NYELVFÜGGETLEN PROGRAMOZÁS

4. forduló

A kategória támogatója: SAP Hungary Kft.

Ismertető a feladathoz

A 4. forduló után elérhetőek lesznek a helyezések %-os formában: azaz kiderül, hogy a kategóriában a versenyzők TOP 20% - 40% -60% -ához tartozol-e!

Szeretnénk rá felhívni figyelmedet, hogy a játék nem Forma-1-es verseny! Ha a gyorsaságod miatt kilököd a rendesen haladó versenyzőket, kizárást vonhat maga után!

A feladatlap több csatolmányt is tartalmaz, ezért a megoldását asztali gépen javasoljuk!

- Minden fordulóban két algoritmikus feladat lesz.
- Minden feladat esetében 5 "éles" inputra kell előállítanod az outputokat, amelyeket aztán a versenyfelületen a megfelelő szövegmezőbe kell illesztened.
- A megoldásod ellenőrzését segítendő, minden feladathoz tartozik 2 db példa input és output.
- Pl. ha egy feladat címe "Cica", akkor a cica.peldaX.in.txt-ben lesz a példa input, a cica.peldaX.out.txt-ben pedig az ehhez tartozó példa output (X egy egész szám). A cicaX.in.txt fájlokban lesznek a pontokért megoldandó inputok, ahol X: 1..5.
- Mindezeket a txt fájlokat a csatolt tömörített archívum tartalmazza, melyet a feladatsor indítása után tölthetsz le.
- A megoldásokat bármilyen programnyelven elkészítheted.
- A forráskódot nem kell beküldeni, csak az outputokat.

Jó szórakozást!

Felhasznált idő: 40:00/40:00 **Elért pontszám:** 0/10

Indítás utáni csatolmányok

1. feladat 0/1 pont

Manhatten

Mint tudjuk, Manhattan lakói egy mindkét irányban végtelen négyzetrácson, azaz egész koordinátájú helyeken élnek. Szeretnénk egy fagyizót nyitni Manhattanben, amely tetszőleges olyan rácspontra kerülhet, ahol még nem lakik senki. A forgalom maximalizálása érdekében ezek közül a szabad rácspontok közül egy "súlyponti" helyre akarjuk tenni a fagyizót, azaz olyan rácspontra, melynek az egyes lakosoktól mért Manhattan-távolságainak összege minimális.

Az (x1, y1) és (x2, y2) rácspontok Manhattan-távolsága: abs(x1 - x2) + abs(y1 - y2).

A bemeneti fájl N sorból áll, ahol N a lakosok száma. Mindegyik sorban két egész szám található, szóközzel elválasztva, mégpedig egy lakó házának x és y koordinátája.

A kimenet 2 db egész szám legyen, szóközzel elválasztva: 1) a minimálisan elérhető össz-Manhattan-távolság, valamint 2) a minimumhelyek száma, azaz hogy a még nem foglalt rácspontok közül hány adja a minimális össztávolságot. A második szám tehát azt adja meg, hogy hányféle szabad helyre rakhatjuk a fagyizót úgy, hogy a Manhattan-távolságösszeg minimális legyen. A második szám tehát sosem lehet 0. Pontszám csak akkor jár, ha mindkét érték helyes.

Mi a **manhattan1.in.txt** kimenete?

Válasz

A helyes válasz:

445 51

Magyarázat

Vegyük észre, hogy a célfüggvény f(x) + g(y) alakú, tehát külön-külön lehet x-re ill. y-ra minimalizálni, és azok a pontok lesznek optimálisak, melyeknek x és y koordinátája is egyszerre optimális. Tehát az egydimenziós Manhattan-feladatot kell csak megoldani, itt pedig könnyen belátható, hogy a medián lesz az optimum, vagy, ha páros sok pont van, akkor a két mediánpont közötti szakasz (a pontokat is beleértve). A k-dimenziós Manhattan-probléma optimumhelyeinek a halmaza tehát egy k-dimenziós téglatest. Arra is figyelni kell, hogy ezek közül az optimális pontok közül csak a még nem foglaltakat kell megszámolni. Gond csak akkor van, ha ezek a helyek mind foglaltak, mert ilyenkor az eggyel rosszabb pontokat kell választani, és ha azok is mind foglaltak, akkor a pontosan kettővel rosszabbakat (ha vannak ilyenek), stb. Ilyen teszteset azonban az egyszerűség kedvéért nem volt feladva.

```
#!/usr/bin/env python3

def solveFile(fn:str, fOut):
 points = []
 with open(fn) as f:
  for line in f:
   x, y = map(int, line.strip().split())
   points.append((x, y))

 assert points
 n = len(points)

 points.sort(key = lambda xy: xy[0])
 xMin = (points[n // 2] if n % 2 == 1 else points[n // 2 - 1])[0]
 xMax1 = points[n // 2][0] + 1

 points.sort(key = lambda xy: xy[1])
 yMin = (points[n // 2] if n % 2 == 1 else points[n // 2 - 1])[1]
 yMax1 = points[n // 2][1] + 1

 nBest = (xMax1 - xMin) * (yMax1 - yMin)
 for x, y in points:
  if (xMin <= x < xMax1) and (yMin <= y < yMax1):
   nBest -= 1

 # Ha minden jó hely foglalt, akkor szomszédos pixeleket kell nézni, azaz az eggyel rosszabb távolságúakat.
 # Ha nem brute force akarjuk megoldani, akkor kicsit bonyolult ez az eset, mert pl. mi van, ha több pixel is
 # Így ilyen tesztesetet nem adtunk fel.
 assert nBest

 bestDistSum = 0
 for x, y in points:
  bestDistSum += abs(x - xMin) + abs(y - yMin)
```

```
  # Ellenőrzés: pl. (xMax1 - 1, yMax1 - 1) is optimális kell, hogy legyen.
  bestDistSum2 = 0
  for x, y in points:
   bestDistSum2 += abs(x - (xMax1-1)) + abs(y - (yMax1-1))
  assert bestDistSum2 == bestDistSum

  result = "%s %s" % (bestDistSum, nBest)
  message = "Output for %s: %s" % (fn, result)
  print(message)
  fOut.write(message+"\n")

  if "pelda" in fn:
   fnPeldaOut = fn.replace(".in.", ".out.")
   assert fnPeldaOut != fn
   with open(fnPeldaOut, "w") as fPeldaOut:
    fPeldaOut.write(str(result))

def main():
 with open("out.txt", "w") as fOut:
  for i in range(1, 6):
   solveFile("manhattan%s.in.txt" % (i,), fOut)
  for i in range(1, 3):
   solveFile("manhattan.pelda%s.in.txt" % (i,), fOut)

if __name__ == "__main__":
 main()
```

C#-ban egy lelkes versenyző kétféle megoldása: NyelvfuggetlenTopic

2. feladat 0/1 pont

Mi **manhattan2.in.txt** kimenete?

Válaszok

A helyes válasz:

13783 1

13783, 1

Magyarázat

ld.fent

3. feladat 0/2 pont

Mi **manhattan3.in.txt** kimenete?

Válasz

A helyes válasz:

1008551 2052

Magyarázat

ld. fent

4. feladat 0/3 pont

Mi **manhattan4.in.txt** kimenete?

Válasz

A helyes válasz:

501613927 7

Magyarázat

ld. fent

5. feladat 0/3 pont

Mi **manhattan5.in.txt** kimenete?

Válaszok

A helyes válasz:

500042640219 1

500042640219, 1

Magyarázat

ld.fent

Megjelenés

Világos