Korszerű számítógépes módszerek a fizikában 1.

Legrövidebb út algoritmus

Marx Pál Fülöp

2021. 04. 26.

1. Feladat

A feladat a legrövidebb út algoritmus implementálása volt, azaz egy olyan program írása ami egy gráfban megkeresi a legrövidebb utat A és B csúcsok között. Ehhez a Dijkstra algoritmust kellett C++ programnyelvre ültetni.

2. A Dijkstra algoritmus általánosan

A Dijkstra algoritmus súlyozott gráfokban keresi a legrövidebb utakat egy csúcsból kiindulva. Az algoritmus futása közben nyilvántartja a gráf minden csúcsára az adott csúcs és a kiindulási csúcs között eddig megtalált legrövidebb utat. Ez kezdetben nulla a kiindulási csúcsra, és végtelen a gráf összes többi csúcsára. Az algoritmus futása közben a csúcsokat két halmazba sorolja. Az egyik halmazban (S), vannak azok a csúcsok amikhez az algoritmus már megtalálta a legrövidebb utat, a másikban (Q) a gráf többi csúcsa. Az algoritmus egy lépésében kiválasztjuk a második (Q) halmaz azon elemét (N csúcsot) amelyik a legkissebb távolságra van a kiindulási csúcstól. Ezt a csúcsot áttesszük az S halmazba, és közben ellenőrizzük az összes szomszédjára, hogy a hozzá vezető legrövidebb útnál rövidebb-e, ha N-be elmegyünk a legrövidebb úton, aztán az őket összekötő élen lépünk egyet tovább. Ha az így kapott út rövidebb, akkor ezt mentjük el az adott csúcsba vezető eddigi legrövidebb útként. Az algoritmus ezt addig ismétli, amíg a Q halmaz üressé nem válik, azaz meg nem találja a gráf összes csúcsába vezető legrövidebb utat.

3. Implementáció

A projekthez két kódot írtam. A lényeget tartalmazó "g_project.cpp"-t és a bemenetet legeneráló "g_proj_gen.cpp"-t.

3.1. Bemenet

Bemenetként egy 100 csúcsot és 1000 élt tartalmazó gráfot generáltam. A program random kisorsol egy kiindulási és egy vég csúcsot, melyeket a csúcsok és élek számával együtt a majdani bemeneti fájl első sorába ír. Ezután legenerálja az éleket a gráfba. Random választ két csúcsot, és egy súlyt a köztük lévő élnek, és ezt a fájl következő sorába írja, majd ezt ismétli meg újra és újra, míg le nem generálta az összes élt. Ezzel a módszerrel nem védtük ki sem a hurokélek sem a többszörös élek megjelenését a gráfban, és azt sem garantáltuk, hogy a gráf összefüggő, hiszen a valóságban is előfordulhat hogy két földrajzi pont között nincs út (például autóval nem lehet eljutni Hawaiiról Budapestre), vagy esetleg több út is van. Az sem kizárható, hogy egy úton elindulva ugyanoda érjünk vissza ahonnan elindultunk, így ezeket az eseteket az algoritmusnak tudni kell kezelni.

3.2. Az algoritmus

A projekt lényegi részét megvalósító kód két függvényből áll. Az első függvény maga a Dijkstra algoritmus, a második a main függvény.

3.2.1. Dijkstra algoritmus

A dijkstra függvény bemenetként a kiindulási és a végcsúcsot várja, valamint a gráf éleit éllista formájában. A Dijkstra algoritmust a program a fent ismertetettől kicsit eltérően valósítja meg. Ahelyett, hogy két halmazra osztaná a csúcsokat, és így két vectornak foglalná a memóriát, csak azokat a csúcsokat tárolja, ahol már járt, azaz amikhez tud a végtelentől különböző úthosszt rendelni, ezt "gray" azaz szürke néven használtam a kódban. Illetve, a legrövidebb utak felépítéséhez egy vectorban letárolja azt is, hogy melyik csúcsba melyik csúcsból lépünk, ha a legrövidebb úton érkezünk meg. Kezdetben a gray csak a függvény által bekért kiindulási csúcsot tartalmazza, majd minden lépésben az alábbiakat hajtja végre:

1. Megkeresi a gray halmaz azon elemét amelyik a legközelebb van a kiindulási csúcshoz
2. Törli ezt a csúcsot a gray halmazból
3. Végignézi a törölt csúcs összes szomszédját
4. Ha adott szomszéd még nem eleme a gray halmaznak, akkor hozzáadja a halmazhoz

   A legrövidebb utat úgy számolva, hogy a törölt csúcshoz tartozó úthoz hozzáadja a kettő közti él súlyát

   És az utakat tároló vectort is frissíti
5. Ha adott szomszéd eleme a gray halmaznak, akkor ellenőrzi, hogy az újonnan talált út rövidebb-e a korábban már megtaláltnál

   Ha igen, akkor felülírja a legrövidebb utat és annak hosszát

   Ha nem, akkor nem kell foglalkozni az adott szomszéddal

A fentieket addig ismételi a program amíg a gray halmaz teljesen ki nem ürül. Ez nem jelenti azt, hogy a gráf minden élét megvizsgálta, hiszen a gráf nem biztos, hogy összefüggő, de amelyik csúcs elérhető a kiindulási csúcsból az élek mentén haladva, azt megtalálja.

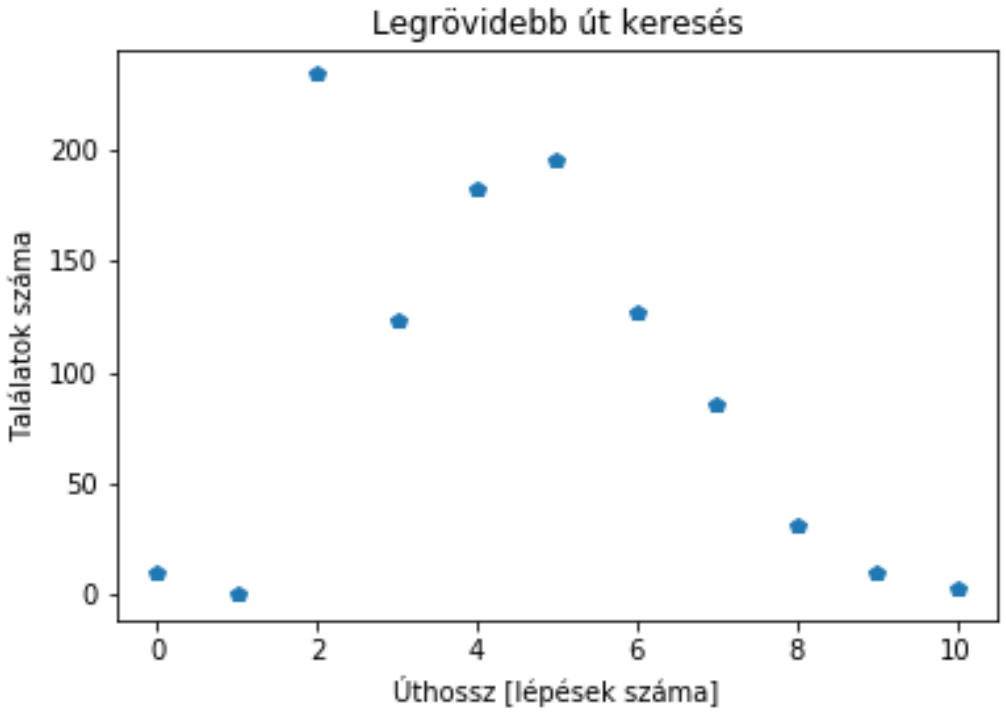
Miután a dijkstra függvény megtalálta az összes megtalálható legrövidebb utat, az elmentett "melyik csúcsból érkeztünk" adatokból felépíti a legrövidebb utat. Ha nincs út a két megadott csúcs között akkor egy üres úttal tér vissza a függvény, egyébként a csúcsok felsorolásával.

3.2.2. A main függvény

A main függvény, miután megnyitotta a bemeneti fájlt beolvassa annak első sorát, melyből megtudja, nem csak a kiindulási és végcsúcs azonosítóját, de a gráf csúcsainak és éleinek számát is. Ezután a bemeneti fájl többi sorát beolvasva a gráf éleit éllistába menti. Az éllistát úgy építi fel, hogy az éllista-vector i-edik eleme az a vector amelyben az i-ből induló éleket tároljuk. Ebben a vectorban pair típusban van eltárolva, hogy az adott él hova megy, és mekkora a súlya. Ezután meghívja a dijkstra függvényt a beolvasott kiindulási és végcsúcsot, illetve a felépített éllistát megadva bemenetként, és a visszatérési értéket kiírja egy fájlba.

4. Eredmény

A fent ismertetett programot lefuttattam ezerszer egymás után véletlenszerűen választott kiindulási és végcsúcsokkal, hogy megnézzem, átlagosan hány lépés egy legrövidebb út, ahol egy lépésnek azt vettem, amikor egyik csúcsból egy él mentén egy vele szomszédosba lépek (tehát ennél a vizsgálatnál az élek súlyát nem vettem külön figyelembe, nem is írtam ki a kimeneti fájlba). Átlagos lépésszámnak 4.26-ot kaptam, és pythonnal ábrázolva a különböző lépésszámok előfordulásait az alábbi ábrát készítettem:



A lépések száma tehát egyszer sem haladta meg a 10-et, és egyetlen kiugró értéket leszámítva harang szerű görbét követ. Ezentúl azt is észrevehetjük, hogy a 0 hosszú utak száma nem 0, azaz feltehető hogy volt olyan eset amikor az algoritmus nem talált utat a

megadott két csúcs között (az sem lehetetlen, hogy a kiindulási és a végcsúcs megegyezett, de ilyen gyakorisággal nem kellene előforduljon).

5. Diszkusszió

A projekt során a Dijkstra algoritmust implementáltam súlyozott, irányított gráfokra, kezelve a hurokéleket, a többszörös éleket és azt ha a gráf nem összefüggő, hogy valós útkeresésekhez hasonlítson. Megállapítottam, hogy valóban előfordul, hogy a gráf nem összefüggő, illetve azt is, hogy az ilyen gráfokban a csúcsok számához képest nagyságrendekkel kissebb a legrövidebb utak hossza. Mindeközben megtanultam az éllisták kezelését, és a pair típus használatát, amik más problémák kapcsán is hasznomra válhatnak.

Bibliography

[1] https://hu.wikipedia.org/wiki/Dijkstra-algoritmus