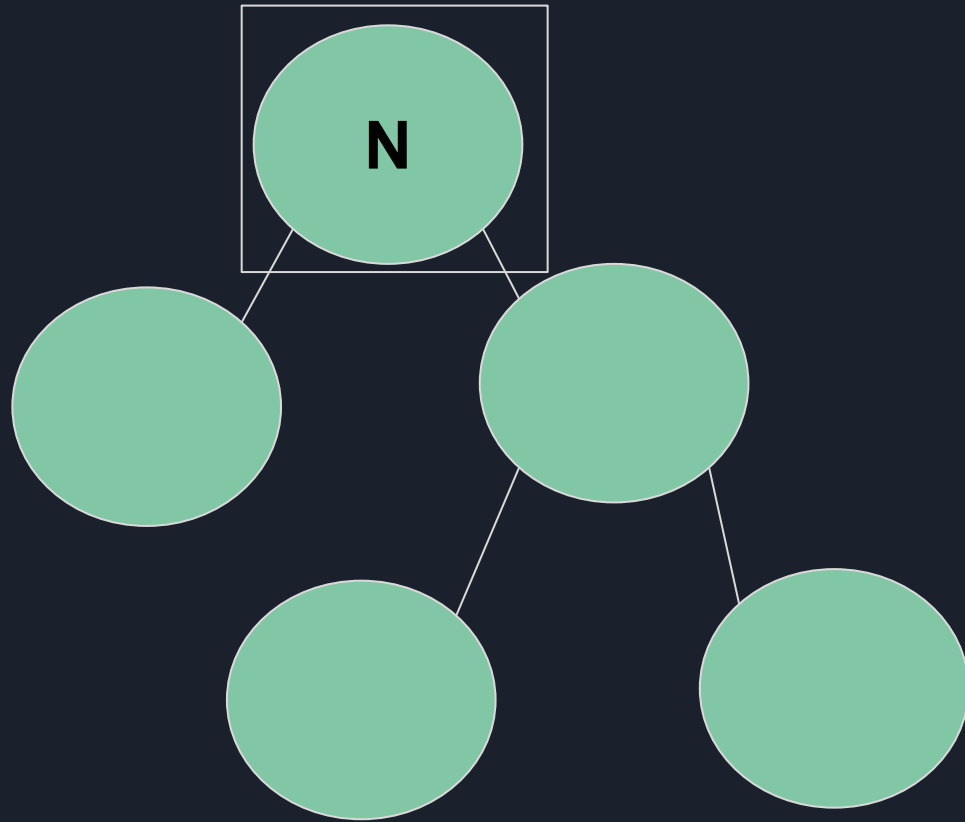


# Square Root Decomposition / Sqrt N Tricks


Reynaldo Wijaya Hendry - Universitas Indonesia





# Jenis - Jenis Square root decomposition

1. Segment Array / sqrt bucket
2. Heavy Light Vertex
3. Mo's Algorithm
4. Pending Update / sqrt bucket on query




# Segment Array / Sqrt Bucket

## Motivasi

Ada Array ukuran  $N$  dan  $Q$  query, ada 2 jenis query


1. Ubah nilai menjadi  $x$  pada range  $L$  sampai  $R$
2. Hitung nilai max pada range  $L$  sampai  $R$

$1 \leq N, Q \leq 100.000$




## Segment Array / Sqrt Bucket Motivasi

Classical Range Max Query Problem, bisa di solve pake segment tree, tapi kalo solusinya pake itu, ngapain kita bahas disini.



# Segment Array / Sqrt Bucket Motivasi


Mari kita bagi array - array itu menjadi  $K$  segmen.



# Segment Array / Sqrt Bucket Motivasi

Mari kita bagi array - array itu menjadi  $K$  segmen.

Size tiap segmen pasti  $N / K$




# Segment Array / Sqrt Bucket Motivasi

Mari kita bagi array - array itu menjadi  $K$  segmen.

Size tiap segmen pasti  $N / K$

Misalkan kita ingin mengupdate di satu segmen, kita bisa lakukan lazy pada segmen tersebut






## Segment Array / Sqrt Bucket Motivasi

Mari kita bagi array - array itu menjadi  $K$  segmen.

Size tiap segmen pasti  $N / K$

Misalkan kita ingin mengupdate di satu segmen, kita bisa lakukan lazy pada segmen tersebut

Kalo tidak sampe satu segmen? Update saja langsung




## Segment Array / Sqrt Bucket Motivasi

Mari kita bagi array - array itu menjadi  $K$  segmen.

Size tiap segmen pasti  $N / K$

Misalkan kita ingin mengupdate di satu segmen, kita bisa lakukan lazy pada segmen tersebut

Kalo tidak sampe satu segmen? Update saja langsung satu-satu



# Segment Array / Sqrt Bucket Motivasi

Coba kita bagi kasus,


Update dulu deh:

- Misal Updatenya semua maka kita tinggal update semua segment nya sehingga kompleksitasnya menjadi  $O(K)$
- Kalo Updatnya  $\text{size}(\text{segmen}) - 1$  maka kompleksitasnya menjadi  $O(N / K)$

Hal ini juga berlaku untuk Query

Maka

K optimumnya adalah?



# Segment Array / Sqrt Bucket Motivasi


Supaya K optimum kita ingin maka persamaan ini berusaha kita penuhi

$$K = N / K$$

$$K^2 = N$$

$$K = \text{sqrt}(N)$$

Maka K optimumnya adalah  $\text{sqrt}(N)$



# Segment Array / Sqrt Bucket Kode

Ga butuh kodenya la ya? Ini gampang kan

Algoritmanya kurang lebih

1. Bagi Array menjadi  $K$  bucket
2. Untuk update dan query kalo misal dia memenuhi satu bucket langsung apply saja ke bucket itu (jangan satu-satu) maksimal kompleksitas  $O(K)$
3. Kalo misal ga sampe satu bucket maka langsung aja satu-satu di update atau di query maksimal kompleksitas  $O(N / K)$

Karena  $K = \sqrt{N}$  maka keduanya menjadi  $O(\sqrt{N})$



# Simulasi Segment Array

Value	5	1	3	9	2	6	4	6	1
Indeks	1	2	3	4	5	6	7	8	9



# Simulasi Segment Array

Bagi jadi K segmen ,  $K = \text{sqrt}(9) = 3$

Value	5	1	3	9	2	6	4	6	1
Indeks	1	2	3	4	5	6	7	8	9
	5			9			6		



# Simulasi Segment Array

Update 2 - 8 , jadi 0

Value	5	0	0	9	2	6	0	0	1
Indeks	1	2	3	4	5	6	7	8	9
Val Segmen	5			0			1		
Lazy	INF			0			INF		





# Simulasi Segment Array

Update 2 - 8 , jadi 0


Value	5	0	0	9	2	6	0	0	1
Indeks	1	2	3	4	5	6	7	8	9
Val Segmen	5			0			1		
Lazy	INF			0			INF		



# Simulasi Segment Array

Query 1 - 5 = 5

Value	5	0	0	0	0	0	0	0	1
Indeks	1	2	3	4	5	6	7	8	9
Val Segmen	5			0			1		
Lazy	INF			INF			INF		



# Segment Array / Sqrt Bucket QnA

Ada Pertanyaan?



## Heavy Light Vertex Ide

- Suatu vertex itu light apabila degree vertex itu  $\leq \sqrt{N}$  selain itu heavy
- Ada berapa banyak maksimal heavy vertex?



## Heavy Light Vertex Ide

- $\text{Sqrt}(N)$ , anak TK juga tau



## Heavy Light Vertex SPOJ - DIST2

Ada graf undirected  $N$  vertex  $M$  edge, ada  $Q$  query tiap query dikasih  $U$  sama  $V$  ditanya berapa jarak terpendek dari  $U$  ke  $V$  kalo  $\geq 3$  keluarkan ga bisa, kalo  $\leq 2$  print pendeknya dan banyak jalannya

$1 \leq N, Q \leq 100.000$

$1 \leq M \leq 200.000$



# Heavy Light Vertex

## SPOJ - DIST2

Solusi :

- Jarak 0, ez
- Jarak 1, tinggal for adjlist pake binser
- Jarak 2, tinggal for salah satu adjlist lalu pake jarak 1



# Heavy Light Vertex

## SPOJ - DIST2

ITU  $O(Q * M)$ , TLE LAH





# Heavy Light Vertex

## SPOJ - DIST2

Solusi :

- Jarak 0, ez
- Jarak 1, tinggal for adjlist pake binser
- Jarak 2, tinggal for salah satu adjlist yang **degreenya lebih kecil** lalu pake jarak 1



Heavy Light Vertex  
SPOJ - DIST2

MASIH  $O(Q * M)$ , KOK  
NGOTOT SIH!



# Heavy Light Vertex

## SPOJ - DIST2

Solusi :

- Jarak 0, ez
- Jarak 1, tinggal for adjlist pake binser
- Jarak 2, tinggal for salah satu adjlist yang **degreenya lebih kecil** lalu pake jarak 1
- Kalo sudah pernah di Query di memo



Heavy Light Vertex  
SPOJ - DIST2

WOI GOBL\*K BISA NGITUNG  
BIG OH GA SIH ITU KAN  
TETAP  $O(Q * M)$

# Heavy Light Vertex

## SPOJ - DIST2

### Reynaldo Wijaya Hendry: submissions Jimmy 's Travel Plan

ID		DATE	PROBLEM	RESULT	TIME	MEM	LANG
22468872	<input checked="" type="checkbox"/>	2018-10-09 06:57:16	Jimmy 's Travel Plan	<b>accepted</b> <a href="#">edit</a> <a href="#">ideone it</a>	0.45	31M	CPP14- CLANG
22468852	<input type="checkbox"/>	2018-10-09 06:48:49	Jimmy 's Travel Plan	time limit exceeded <a href="#">edit</a> <a href="#">ideone it</a>	-	29M	CPP14- CLANG
22468851	<input type="checkbox"/>	2018-10-09 06:48:17	Jimmy 's Travel Plan	time limit exceeded <a href="#">edit</a> <a href="#">ideone it</a>	-	28M	CPP14- CLANG
22468847	<input type="checkbox"/>	2018-10-09 06:47:21	Jimmy 's Travel Plan	time limit exceeded <a href="#">edit</a> <a href="#">ideone it</a>	-	28M	CPP14
22468845	<input type="checkbox"/>	2018-10-09 06:47:14	Jimmy 's Travel Plan	time limit exceeded <a href="#">edit</a> <a href="#">ideone it</a>	-	28M	CPP14
22468837	<input type="checkbox"/>	2018-10-09 06:43:05	Jimmy 's Travel Plan	time limit exceeded <a href="#">edit</a> <a href="#">ideone it</a>	-	24M	CPP14
22468835	<input type="checkbox"/>	2018-10-09 06:41:06	Jimmy 's Travel Plan	time limit exceeded <a href="#">edit</a> <a href="#">ideone it</a>	-	24M	CPP14
22468832	<input type="checkbox"/>	2018-10-09 06:40:58	Jimmy 's Travel Plan	time limit exceeded <a href="#">edit</a> <a href="#">ideone it</a>	-	24M	CPP14
22468822	<input type="checkbox"/>	2018-10-09 06:35:59	Jimmy 's Travel Plan	time limit exceeded <a href="#">edit</a> <a href="#">ideone it</a>	-	24M	CPP14



# Heavy Light Vertex

## SPOJ - DIST2





# Heavy Light Vertex

## SPOJ - DIST2

Terlalu lebay sih tadi itu but mirip - mirip la

### Analisis

- Untuk kasus light vertex maka gampang, kompleksitasnya  $O(\sqrt{N} * \log N)$
- Untuk kasus heavy vertex, maka worst case iterasi semua edge  $O(M)$  karena heavy vertex maximal ada  $\sqrt{N}$  maka  $O(\sqrt{N} * M * \log N)$  lalu jangan lupa di memo



## Heavy Light Vertex Cara lain

Penggunaan Heavy Light Vertex ga hanya gitu, kita bisa memperlakukan Heavy vertex beda dengan light vertex

Untuk contoh lebih lengkapnya ga bakal kujelasin disini tapi kamu bisa coba soal SPOJ - MAXCHILDSUM





# Heavy Light Vertex QnA

Ada Pertanyaan?

# Mo's Algorithm



**Moo.**



# Mo's Algorithm

## Motivasi

Ada array  $N$  element, ada  $Q$  query dikasih range  $L$  sama  $R$   
hitung berapa banyak elemen yang frekuensinya  $> 5$



# Mo's Algorithm

## Naive

```
int L = 1, R = 1;
// [L, R)
for(query q : queries) {
    while(R <= q.r)    update(array[R++]);
    while(L-1 >= q.l)  update(array[--L]);
    while(R-1 > q.r)   remove(array[--R]);
    while(L < q.l)     remove(array[L++]);
    ans[q.index] = extractAnswer();
}
```



## Mo's Algorithm Idea

Kita bagi L menjadi blok - blok query berukuran  $\sqrt{N}$

Setelah itu kita sort menaik berdasarkan blok apabila sama bloknya maka berdasarkan R



# Mo's Algorithm Code

```
bool cmp(query a, query b) {  
    if(a.l / SQRT == b.l / SQRT) {  
        return a.r < b.r;  
    }  
    return a.l / SQRT < b.l / SQRT;  
}  
  
int L = 1, R = 1;  
// [L, R)  
sort(queries.begin(), queries.end(), cmp);  
for(query q : queries) {  
    while(R <= q.r)    update(array[R++]);  
    while(L-1 >= q.l)  update(array[--L]);  
    while(R-1 > q.r)    remove(array[--R]);  
    while(L < q.l)      remove(array[L++]);  
    ans[q.index] = extractAnswer();  
}
```



# Mo's Algorithm Analisis

Ada  $\sqrt{N}$  bucket untuk  $L$

Misal ada pointer  $L_{(i-1)}$  sama pointer  $R_{(i-1)}$  (setelah di sort), apabila query selanjutnya adalah  $L_i$  dan  $R_i$  maka ada 2 kasus

Apabila  $L_i$  satu bucket dengan  $L_{(i-1)}$ , maka pointer  $L_i$  bisa berjalan sebesar  $\sqrt{N}$ , karena ada  $Q$  query maka  $L_i$  bisa berjalan sebesar  $\sqrt{N} * Q$



## Mo's Algorithm Analisis

Apabila  $L_i$  tidak satu bucket dengan  $L_{(i - 1)}$  maka kita harus mengurangi counter  $R$  kembali menjadi awal, ingat pada kasus bucket yang sama maka mereka sudah sorted increasing, karena hanya ada  $\sqrt{N}$  bucket maka nilai  $R$  hanya akan diganti sebanyak  $\sqrt{N}$  kali,  $Q * \sqrt{N}$





## Mo's Algorithm Kompleksitas

Kompleksitas total jadi  $Q * \sqrt{N}$

Apakah Mo's bisa dilakukan kalo ada query update?



## Mo's Algorithm Update Mo's

Bisa, kompleksitasnya jadi  $O(N^{5/3})$  tapi tak akan dibahas disini

Buat yang mau tau bisa coba buka

<http://rwhendry.blogspot.com/2017/06/MoUpdate.html> (sekalian promosi blog)





# Mo's Algorithm QnA

Apakah ada pertanyaan?



# Pending Update / sqrt block on query

## Motivasi

### Classic Problem Range Sum Query

Ada array  $N$  elemen, ada  $Q$  query, querynya ada 2 tipe


- Ubah elemen di indeks ke  $A$  jadi  $B$
- Sum dari range  $L$  ke  $R$



## Pending Update / sqrt block on query Naive


Kalo ada Update bikin prefix sum baru

kalo query tinggal  $O(1)$  aja



## Pending Update / sqrt block on query Ide

Gimana kalo gw mager buat update terus, jadi updatenya gw simpan. Misal ada query ubah elemen di indeks ke 2 sama ke 3. Maka pas ada query misal 1 ke 5 gw cek di query - query yang gw simpan tadi ada berapa yang berpengaruh ke query gw.



## Pending Update / sqrt block on query Ide

Setelah ada  $K$  query yang gw simpan, gw build ulang aja lagi prefix sumnya

### Analisis Kompleksitas

- Tiap Query  $O(K)$
- Banyak build ulang prefix sum  $O(Q / K)$

Kok familiar ya





## Pending Update / sqrt block on query Ide

Setelah ada  $K$  query yang gw simpan, gw build ulang aja lagi prefix sumnya

### Analisis Kompleksitas

- Tiap Query  $O(K)$
- Banyak build ulang prefix sum  $O(Q / K)$

Kok familiar ya


$K$  yang memenuhi  $\sqrt{Q}$



## Pending Update/ sqrt block on query

### Source Code

```
for(query q : queries) {  
    if(q.type == "UPDATE") {  
        pendingUpdates.push_back(q);  
  
        if(pendingUpdates == PENDING_LIMIT) {  
            applyUpdates(); // fast; not dependent on PENDING_LIMIT  
        }  
    } else {  
        int ans = ansFromLastApplies;  
  
        for(query pendings : pendingUpdates) {  
            apply(ans, pendings) // applying pending update to ans  
        }  
    }  
}
```



## Pending Update / sqrt block on query Queries on Tree


[codeforces.com/gym/100589/problem/A](https://codeforces.com/gym/100589/problem/A)

Dikasih tree ukuran  $N$  sama  $M$  query

Query-nya:

1. Tambahin  $x$  ke nilai semua node dengan depth  $y$
2. Itung jumlah nilai semua node di subtree  $x$

$1 \leq N \leq 100.000$ ,  $1 \leq M \leq 10.000$



## Pending Update / sqrt block on query Queries on Tree

Precompute dulu preorder sama vector untuk tiap level, bikin segment treenya

Untuk Update tinggal tandain level level mana aja yang diubah

Untuk Query iterasi untuk setiap level ada berapa banyak nodenya yang terupdate, bisa pake binser sama preorder buat nyari node yang masuk



## Pending Update / sqrt block on query Queries on Tree

Untuk apply updatanya , ya udah bikin tree baru aja sama  
precompute precomputenya



Pending Update  
QnA

Ada Pertanyaan?



# Square Root Decomposition QnA

Ada Pertanyaan?



## Latihan Soal

Ada array isi  $N$  elemen  $x_i$ , ada  $Q$  query

Query 1 : Update nilai di indeks  $ke$  -  $a$  jadi  $b$

Query2 : dari range  $L$  sampe  $R$  ada berapa bilangan yang  $< X$





Thank You