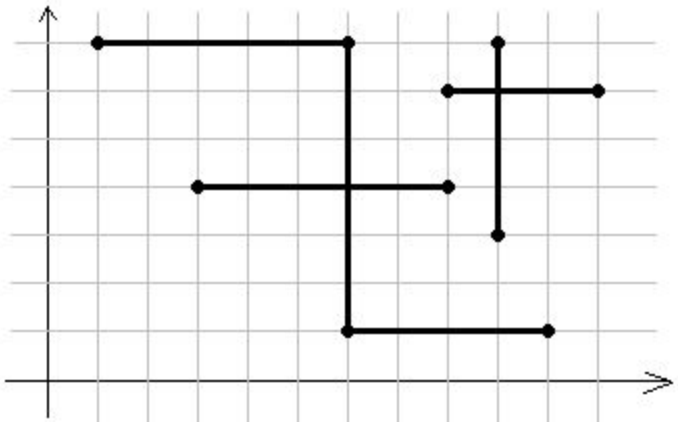
Sweep

Sweep, odnosno sweeping line pristup rješavanju problema je pristup u kojem se problem rješava tako da se zamisli nekakva linija koja se kreće s jedne strane na drugu "skenirajući" zadani problem(nerijetki su i problemi u kojima se ta linija vrti u krug).
Promatrat ćemo ovaj pristup na konkretnom zadatku:

Zadano je N (1<=N<=100000) dužina u ravnini. Dužine su paralelne s koordinatnim osima, dakle ili su horizontalnom ili u vertikalnom položaju. Dužine su zadane tako da se nijedne dvije dužine istog tipa (dvije horizontalne ili dvije vertikalne) ne diraju.
Izračunajte koliko ima parova dužina koje se sijeku. Koordinate su [1..100000]



Input za gornji slučaj je:
6
1 7 6 7
3 4 8 4
6 1 10 1
8 6 11 6
6 1 6 7
9 3 9 7
a output je
4

Ajmo odmah na početku definirati strukturu duzina:

```
struct duzina {
   int x1, x2, y1, y2;

   duzina() {
      scanf( "%d%d%d%d", &x1, &y1, &x2, &y2 );
      if( x1 > x2 ) swap( x1, x2 );
      if( y1 > y2 ) swap( y1, y2 );
   }
};
```

Ovdje je sve jasno… napravili smo strukturu koja u konstruktoru učitava podatke s inputa.

Dakle, već smo rekli da ćemo rješavati sweepanjem… Neka ta zamišljena linija ide s lijeva na desno. Međutim, kako nam konkretno zamišljanje te linije pomaže i kako ćemo simulirati njeno kretanje?

Ta linija svojim kretanjem trigerira (pokreće?) nekakve događaje. I sve se zapravo svodi na te događaje. Cijeli algoritam treba definirati preko tih događaja tako da se definira što se radi kod kojeg događaja. I kad to napravimo biti će jasno da je algoritam korektan.

U našem zadatku ta skenirajuća linija u svojem životu može doživjeti 3 različita tipa događaja.

1) Naletjela je na početak (lijevu točku) horizontalne dužine
2) Naletjela je na vertikalnu dužinu
3) Naletjela je na kraj (desnu točku) horizontalne dužine

Prvo… Trebati će nam struktura event koja ima tri varijable where, type, who. Više-manje u svakom zadatku gdje se koriste eventi oni se mogu opisati s te tri varijable. Ovdje nam where govori x-koordinatu događaja, type nam govori tip događaja (1, 2 ili 3 kako smo gore rekli), a who nam govori o kojoj se dužini radi.

```
struct event {
   int where, type, who;
   event( int a, int b, int c ) { where=a; type=b; who=c; }
};
bool operator < ( const event &A, const event &B ) {
   if( A.where != B.where ) return A.where < B.where;
   if( A.type != B.type ) return A.type < B.type;
   return A.who < B.who;
}
```

Definirali smo i operator < koji nam treba da bi sortirali te događaje (jer ih moramo obrađivati s lijeva na desno redom kojim je i zamišljena linija naletjela na njih)

I sad jedino što nam je preostalo je da definiramo algoritam preko tih događaja.

1) Početak horizontalne dužine – dodamo tu dužinu u nekakvu strukturu
2) Vertikalna dužina – zanima nas koliko ima dužina koje se sjeku s tom vertikalnom dužinom. Radimo nekakav query na strukturi koju smo odabrali da bi saznali taj broj
3) Kraj horizontalne dužine – mičemo dužinu iz strukture

main dakle ovako izgleda:

```
int main( void ) {
   int n;
   scanf( "%d", &n );

   vector<duzina> D;
   vector<event> E;
   for( int i = 0; i < n; ++i ) {
      duzina A;
      D.push_back( A );

      if( A.y1 == A.y2 ) { // horizontalna
         E.push_back( event( A.x1, 1, i ) );
         E.push_back( event( A.x2, 3, i ) );
      } else { // vertikalna
         E.push_back( event( A.x1, 2, i ) );
      }
   }

   int ret = 0;
   sort( E.begin(), E.end() );
   for( vector<event>::iterator it = E.begin(); it != E.end(); ++it ) {
      if( it->type == 1 ) {
         // Dodaj horizontalnu duzinu D[it->who] u neku strukturu podataka
      }
      if( it->type == 2 ) {
         // Nadji koliko ima duzina kojima je y koordinata izmedju
         // D[it->who].y1 i D[it->who].y2
      }
      if( it->type == 3 ) {
         // Makni horizontalnu duzinu D[it->who] iz strukture
      }
   }
   printf( "%d\n", ret );

   return 0;
}
```

Još jedino moramo odabrati strukturu… Hmm.. bas se pitam koja to struktura daje odgovor na pitanje koliko ima brojeva iz intervala [a,b]

Još nešto… linija može u nekom trenutku naletjeti na hrpu događaja (svi na istoj x-koordinati). Dakle, nisam slučajno numerirao događaje s 1, 2 i 3 na baš taj način jer je bitno da napravimo query nakon što smo ubacili sve horizontalne dužine na čiji početak smo naletjeli, a prije nego izbacimo one na čiji smo kraj naletjeli.

Još nešto #2… gornji source ima bug ☺

Zaključak: Ova tehnika rješavanja se relativno često koristi i ne samo u geometrijskim problemima. Ja sam osobno fakat puno puta pisao strukturu event. Algoritmi definirani tako preko događaja su vrlo jasni i lagano se uvjeriti da je rješenje valjano, a i lakše je debugirati i općenito provjeriti je li kod ispravan. Uostalom, možete vidjet da sweeping line koriste i u

star-trekolikim filmovima kad skeniraju neki brod. Ne koriste je zato jer je kul nego jer im tako radi algoritam... ☺

Zadaci:

1) Završite gornji program.

2) Smatramo da se dužine sjeku samo ako od točke sjecišta postoje dužine na sve četiri strane. Odnosno ako se sjecište dužina nalazi u jednoj od točaka jedne od dužina smatramo da se dužine ne sjeku. Dakle u gornjem primjeru imamo samo 2 sjecišta.

3) Kao i 1) ali koordinate su double a ne integeri.

Nakon toga imate još tri zadatka u atačmentu.

Nakon toga možete u kormenu pročitati “Determining whether any pair of segment intersects”…

Šaljite kako što riješite. Kad ste gotovi javite se za još zadataka.

Luka Kalinovčić