S.82

Fonyó Viktória

Keszthely, Vajda J. Gimn., 12. o. t.

fonyoviki@gmail.com

Fordítás és tesztelés

A program Code::Blocks 12.11-es fejlesztői környezettel készült, g++ 4.7.1 fordítóval fordítottam. Futtatáskor alapértelmezetten a program a standard inputról várja az adatokat, és az eredményt a standard outputra írja ki. Azonban, ha megadunk 1 vagy 2 parancssori paramétert, akkor lehetőség van az inputot az első paraméterben megadott fájlból beolvasni, illetve az eredményt a második paraméterben megadott fájlba írni. Ez utóbbi lényegesen gyorsít a program futásán, a legnagyobb bemenet esetén is a kívánt határidőn belül van. Azonban standard outputra kiíratás esetén a kiírás része nagyon sokáig tart.

Rövid ismertető

A megoldás során külön fogjuk vizsgálni a bástyák sor, és oszlop szerinti elhelyezkedését, azaz külön fogjuk meghatározni az egyes bástyák x és y koordinátáját. Mivel megadták a téglalapok kezdetét és végét x és y koordinátákra nézve, ezért tekinthetünk rájuk intervallumokként. Csinálunk az x és y intervallumoknak 1-1 vektort, amiben eltároljuk az adatokat beolvasás során. Eztán rendezzük az elemeket aszerint, hogy az intervallumok minimuma mennyi. Majd bepakoljuk az 1-essel kezdődő intervallumokat egy prioritásos sorba. Kiszedjük a legnagyobb elemet, amely az lesz, amelyik intervallumnak leghamarabb vége van, azaz amelyiknek a legkevesebb a maximuma. Ha ez megvan, elmentjük a legnagyobb elemet és töröljük. Ezután beletesszük a sorba a 2-essel kezdődő intervallumokat, majd kiválasztjuk a legnagyobb elemet, elmentjük és töröljük. Ez így halad addig, míg a végéhez nem érünk, vagy hibát nem találunk. Ezt megcsináljuk külön az x és az y koordináta-intervallumokra, majd kiíratjuk a megoldást.

A program működése

Először csinálunk két struktúrát. Az egyikben az x koordináta minimumát, maximumát tároljuk, továbbá a bástya sorszámát, a másikban az y koordináta ezen adatait. Ezeken belül csinálunk egy olyan függvényt (ami később a prioritásos sorhoz kell), amely visszaadja a korábban befejeződő intervallumú, azaz kisebb maximummal rendelkező bástyát.
Utána két függvényt csinálunk, mely a rendezéshez kell majd, az egyik az x-, a másik az y-koordinátákhoz. Mindkettő a kisebb minimummal rendelkező bástyát adja vissza értékül.

Ezután beolvassuk N-t, majd csinálunk 2 N-hosszú vektort, „vx”-et és „vy”-t, melyekbe beolvassuk az adatokat.
Felveszünk egy „lehet” nevű kezdetben igaz logikai változót, mellyel azt fogjuk nyomon követni, hogy az egyes intervallumokból biztosan olyan elemet választott-e ki a program, amely benne van.
Létrehozunk az x és y intervallumoknak 1-1 prioritásos sort.
Ezután rendezzük a „vx” és „vy” vektor elemeit a minimum értékük szerint.
Csinálunk két vektort, „xkord” és „ykord” néven, amelyekbe a bástyák végleges elhelyezésének koordinátáit fogjuk menteni.
Ezután egy számláló ciklusban először beletesszük az „pqx” prioritásos sorba az összes olyan elemet, melynek minimuma i. Amennyiben a prioritásos sor üres, a hibát a „lehet” logikai változóban elmenjük, ha pedig nem, megnézzük, hogy mi a prioritásos sor legnagyobb eleme (a már korábban megírt segédfüggvény segítségével), elmentjük, majd töröljük. Ugyanezt megcsináljuk az y-koordináták intervallumaival a „pqy” prioritásos sorban, a legnagyobb elemet elmentjük és töröljük a sorból. Közben, ha a prioritásos sorok legnagyobb elemének maximuma nem éri el i értékét, azaz egy olyan helyre szeretnénk tenni a bástyát, ami a téglalapon kívül esik, a „lehet” logikai változónk hamis értéket vesz fel és leáll a ciklus.
Végül a programunk megadja a végeredményt.
A program futásideje $O(N \cdot \log N)$ lesz, mivel az általunk használt rendezés a minimumok esetében $O(N \cdot \log N)$ futásidő, és a prioritásos sort használó rész is $O(N \cdot \log N)$ idő alatt lefut, hiszen minden egyes beszúrás és törlés ideje $O(\log N)$, és minden egyes intervallumot legfeljebb egyszer szúrunk be, illetve törlünk.

A megvalósítás során

- használjuk a rendezéshez az STL algorithm csomag „sort” rendezési algoritmusát az „xkisebb” és „ykisebb” függvényekkel
- használjuk az STL priority_queue adatszerkezetét a BastyaX és BastyaY struktúrával, a prioritásos sor használatához. Ehhez definiálnunk kell a BastyaX és BastyaY struktúrákon a < operátort. Mivel a top() a legnagyobb elemet adja vissza, és nekünk a legkorábban végződő intervallumra van szükségünk, ezért a max adattag szerint kell fordítottan rendezni.
- a fájlok beolvasásához az STL fstream csomag kell
- a vektorok használatához az STL vector csomag kell