Hírvivők csoportosítása

Innen: Algowiki

Tartalomjegyzék

Feladat

A feladat (https://www.oktatas.hu/pub_bin/dload/kozoktatas/tanulmanyi_versenyek/oktv/oktv2019_2020_donto/info2_flap_d_oktv_1920.pdf#page=2) szerint adott $1 \leq H \leq 1000$ pont egy $N \times M$-es rácson. ($1 \leq N, M \leq 10000$) Vegyük a pontok összekötésével kapott teljes gráfot, ahol az egyes élek hosszán a két végpontjuk Manhattan távolságát értjük. (Manhattan távolság: $|a_x - b_x| + |a_y - b_y|$) A pontokat csoportokba soroljuk úgy, hogy minden így kapott csoport bármely két pontja között létezik olyan út, amely minden éle legfeljebb $0 \leq L \leq (N-1) + (M-1)$ hosszú. A feladat megadja az így képzett csoportok maximális $1 \leq K \leq H$ számát, mi pedig keressük a legkisebb L-et, amelyre maximum K csoportot lehet képezni.

Megoldás

1. ötlet

Próbáljuk meg minden egyes lehetséges L-re megszámolni a csoportokat és megkeresni a legnagyobb megfelelőL-t. Adott L-re szélességi vagy mélységi bejárást alkalmazva meghatározhatjuk az innen maximum L hosszúságú éleken keresztül elérhető pontokat, ezzel egy-egy összefüggő komponenst megtalálva. Ezek a komponensek a keresett csoportok. Egy ilyen bejárás költsége $O(V+E) = O(H+H^2)$. Ez akár 1001000 lépés is lehet! Nyilvánvaló ha ezt minden lehetséges L-re meg akarjuk tenni az túl hosszú időbe kerülne.

2. ötlet

Fontos észrevétel: ha $L_1 < L_2$, akkor L_2 esetén a komponensek száma biztosan nem nagyobb L_1 komponenseinek számánál.

Másképp: ha az L-hez tartozó komponensek száma $f(L)$, akkor az f függvény **monoton csökkenő**. Ez lehetővé teszi, hogy L-t bináris kereséssel (https://wiki.prog.hu/wiki/Logaritmikus_keres%C3%A9s_(algoritmus)) határozzuk meg. A keresés kezdetén az intervallum bal széle 0, hiszen lehet, hogy egy élt se kell felhasználni. Jobb széle $N-1+M-1$, ami maximum 19998. Minden egyes lépésben az intervallum középső elemére meghatározzuk a komponensek számát. Ha ez nagyobb K-nál akkor a középső elemet megelőző lesz az új jobb oldal. Ellenkező esetben a bal oldal a középső elem lesz. Ezt addig fojtatjuk, amíg csak egy lehetséges L marad. Ez az L a feladat **megoldása**.

Futásidő

A keresés $O(\log_2(N+M))$ lépésből áll, minden egyes lépésben végrehajtunk egy $O(H+H^2)$ lépésből álló gráfbejárást.

Összesen: $O(\log_2(N+M)\cdot(H+H^2))$

Alternatív megoldás

Futtassuk a gráfon a Kruskal algoritmust!

Az éleket növekvő sorrendbe rendezzük, majd egyesével hozzáadjuk a gráfhoz (Kezdetben a gráfban nincs él). A feldolgozást addig folytatjuk, amíg a komponensek száma K alá nem csökken. Ekkor az utolsó hozzáadott él hossza a megoldás.

Futásidő

Az élek rendezése: $O(H^2\cdot\log_2(H^2))$

Egy él hozzáadása a gráfhoz Unió-holvan adatszerkezetet használva: $\approx O(1)$

Összesen: $O(H^2\cdot\log_2(H^2))$

Implementáció

Az első megoldás C++ implementációja: [becsuk]

```cpp
#include <algorithm>
#include <queue>
#include <math.h>
#include <iostream>

int megold(const int n, const int m, const int h, const int k, const std::pair<int,int>* const poz){
    int* utolso = new int[h];
    std::fill(utolso, utolso+h, -1);

    int proba = 0;
    int bal = 0, jobb = n+m-2;
    while(bal < jobb){
        int kozep = (bal+jobb)/2;


        int db = 0;
        ++proba;
        for(int i = 0; i < h; ++i){
            if(utolso[i] == proba)
                continue;

            ++db;
            std::queue<int> sor;
            sor.push(i);
            utolso[i] = proba;
            while(sor.size() > 0){
                int cur = sor.front();

                sor.pop();

                for(int j = 0; j < h; ++j){
                    int tav = std::abs(poz[cur].first-poz[j].first)+std::abs(poz[cur].second-poz[j].second);
                    if(utolso[j] < utolso[cur] && tav <= kozep){
                        utolso[j] = proba;
                        sor.push(j);
                    }
                }
            }
        }


```

```
42          if(db > k){
43              bal = kozep + 1;
44          }else{
45              jobb = kozep;
46          }
47      }
48
49      delete[] utolso;
50      return bal;
51  }
52
53
54  int main(){
55      int N, M, H, K;
56      std::cin>>N>>M>>H>>K;
57      std::pair<int,int>* poz = new std::pair<int,int>[H];
58      for(int i = 0; i < H; ++i){
59          std::cin>>poz[i].first>>poz[i].second;
60      }
61
62      std::cout<<megold(N,M,H,K,poz)<<"\n";
63
64      delete[] poz;
65      return 0;
66  }
```

A lap eredeti címe: „https://algowiki.miraheze.org/w/index.php?title=Hírvivők_csoportosítása&oldid=1283”