Multidimensional Tree

Pelatnas 2 TOKI 2015

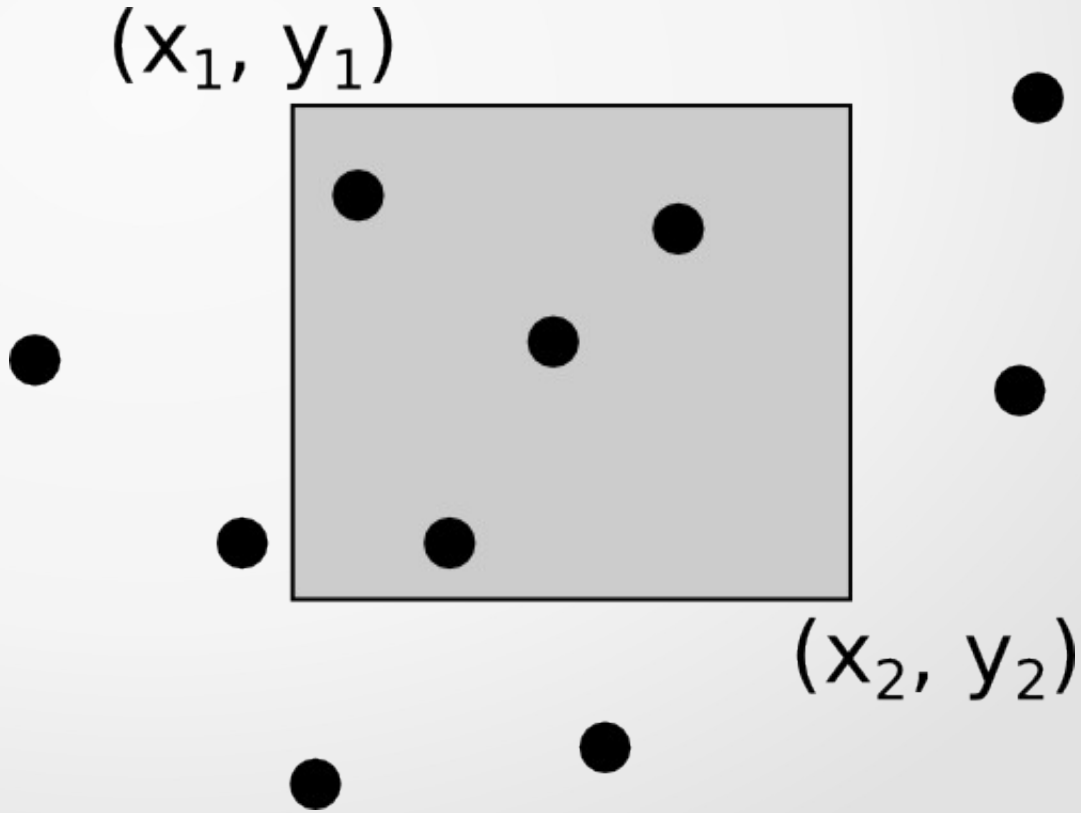
William Gozali

Outline

- Motivasi
- Konsep
- Petunjuk Implementasi
- Latihan

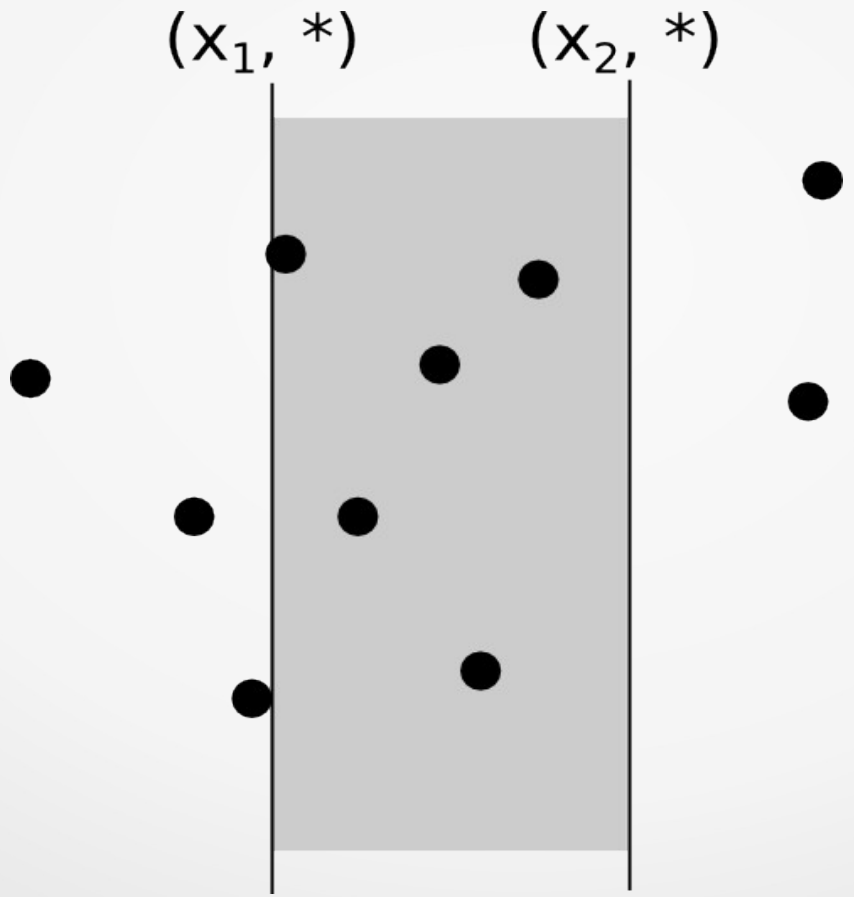
Motivasi

- Diberikan N titik di koordinat Cartesian
- Query: banyaknya titik yang ada di subregion ((x1, y1), (x2, y2))!

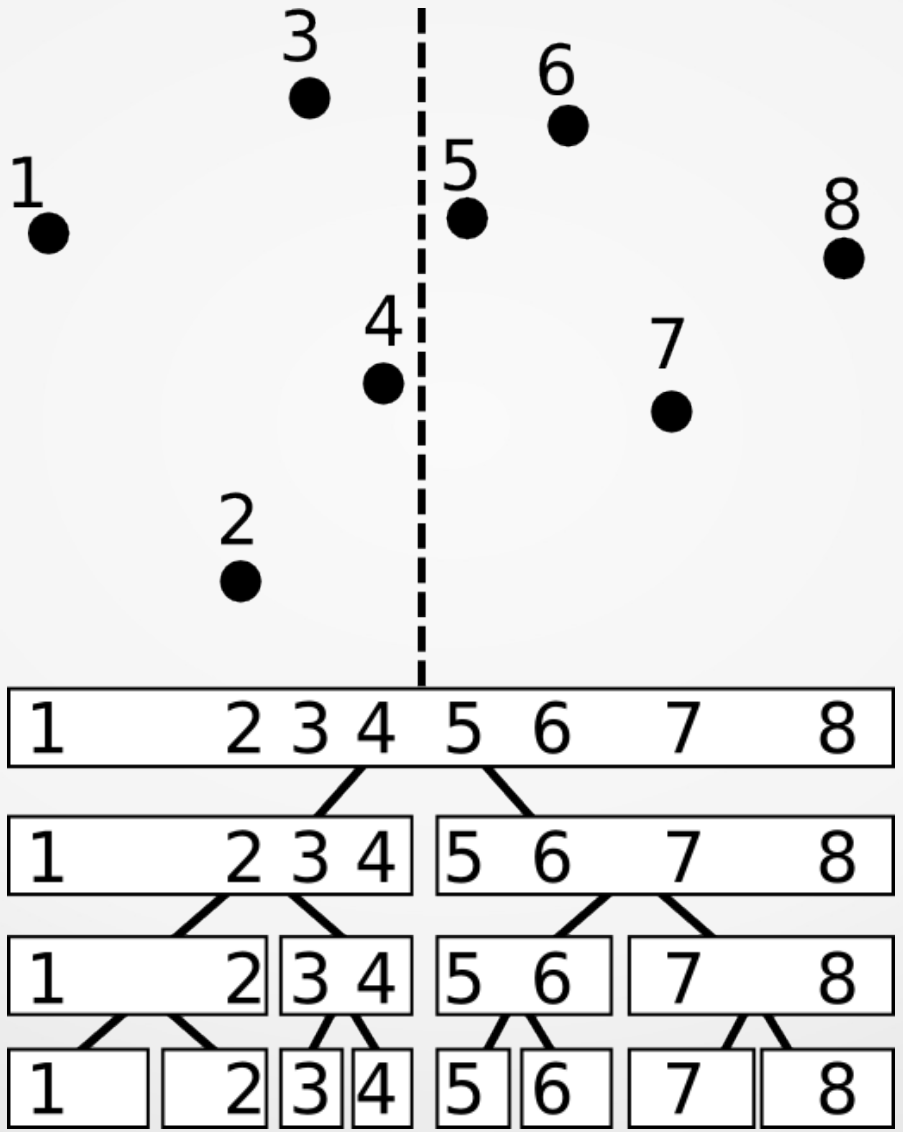


Penyederhanaan:

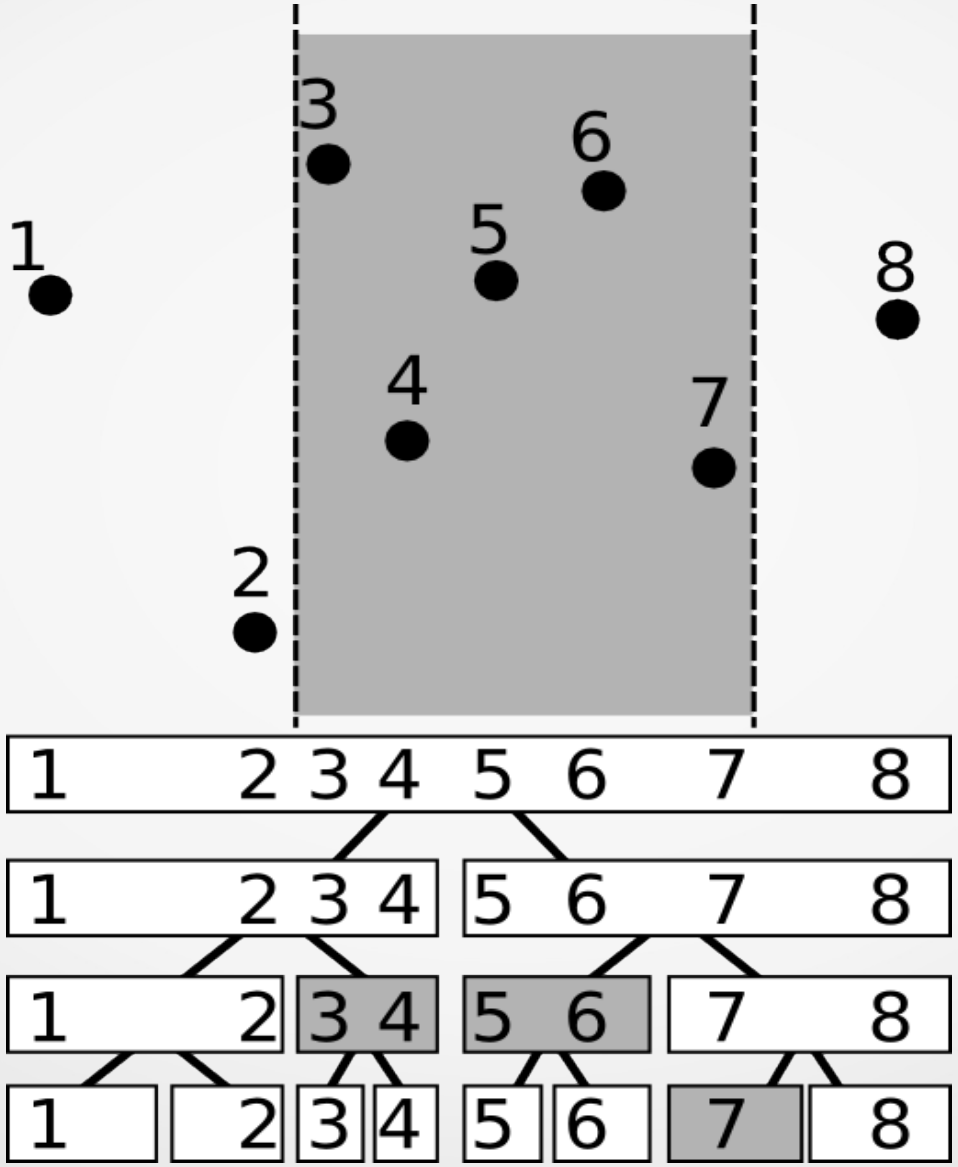
- Anggap query: banyaknya titik di subregion ((x1, *), (x2, *))



Buat Segment tree



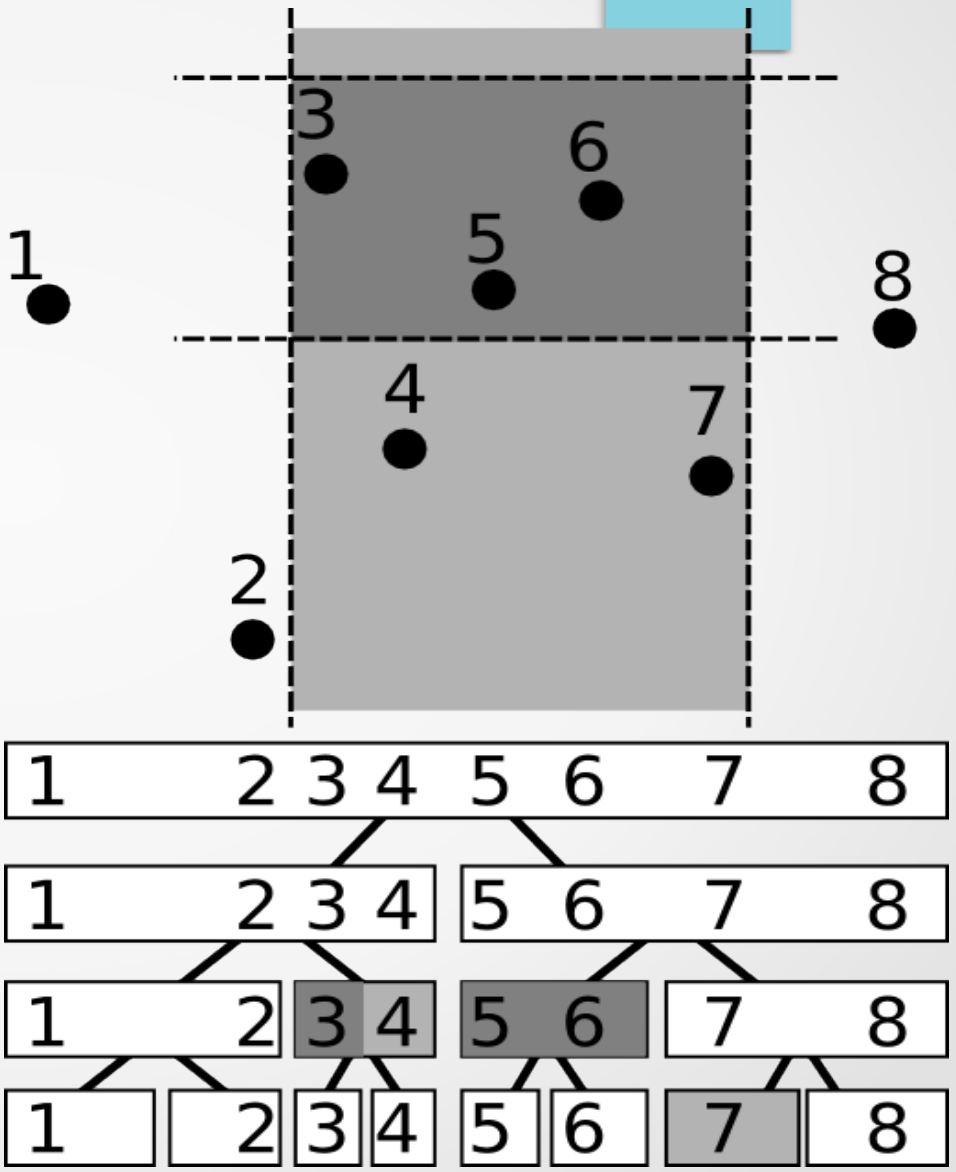
Solusi – O(log N)



Solusi?

+linear search
per segmen?

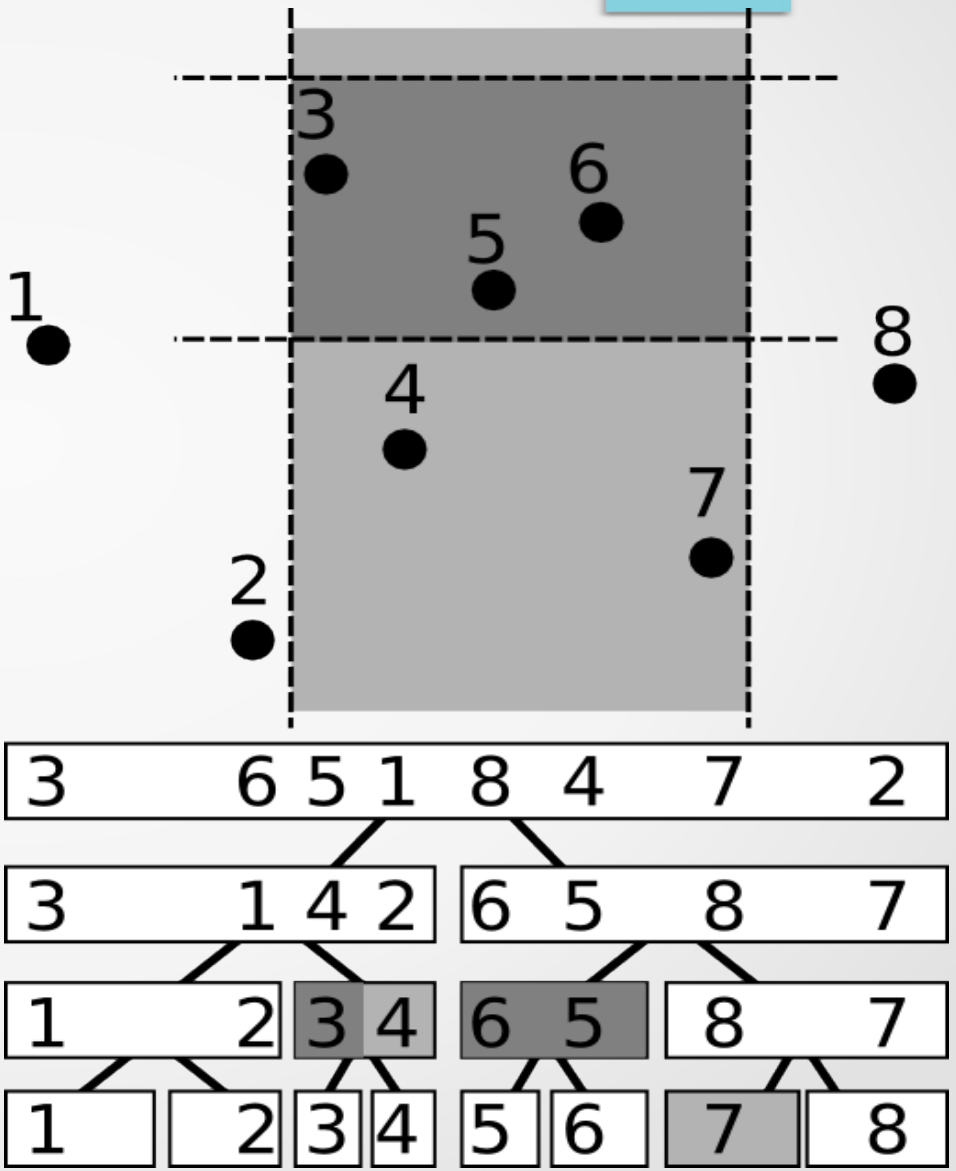
= O(N) per segmen



Solusi - $O(\log^2 N)$

+sort & binary search per segmen?

= O(log N) per segmen

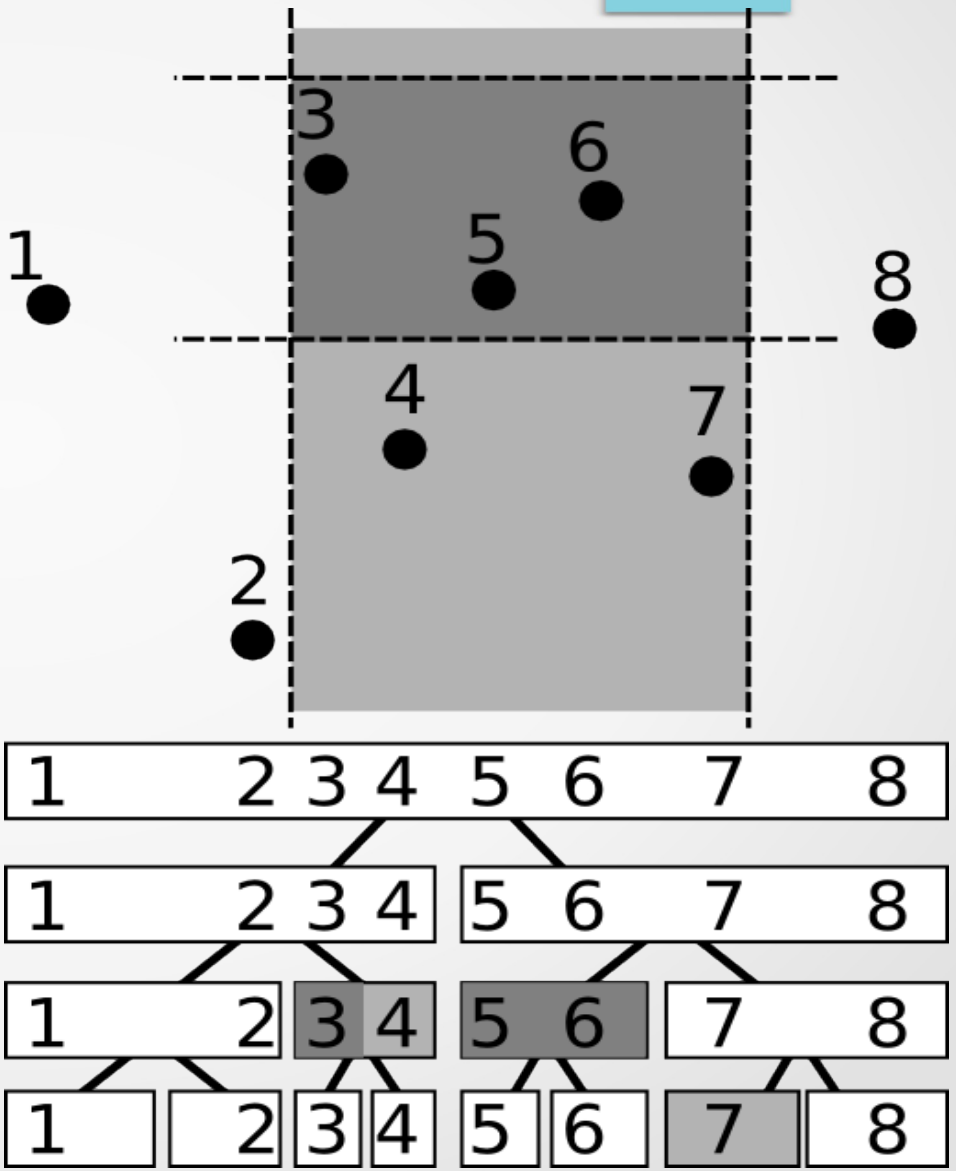


Visualisasi yang lebih cantik

- http://blog.ezyang.com/2012/02/visualizing-range-trees/

Pertanyaan

- Kompleksitas memori?
- Bagaimana cara build?
- Kompleksitas waktu build?



Ringkasan

- vector<int> tree[2*MAXN]
- MAXN: bilangan 2 pangkat terdekat yang >= N
- Memori keseluruhan O(N log N)
- Waktu build O(N $\log^2$ N):
  - O(N log N) untuk masukkan titik-titik, karena O(log N) untuk sekali memasukkan, dan ada N titik
  - O(N $\log^2$ N) untuk sort setiap segmen, karena O(N log N) untuk setiap lapis, dan ada O(log N) lapis
- Waktu query O($\log^2$ N):
  - Terdapat O(log N) segmen, masing-masing di-binary search O(log N)

Contoh kode

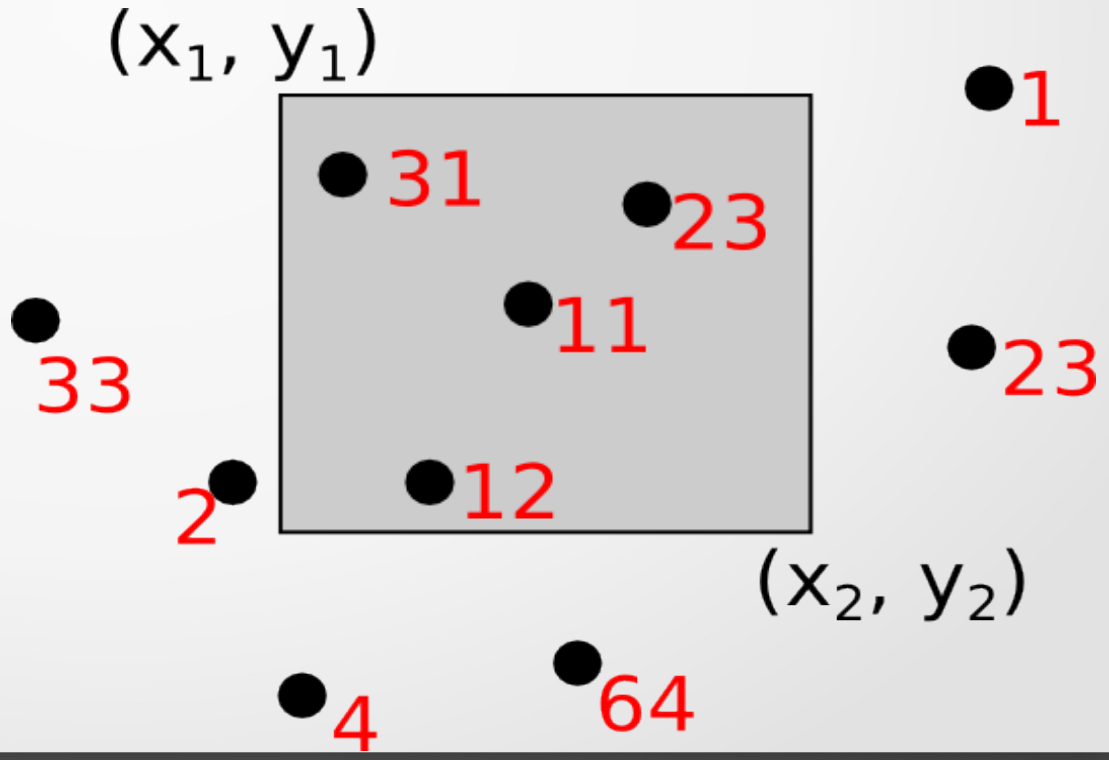
```
void build(){
  for (int i = 0; i < N; i++){
    insert(0, 0, N-1, x[i], y[i]);
  }
  for (int i = 0; i < 2*MAXN; i++){
    sort(tree[i].begin(), tree[i].end());
  }
}

void insert(int nod, int ki, int ka, int x, int y){
  tree[nod].push_back(y);

  if (ki < ka){
    int tgh = (ki + ka) >> 1;

    if (x <= tgh) insert(2*nod+1, ki, tgh, x, y);
    if (x >  tgh) insert(2*nod+2, tgh+1, ka, x, y);
  }
}
```

Contoh kode (lanj.)

```
int countBetween(vector<int> &vec, int y1, int y2){
  int u = upper_bound(vec.begin(), vec.end(), y2);
  int l = lower_bound(vec.begin(), vec.end(), y1-1);
  return l - u;
}

int query(int nod, int ki, int ka, int x1, int y1, int x2, int y2){
  if ((x1 <= ki) && (ka <= x2)){
    return countBetween(tree[nod], y1, y2);
  }else{
    int tgh = (ki + ka) >> 1;
    int ret = 0;

    if (x1 <= tgh) ret += query(2*nod+1, ki, tgh, x1, y1, x2, y2);
    if (x2 >  tgh) ret += query(2*nod+2, tgh+1, ka, x1, y1, x2, y2);

    return ret;
  }
}
```

Modifikasi Soal

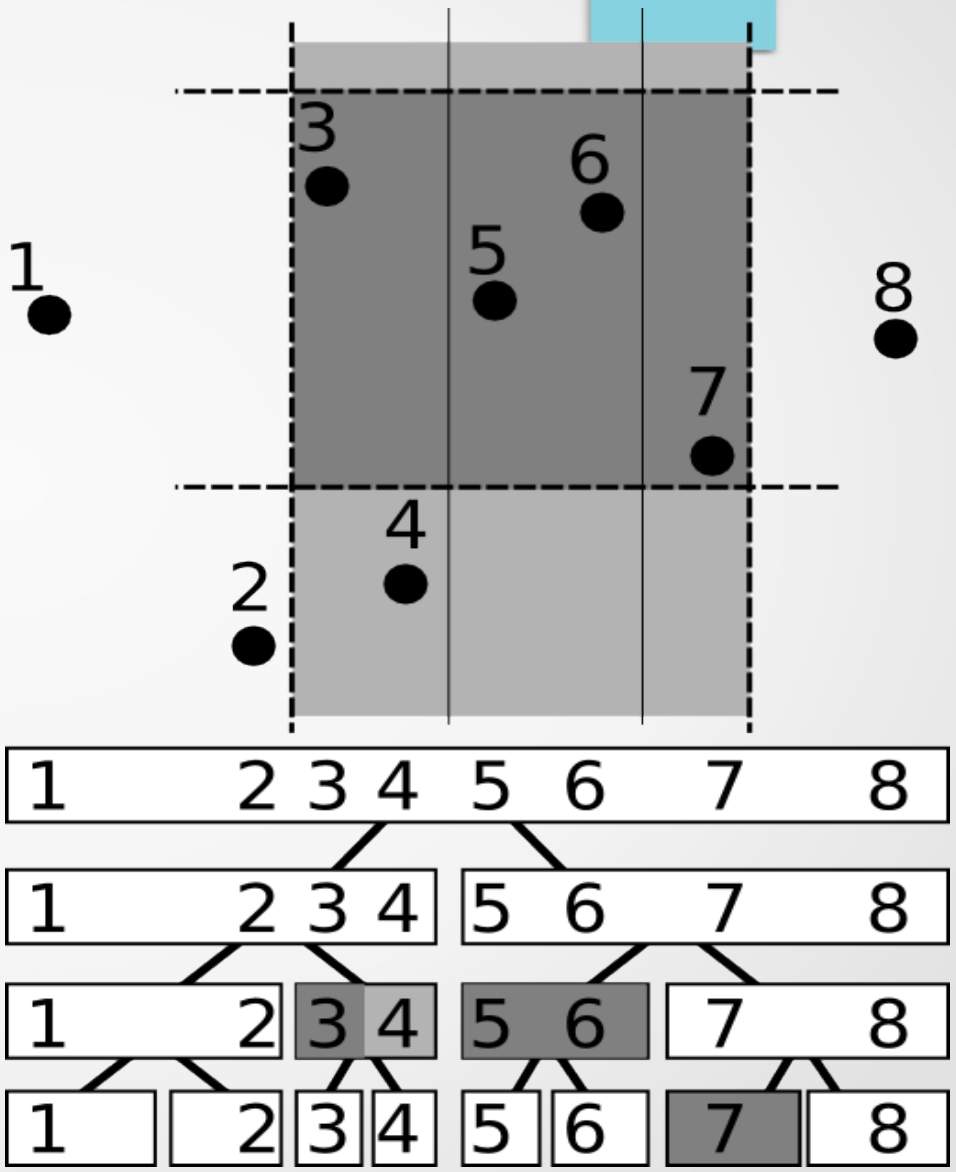
- Titik = koin
- Setiap koin memiliki nilai nominal tertentu
- Query: nominal terbesar koin yang ada di subregion ((x1, y1), (x2, y2))!



Solusi – $O(\log^2 N)$

- Setiap segmen butuh kemampuan melayani queryMax untuk suatu **subbarisan** dengan efisien
- Solusi:

  setiap segmen => RMQ



Kebiasaan Implementasi:

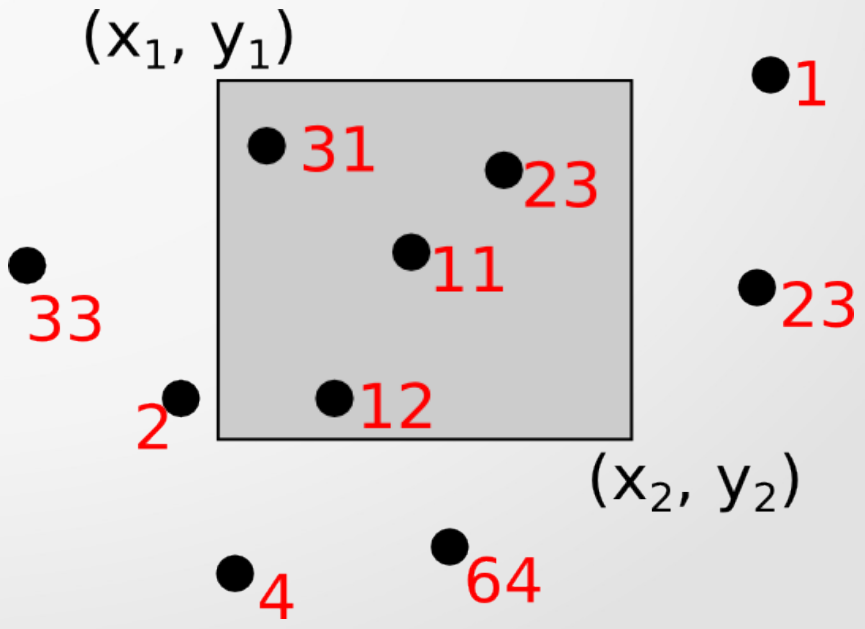
```
struct innerTree{
    vector<int> tree

    void build()
    int queryMax(int a, int b)
}

innerTree rangeTree[2*MAXN]

buildRangeTree()
rangeQuery(x1, y1, x2, y2)
```

Kebiasaan Implementasi - alternatif:

```
struct innerTree{
    vector<int> tree

    void build()
    int queryMax(int a, int b)
}

struct rangeTree{
    vector<innerTree> tree

    build()
    rangeQuery(x1, y1, x2, y2)
}
```

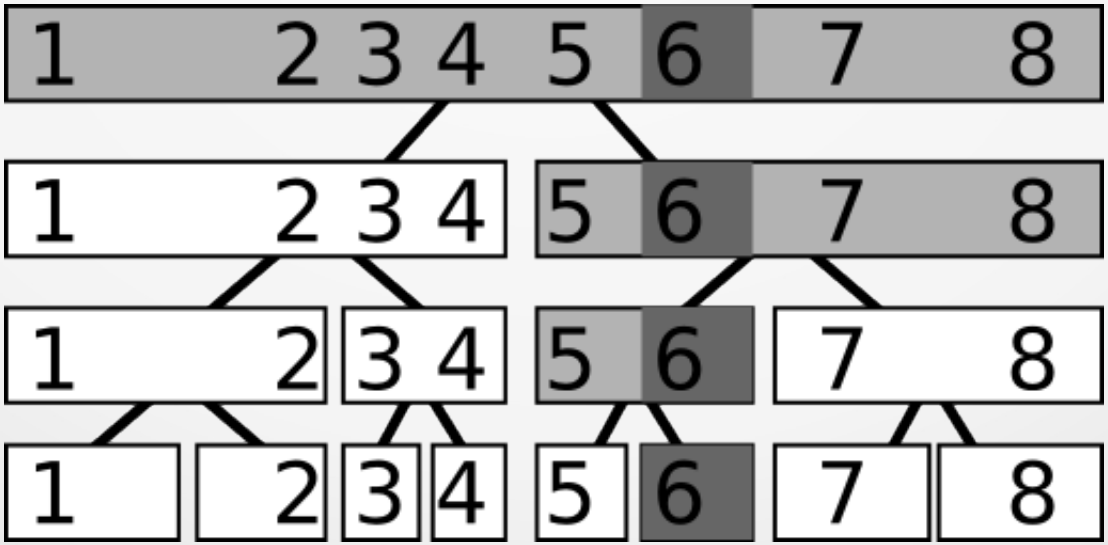
Modifikasi Soal (lagi)

- Titik = koin
- Setiap koin memiliki nilai nominal tertentu
- Update: ganti nominal suatu koin menjadi nilai lain
- Query: nominal terbesar koin yang ada di subregion ((x1, y1), (x2, y2))!



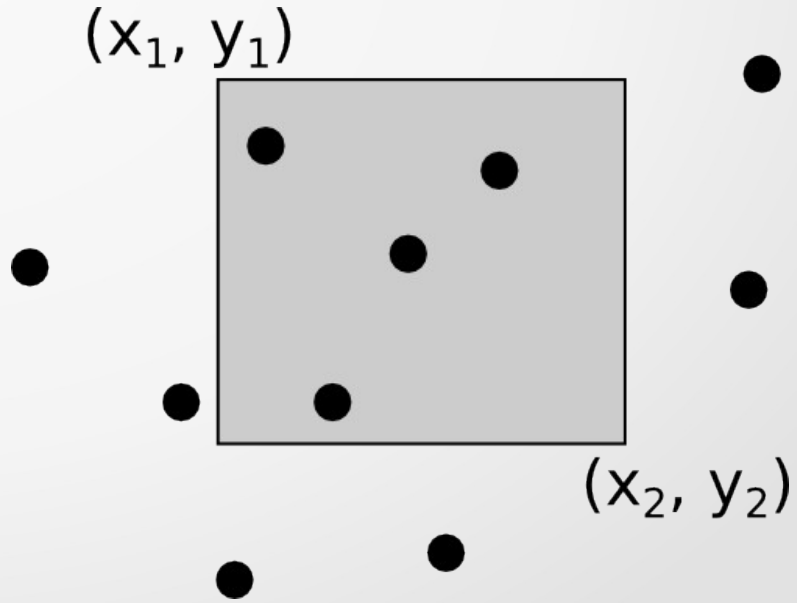
Operasi Update – $O(\log^2 N)$

- Update innerTree, O(log N)
- Maksimal terdapat O(log N) innerTree yang perlu di-update
- Kompleksitas update: $O(\log^2 N)$



Modifikasi Soal (lagi)

- Awalnya tidak ada titik
- Insert: masukkan suatu titik di koordinat (x, y)
- Delete: hapus suatu titik di koordinat (x, y)
- Query: banyaknya titik yang ada di subregion ((x1, y1), (x2, y2))!



Solusi

- Sama seperti solusi yang terakhir, cukup ganti RMQ menjadi range count query
- Range count query bisa diimplementasikan dengan segment tree, BIT, BST

Latihan - 1

- Ada N kota berjejer, dari kota 1 sampai kota N
- Ada M bus, masing-masing bermula dari kota-a dan berakhir di kota-b (a < b)
- Ada banyak query:
  - Diberikan 2 kota (misalnya p dan q), tentukan banyaknya bus yang melewati kedua kota tersebut

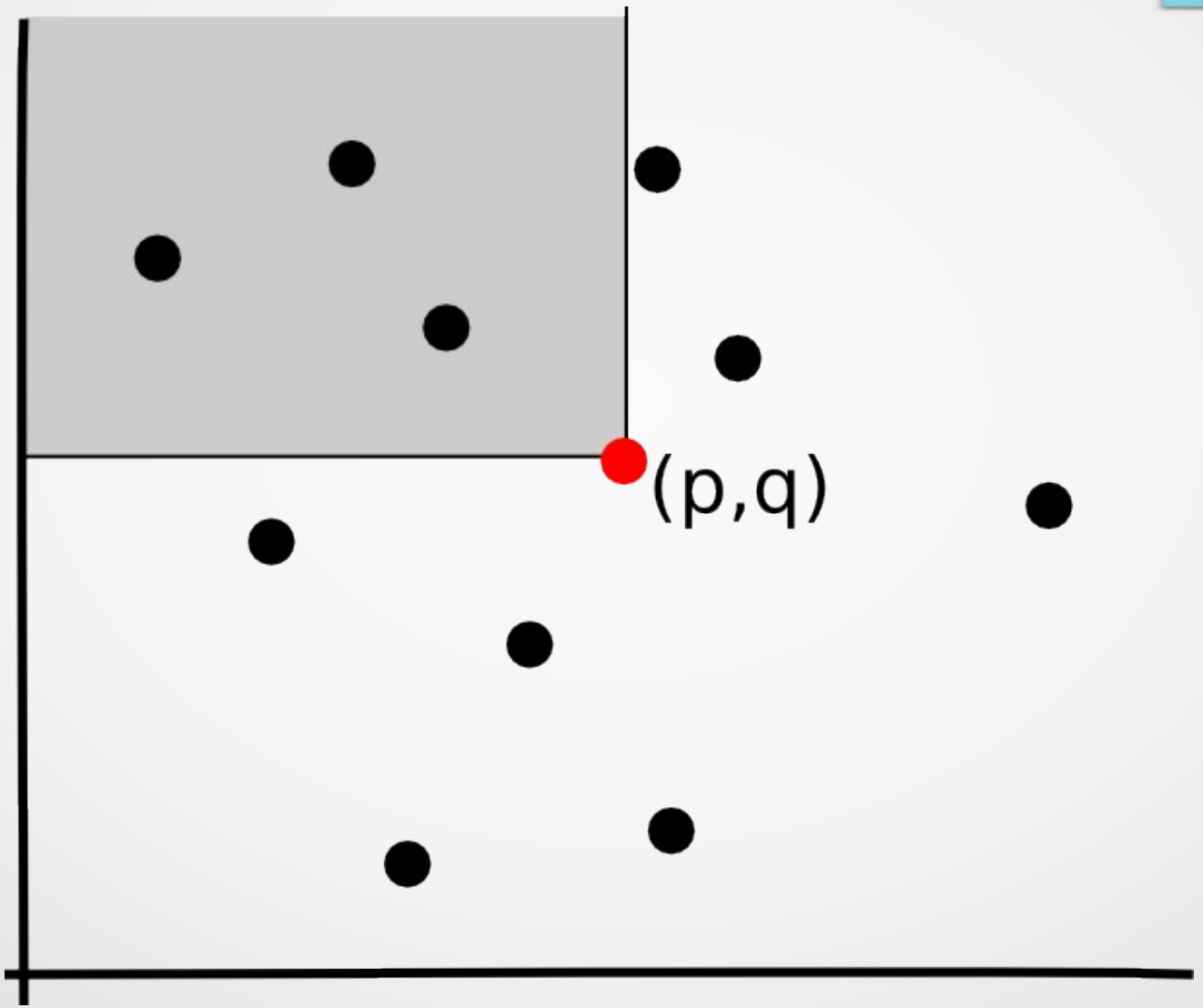
Solusi

- Bus dari kota-a ke kota-b = titik (a, b)
- Diberikan 2 kota (misalnya p dan q), tentukan banyaknya bus yang melewati kedua kota tersebut

  =

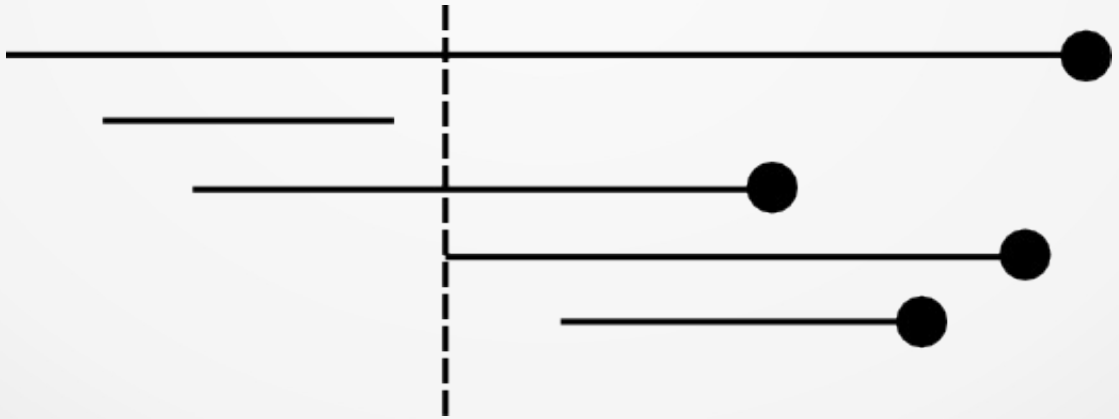
  Diberikan titik (p, q), tentukan banyaknya titik (x, y) yang memenuhi (x <= p) dan (q <= y)
- Kompleksitas $O(\log^2 N)$ per query

Solusi



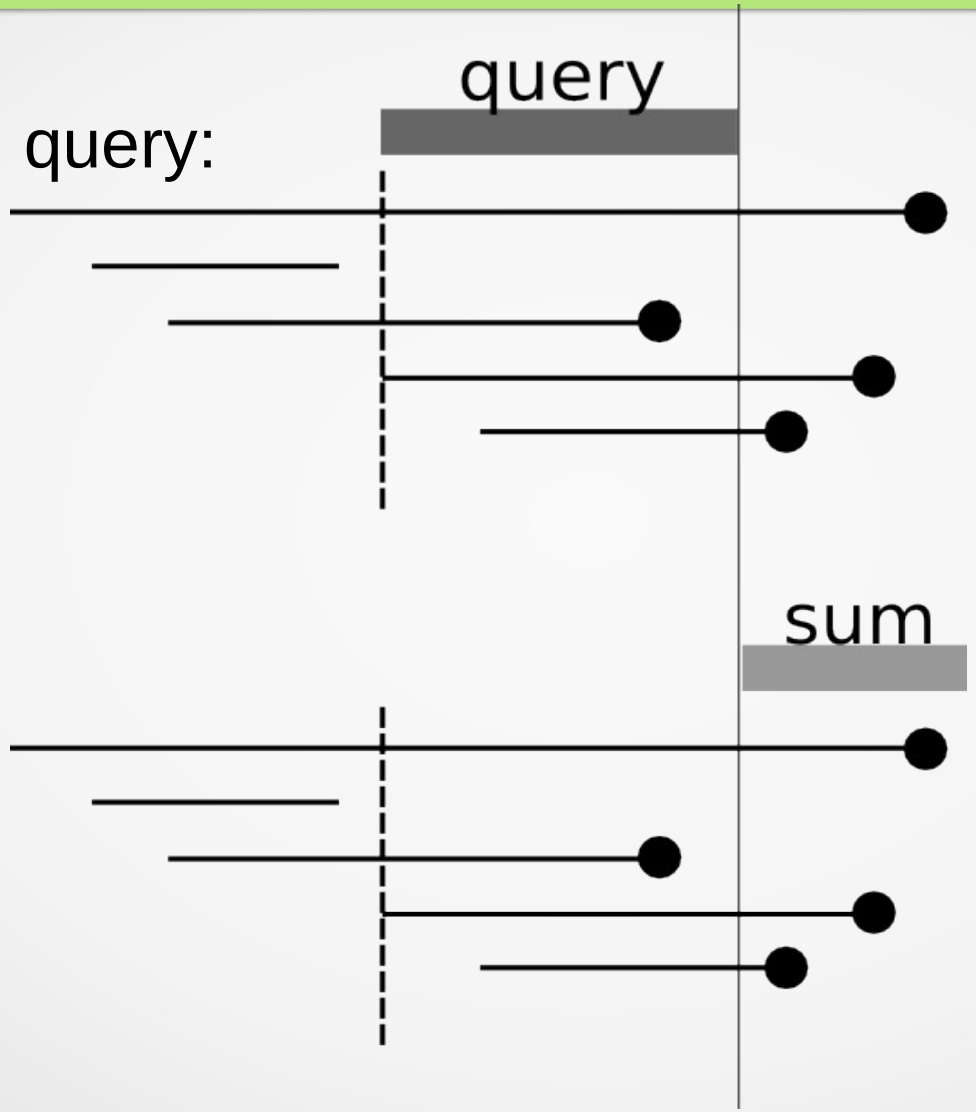
Solusi alternatif – tanpa range tree

- Lakukan offline query
- Sort semua jalur bus dan query berdasarkan indeks kota awal, kemudian line sweep



Solusi alternatif

- Melayani query:



Kompleksitas O(log N) per query