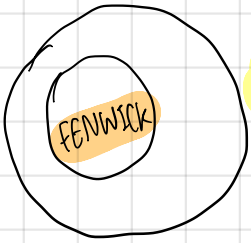
FENWICK TREE

SEGMENT



Segment

อะไรที่ Fenwick ทำได้

Segment ก็ทำได้

↳ ทำได้เยอะกว่า

FENWICK

- UPDATE ช่วงหาจุด yoyo
- UPDATE จุดหา sum ช่วง Ruby
- inversion

อย่างไดอย่างนึ่

SEGMENT

- update จุดหา min max & sum
- update ช่วงหา min max (lazy segment tree)
- update ช่วง ตาม ช่วง

FENWICK

① update ช่วงหาจุด 1 มิติ

- UPDATE a-b ค่า v

Etc update

วนตั้งแต่ a - ขนาดห้น `i+=(i&-i)`

```
for (i=a ; i<=n ; i+=(i&-i)) fw[i]+=v
```

วนลบ ตั้งแต่ b+1 - ขนาดห้น

```
for(j =b+1; j<=n ; j+=(j&-j)) fw[j]-=v
```

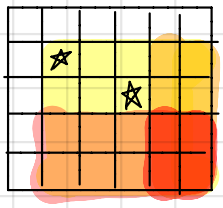
- query หา จุด c

```
for(i=c ; i>0 ; i-=(i&-i)) ans+=fw[i];
```

① update ช่วงหาจุด 2 มิติ

- update ab ถึง xy ค่า V

```
update (a,b,v);
update (a,y+1,-v);
update (x+1,b,-v);
update (x+1,y+1,v);
```



```
for ( i=ii ; i<=n ; i+=(i&-i))
    for (j =jj ; j<=n ; j+=(j&-j))
        fw[i][j]+=v;
```

- query หาช่อง ii jj

```
for(i=ii ; i>0 ; i-=(i&-i))
    for(j =jj ; j>0 ; j-=(j&-j))
        ans+=fw[i][j];
```

② update จุด ตาม ช่วง

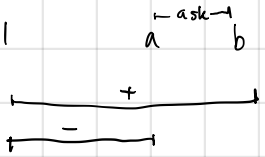
- update จุด x

update (x, v) ได้เลย

```
for (i=x ; i<=n ; i+=(i&-i))
    fw[i]+=v;
```

- query ตาม a-b

query(b) - query(a-1)




```
for(i=b;i>0;i-=(i&-i))
    ans+=fw[i];
for(i=a-1;i>0;i-=(i&-i))
    ans-=fw[i];
```

SEGMENT TREE

SEGMENT TREE

- Segment tree
- Q_Dwarf

LAZY SEGMENT TREE

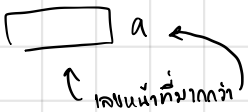
ลด big O

Nigger

Inversion

รับเลขมาทีละเลข

Inversion



เวลารับ a[i] มาเราจะไป query ดูว่าใน fenwick มีเลขมากกว่าเราที่เท่าไหร่ แล้ว update เลขเราลงไป

① query เป็นช่วง

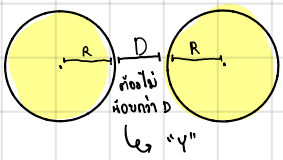
query (เลขมากสุด(ช่วงสุดท้าย))

query (ช่วงหัวถอย)

② update เลขเราลงไป

Closest pair of point $O(n \log n)$

Plantation _ TOI 14

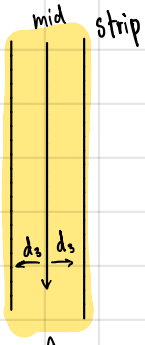


ต้องไม่น้อยกว่า D

↳ "y"

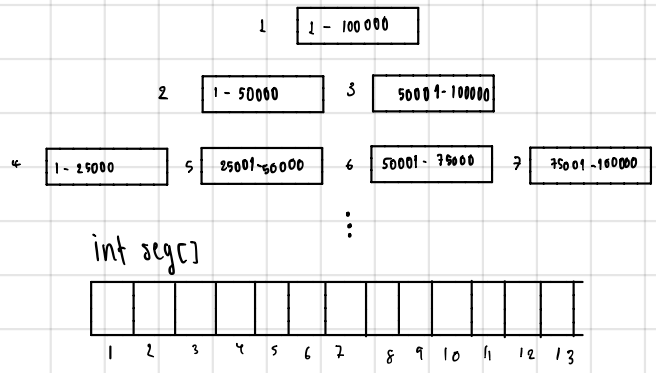
จุดศูนย์กลางห่างกันไม่น้อยกว่า $2R+D$

1. sort จุดตามแกน x
2. Divide (แบ่งตามจำนวน)
3. conquer } เหลือจุดไม่เกิน 3 จุด
   - closest (l, r)
   - mid = (l+r)>>1 ← เร็วกว่า (l+r)/2
   - d_1 = closest (l, mid)
   - d_2 = closest (mid+1, r)
   - d_3 = min(d_1, d_2)
4. sort strip ตาม y



↳ Bruce-Force check มีบางพิจารณ์ไม่เกินสี่จุด

Segment tree



int seg[]

process ของ segment tree

1. update
2. query

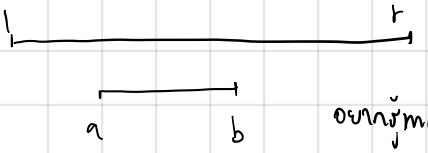
การ build / construct tree

```
void build (int l, int r, int now)
{   if(l==r){
        seg[now] = ค่าของข้อมูลนั้น(a[l])
        return ;
    }
    mid = (l+r)/2 ;
    build (l, mid, now*2);
    build (mid+1, r, now*2+1);
    seg[now] = max(seg[now*2], seg[now*2+1]);
}
```

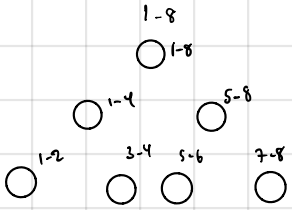
ใน main

```
build(1,n,1)
```

```
void update (int l, int r, int a, int b, int now)
{   if(l>a || r<a) return ; เช็คขอบ
    if(l==r){
        seg[now] += b; return ;
    }
    mid = (l+r)/2;
    update (l, mid, a, b, now*2)
    update (mid+1, r, a, b, now*2+1)
    seg[now] = max(seg[now*2], seg[now*2+1])
}
```



ตอบคำถาม max จาก a ถึง b (query)




```
if(l>a || r<b)
    return -1e9;
if(l==r)
    return seg[now];
else
    printf("%d\n",
query(1,n,a,b,1);
```