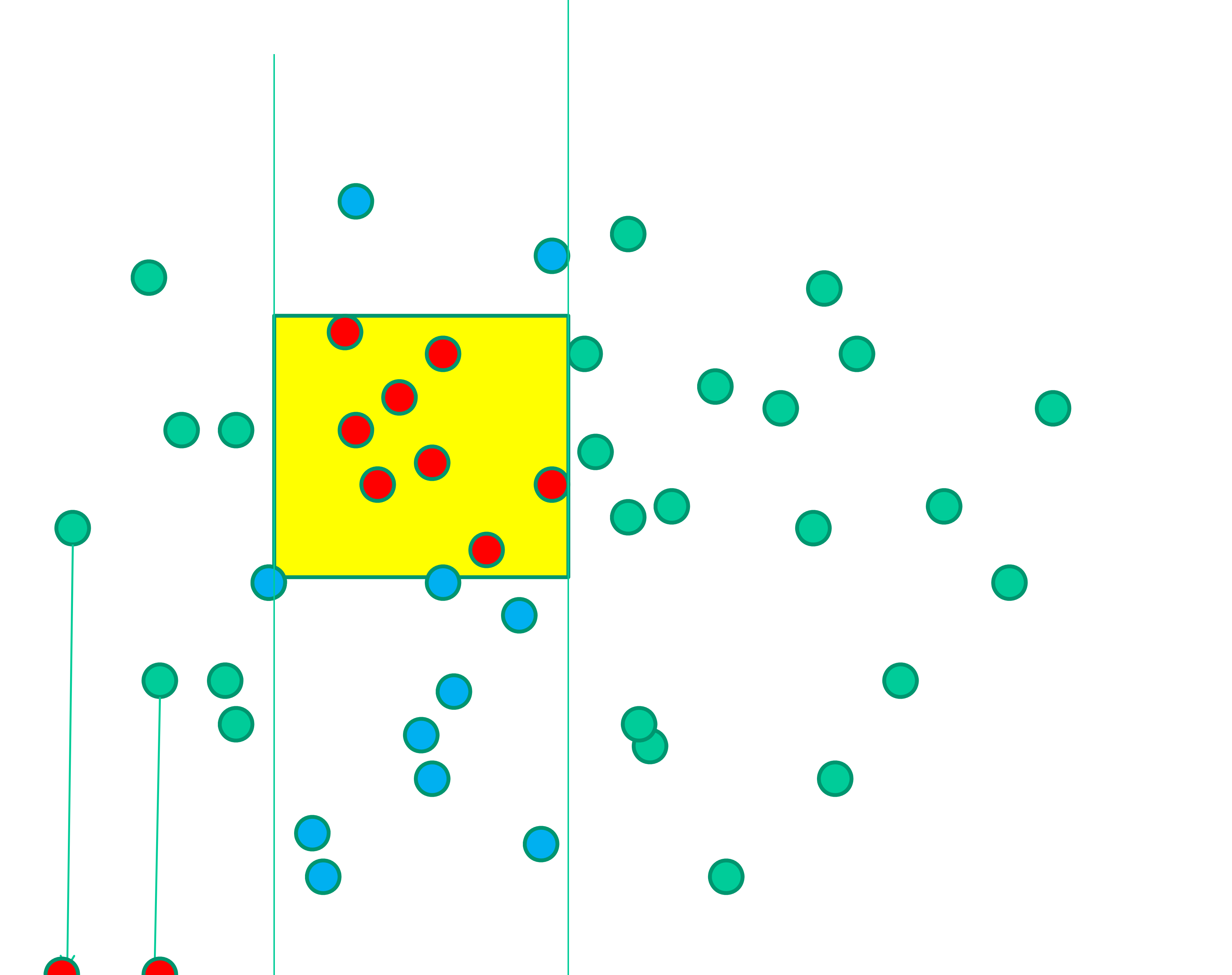


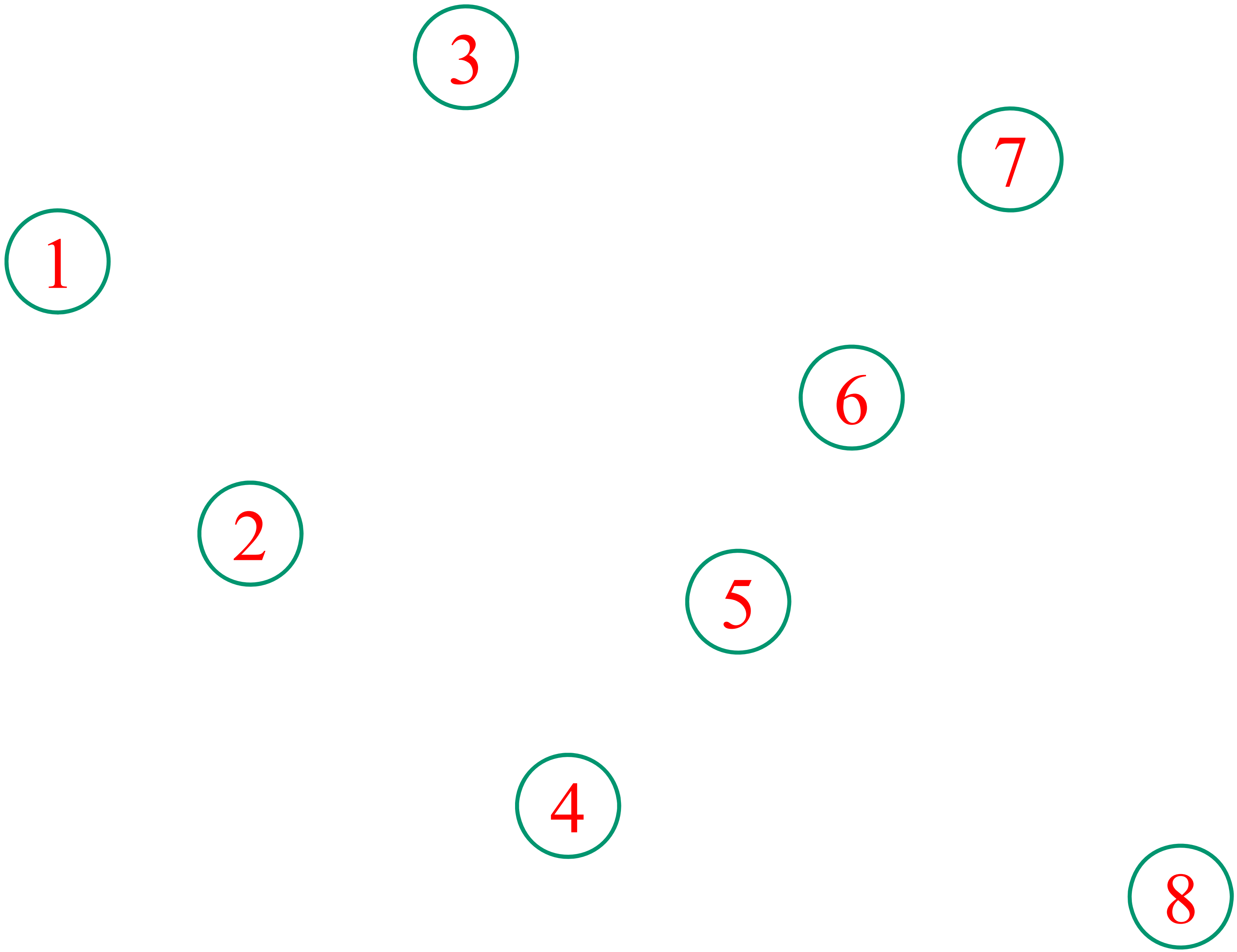


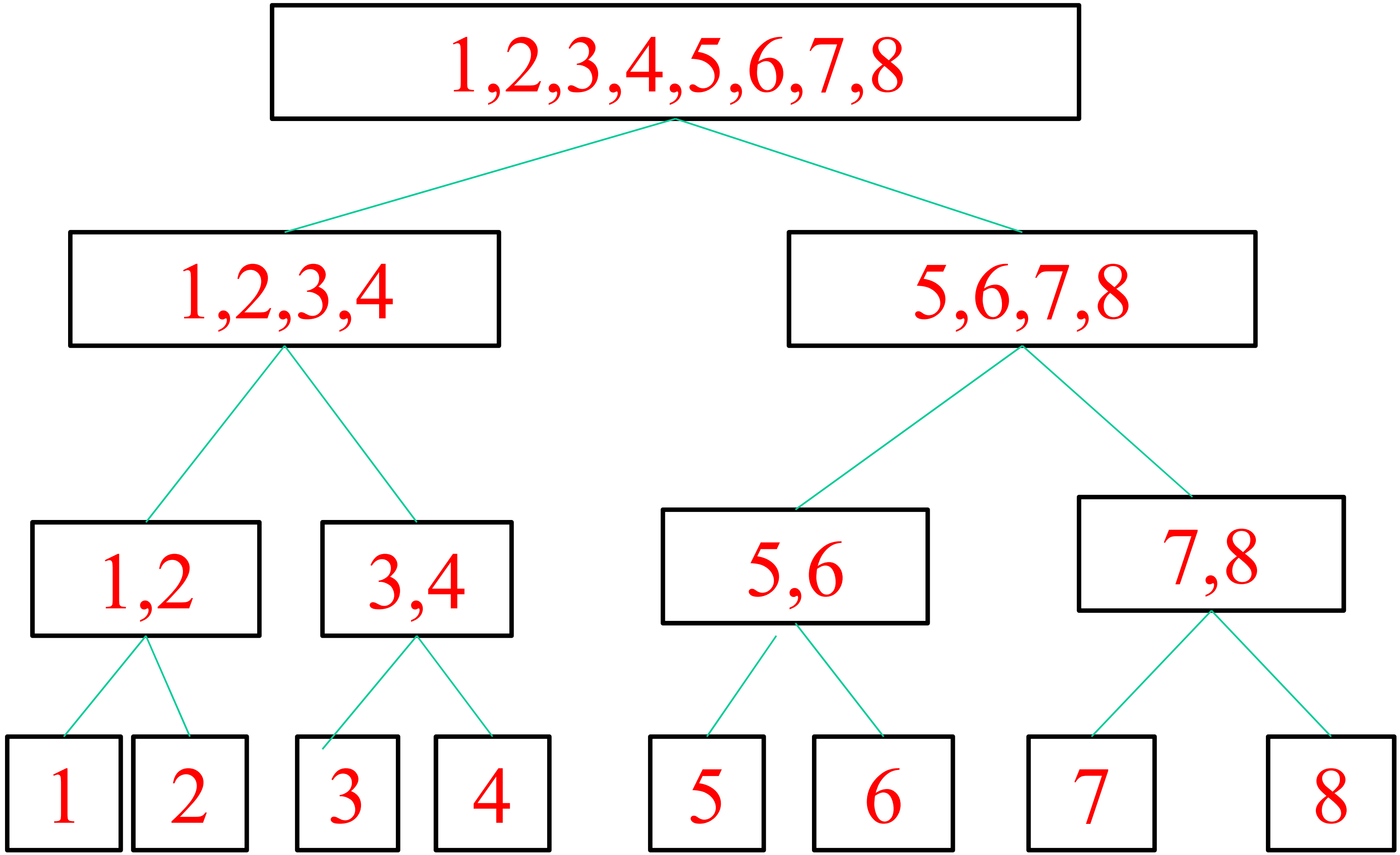
Idea

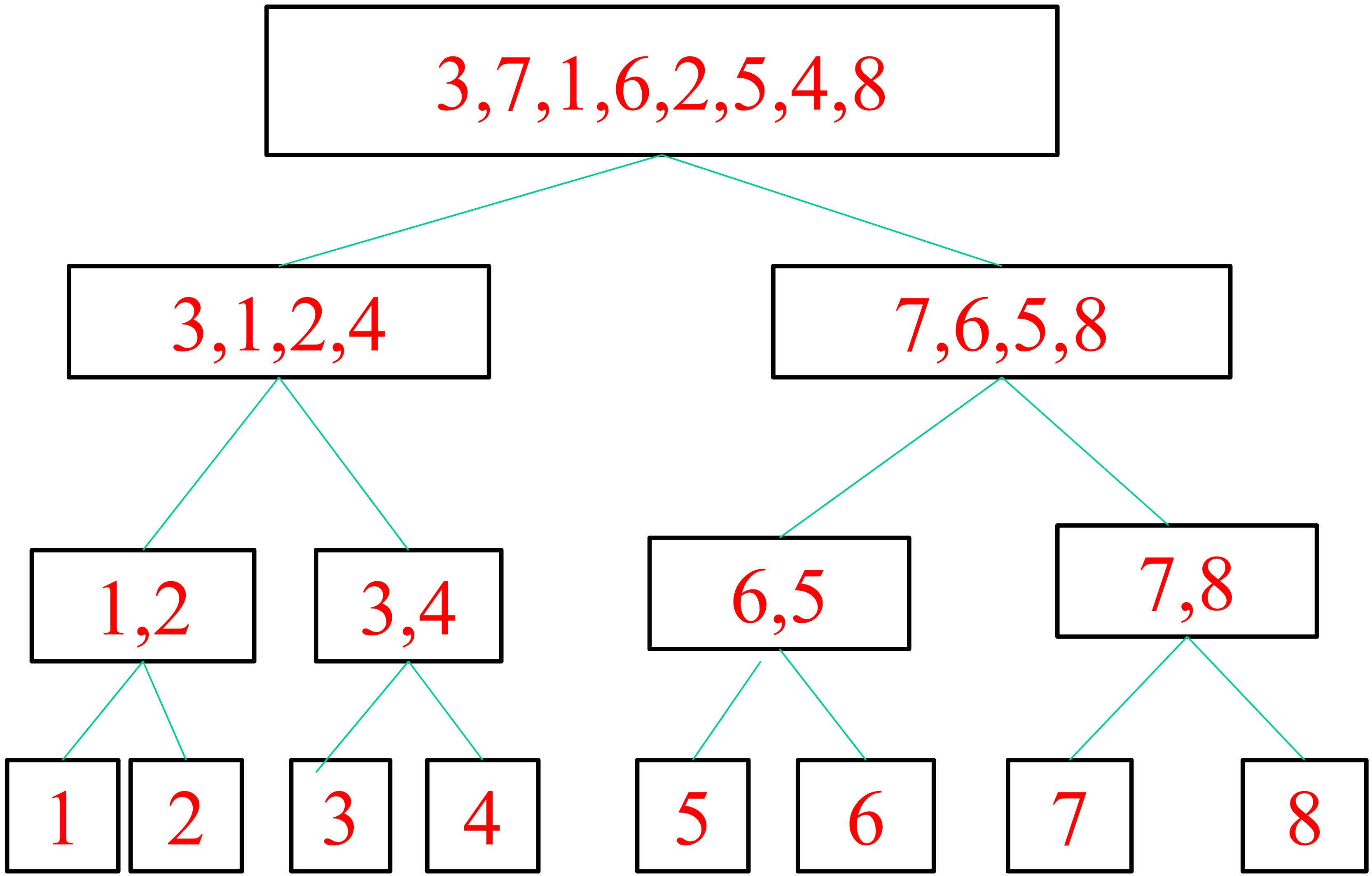
- We can find the set of points locating in a given vertical slab
  - Use interval tree on x-values.
- Then, we can find the set of points locating in the horizontal slab
  - Use interval tree of y-values.



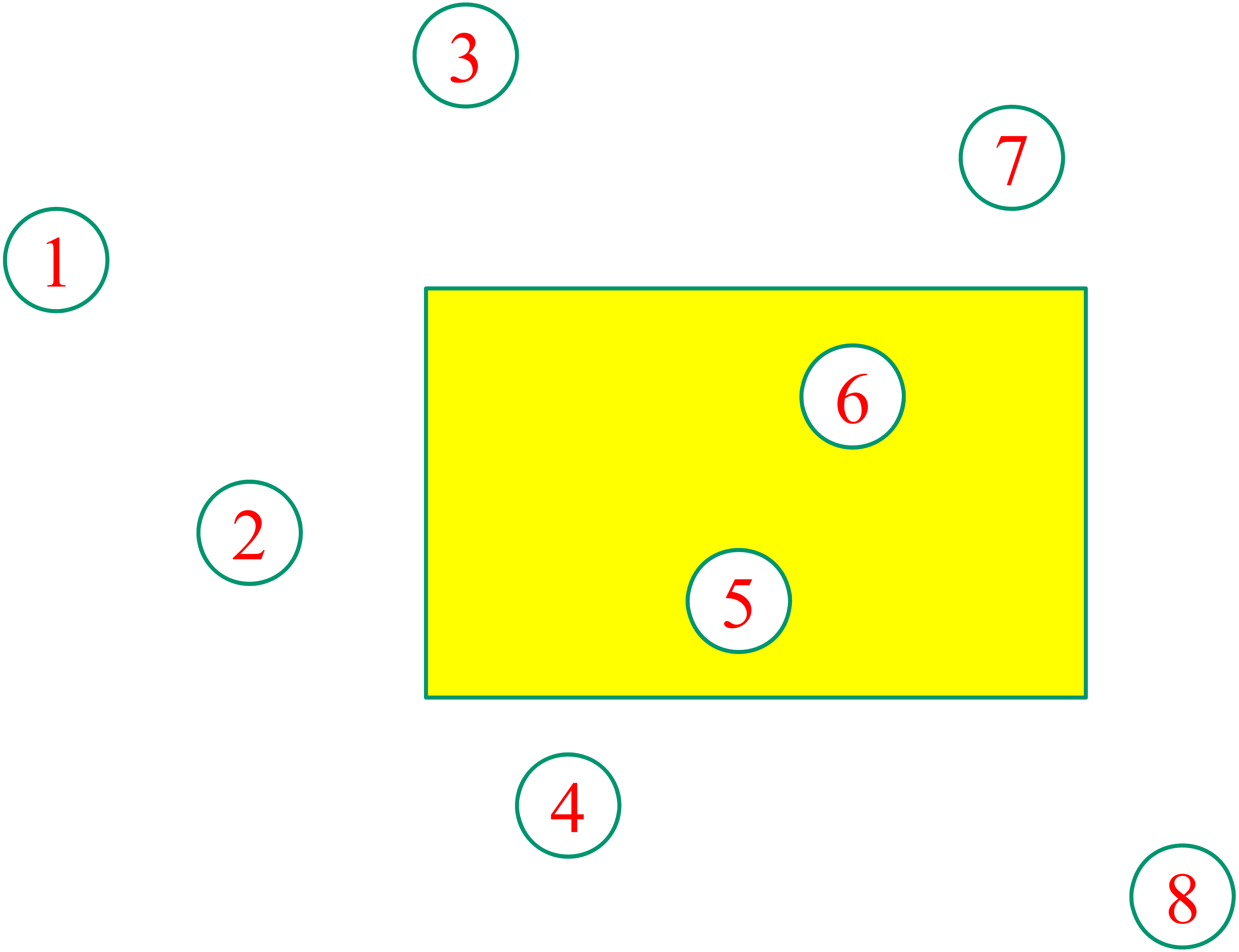


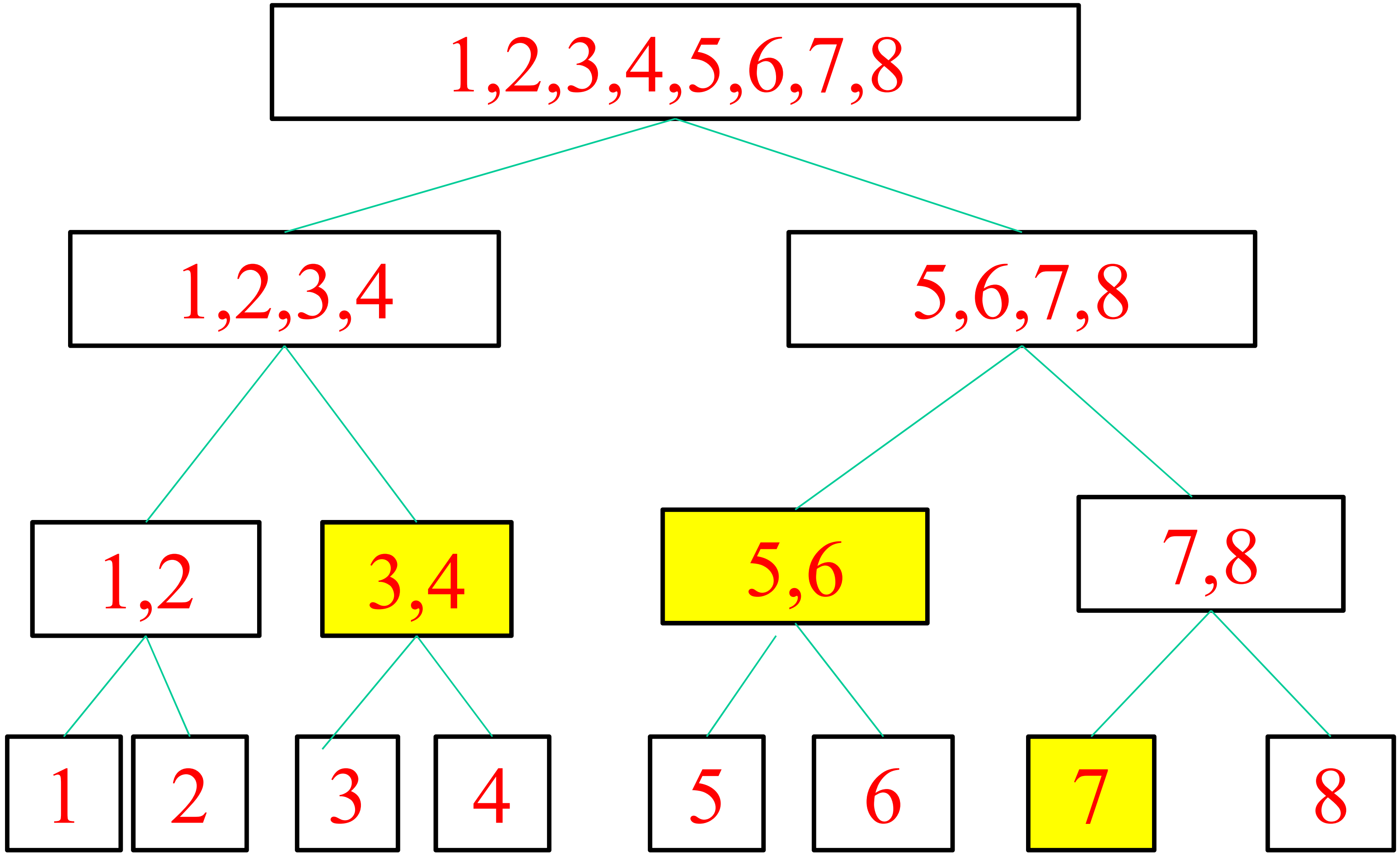

3
7
1
6
2
5
4
8

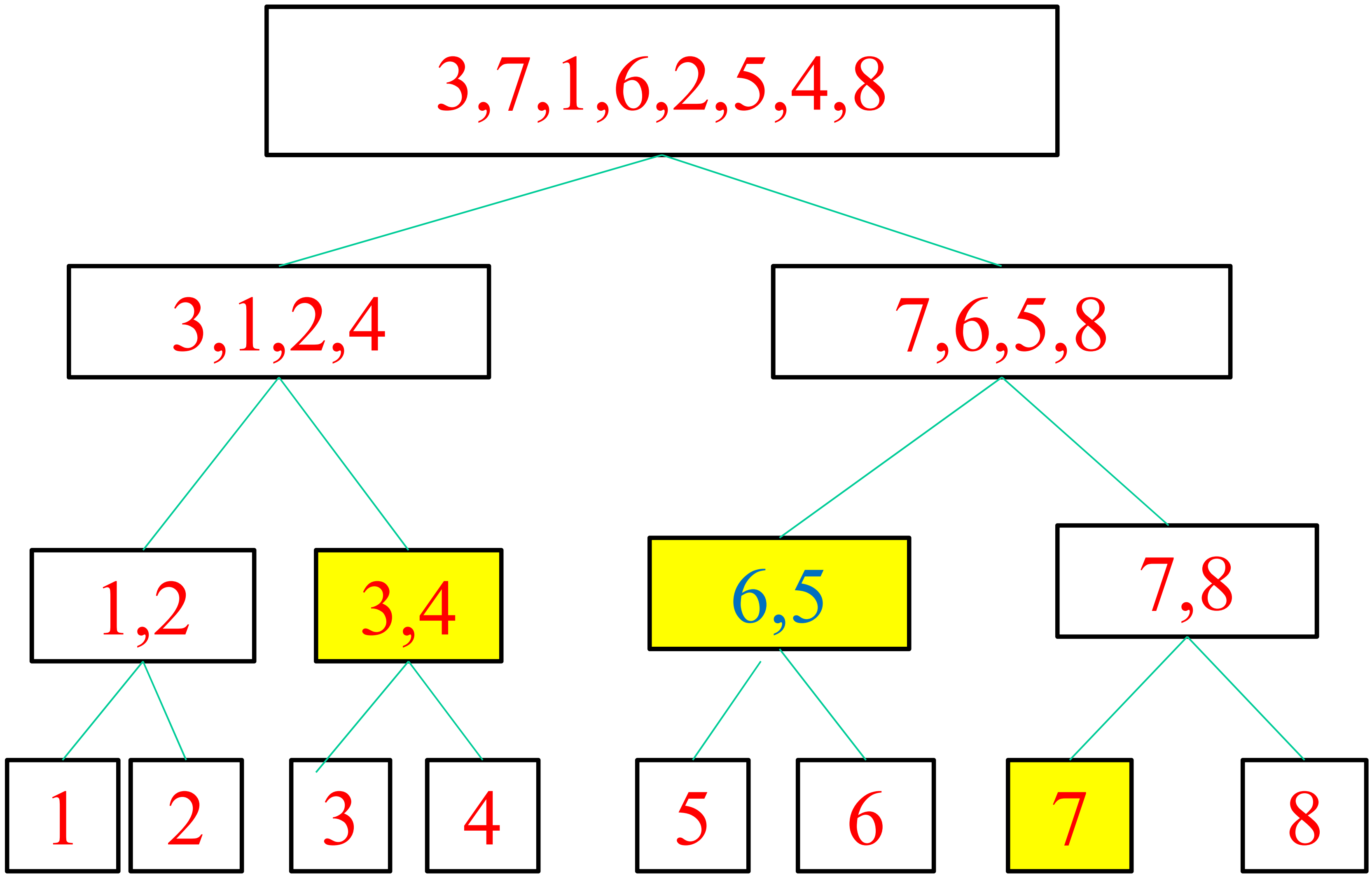

1,2,3,4,5,6,7,8
1,2,3,4
5,6,7,8
1,2
3,4
5,6
7,8
1
2
3
4
5
6
7
8



3,7,1,6,2,5,4,8
3,1,2,4
7,6,5,8
1,2
3,4
6,5
7,8
1
2
3
4
5
6
7
8


3
7
1
6
2
5
4
8


1,2,3,4,5,6,7,8
1,2,3,4
5,6,7,8
1,2
3,4
5,6
7,8
1
2
3
4
5
6
7
8

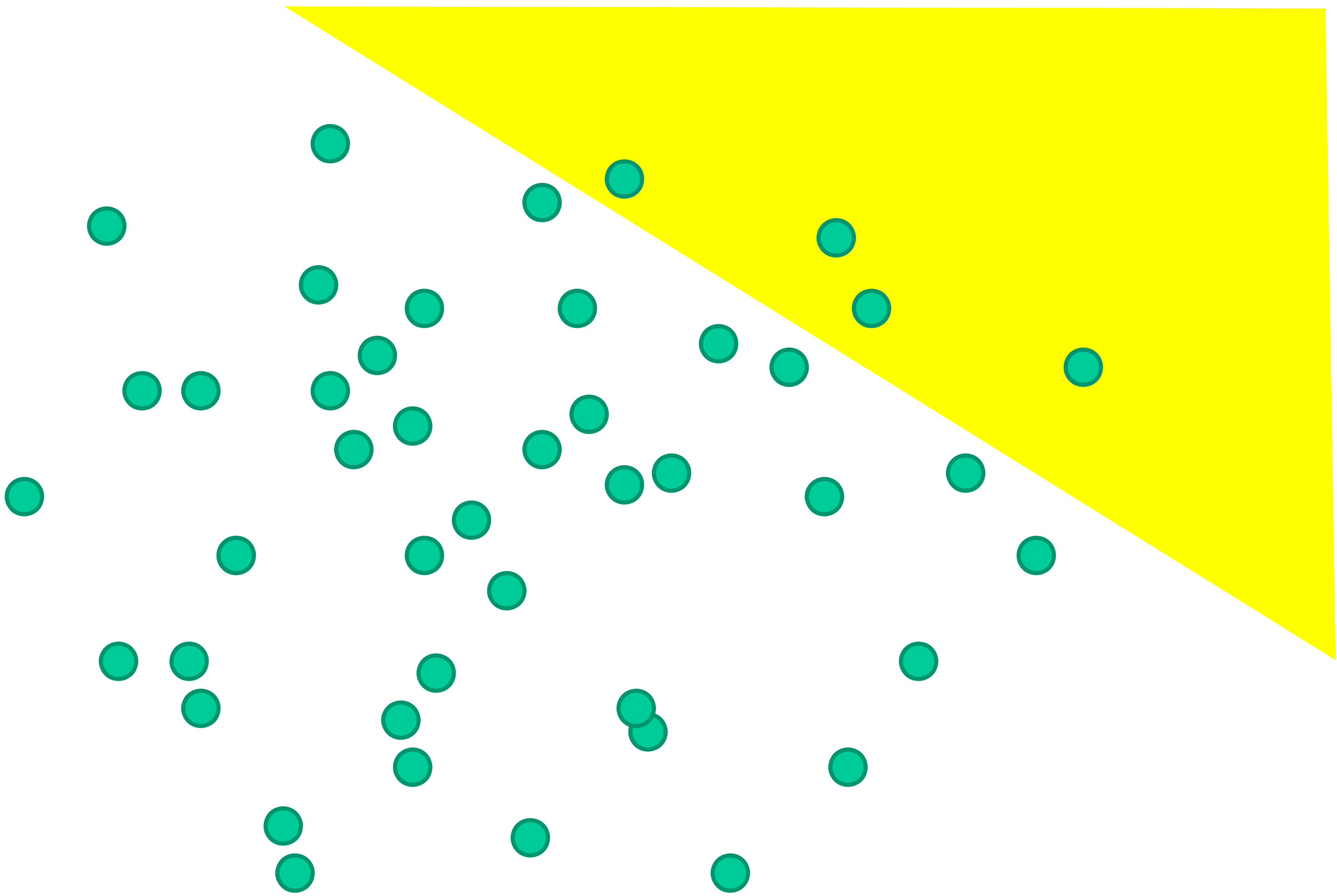

3,7,1,6,2,5,4,8
3,1,2,4
7,6,5,8
1,2
3,4
6,5
7,8
1
2
3
4
5
6
7
8

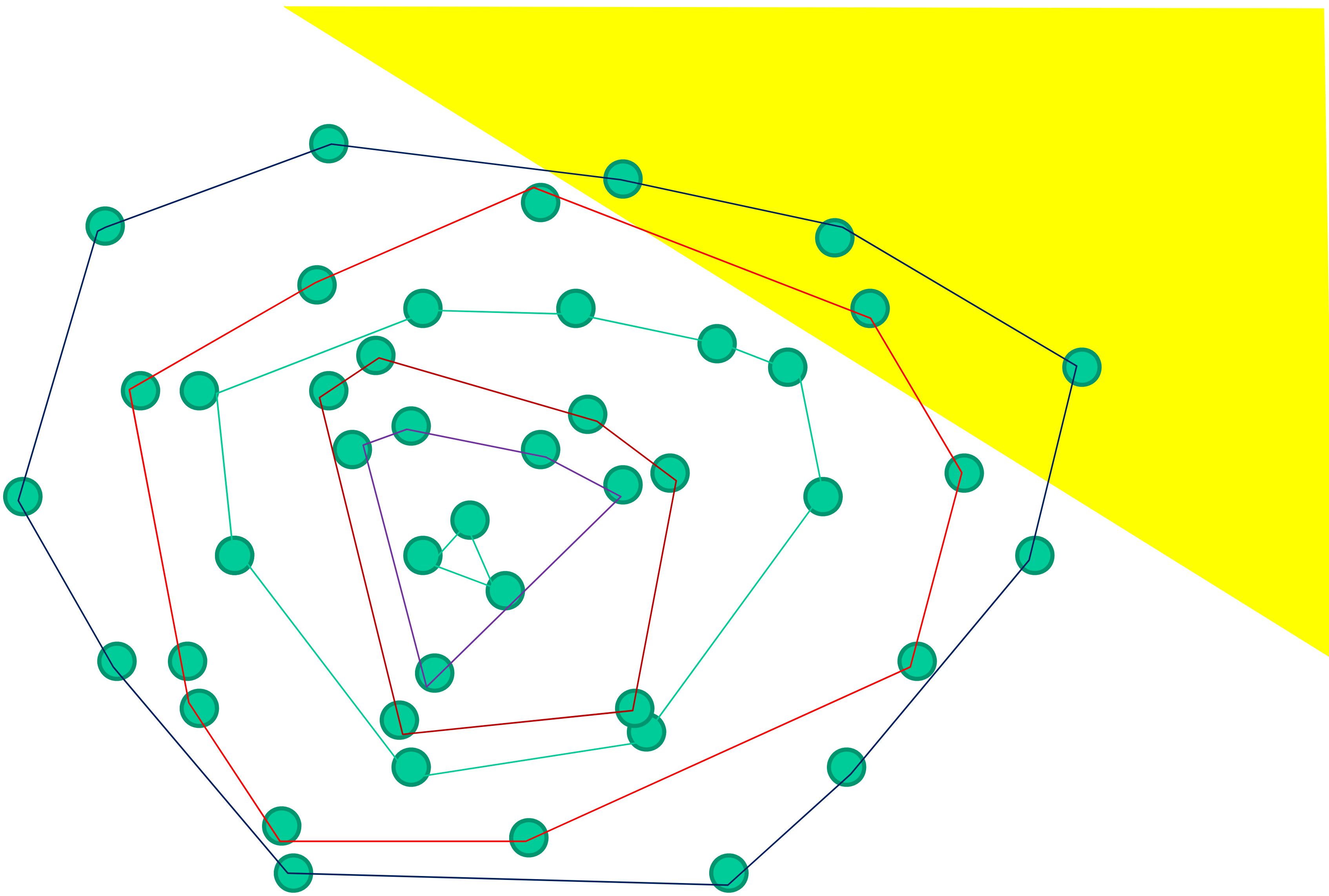
Analysis

- Size of data structure: $O(n \log n)$
- Query: $O(\log^2 n)$
- Improvement to $O(\log n)$ search
  - Similar list search method
  - Fractional Cascading
- Counting query
- High dimensional analogue

Halfplane range searching(d=2)

- Given a set S of n points in the plane, construct a data structure D(S), such that we can do the following query efficiently.
- Halfplane range query: Given a halfplane H, report the set of points of S in H efficiently
- Answer the number of points of S in H.





Onion data

Halfplane reporting query

- Data structure: Onion data structure
  - Chazelle et al 1982
- Query by a halfplane H
  - Find all convex chains intersecting H
    - Find from the outside
    - Intersecting edges are found
  - Trace on convex chains to find all points in the range
- O(k log n) time complexity
- Improved to O(k + log n)

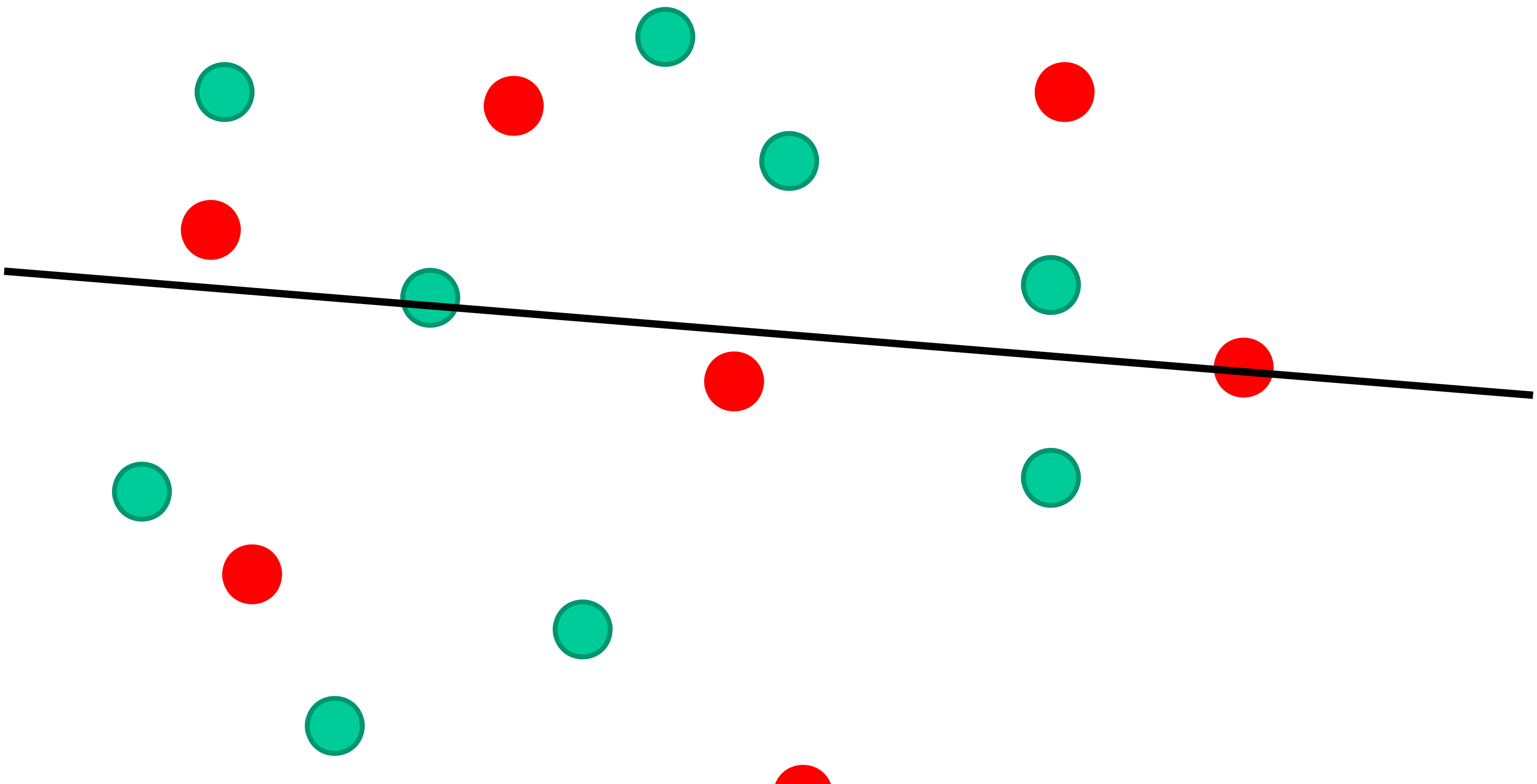
Halfplane conting query

- The onion method is expensive if k is large
- The first idea : A.C.Yao-F. Yao (1985)
  - By space subdivision with three lines
- Next idea : H. Edelsbrunner (1986)
  - By Ham-Sandwich Cut
- Next: D. Hausller and E. Welzl (1987)
  - epsilon-net and epsilon sampling
- Next step: E. Welzl (1988)
  - By low stabbing spanning tree
- Final step: J. Matousek (1990)
  - By cutting of arrangement

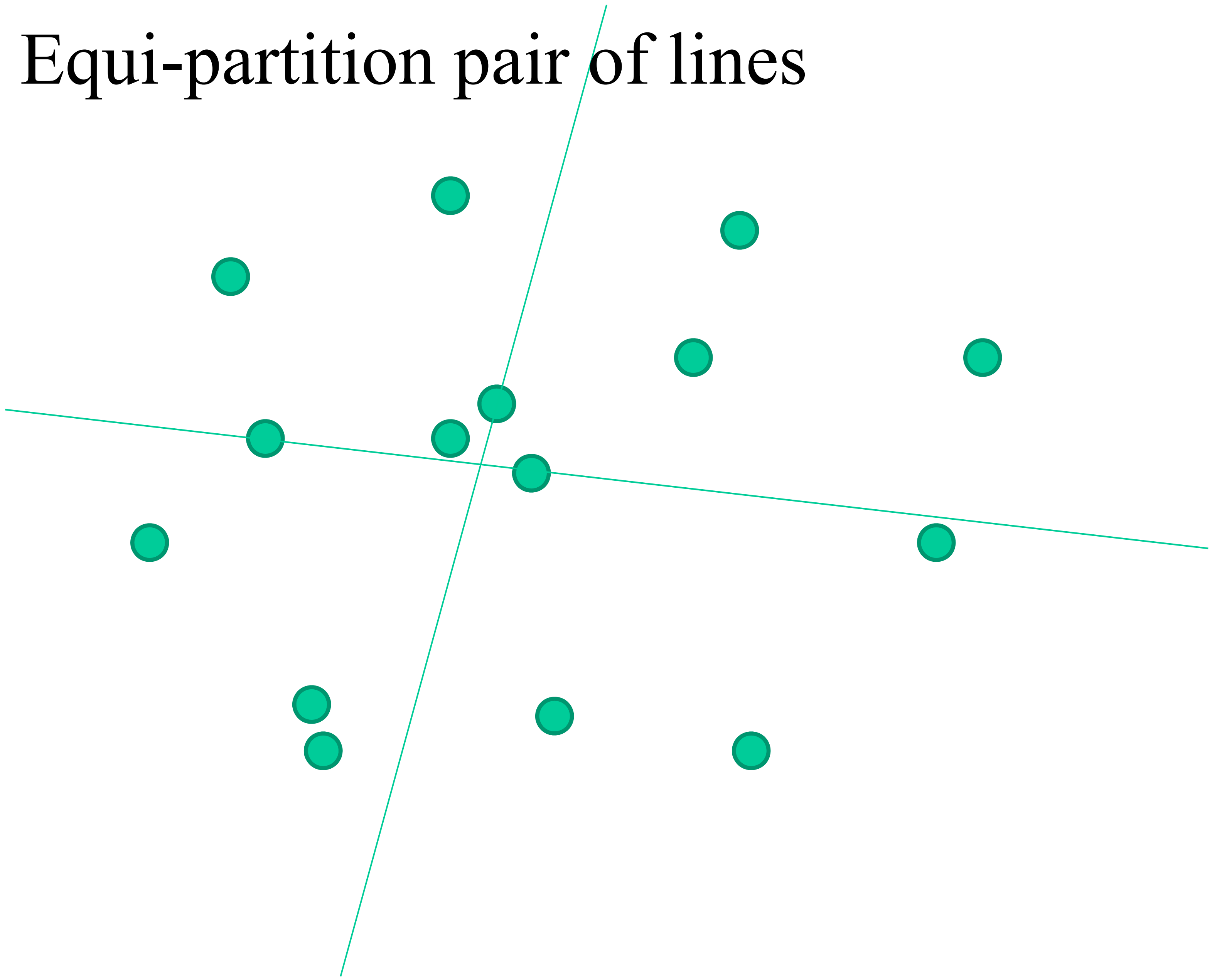
Ham sandwich cut

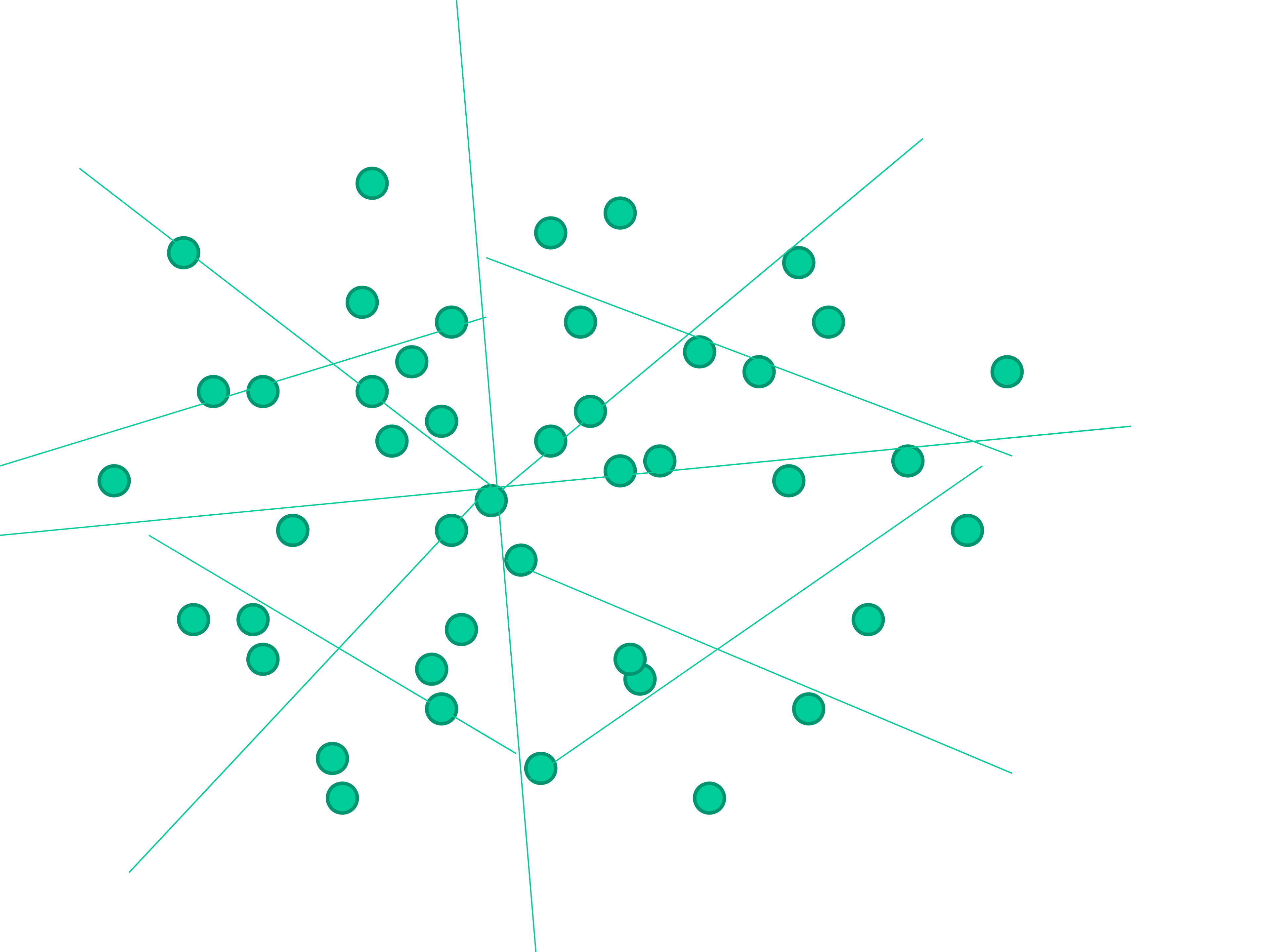
- Theorem

  Given n red points and m blue points in the plane, there is a line L such that at most n/2 red points and m/2 blue points lies in each side of L
- Obtained from Borsuk-Ulam's theorem
  - Any continuous map from circle (sphere) to line (space) has an antipodal pair to map to a same point.
  - J. Matousek: Using the Borsuk-Ulam's Theorem



Equi-partition pair of lines





How to search?

- Suppose lines L and L' partition the space into four cones.
- Then, any line M intersects at mot three cones.
- This gives the following recursion of the search
  - $T(n) = 3\, T(n/4) + O(1)$
  - This gives $T(n) = O(n^{0.7})$