**A képen szöveg látható

Automatikusan generált leírásÚtvonalkeresést vizualizáló program**

A képen térkép látható

Automatikusan generált leírásgráf algoritmusok használatával

matematika – I. kategória

*connect the dots* csapat:

* Tóth Ábel – 10/7.évfolyam
* Buzási Áron – 10/7.évfolyam
* Pálszabó Dóra – 10/7.évfolyam

Alternatív Közgazdasági Gimnázium

felkészítő tanár: Birloni Szilvia

[Kekszilva@gmail.com](mailto:Kekszilva@gmail.com) - [p5dori@gmail.com](mailto:p5dori@gmail.com)

HASZNÁLATA – mire jó?

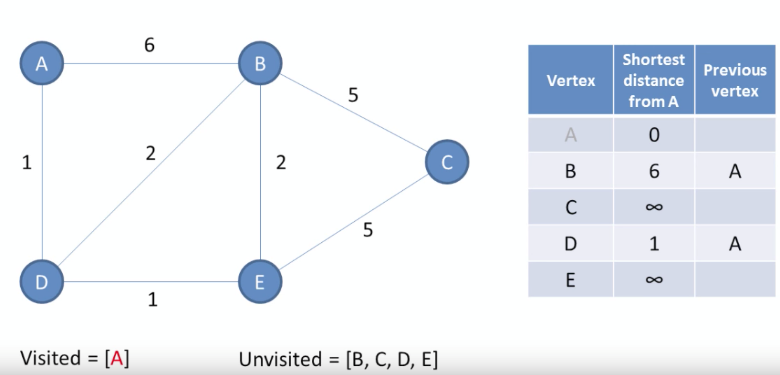
Térképek. Ezt a programot főleg két pont közötti legrövidebb távolság megjelenítésére használhatjuk, viszont adhatunk bele egy csavart is. Tegyük fel, hogy egy kereskedelmi célra szolgáló eper szállítmányt szeretnénk elvitetni Spanyolországból Magyarországra. Ha a lehető legrövidebb utat szeretnénk megtenni, Svájcon keresztül kell átutaznunk, viszont mivel ez az állam nem tagja az Európai Uniónak, magas vám adót kellene fizetnünk. Ezért sofőrünk azt a megbízást kapja, hogy kerülje el Svájcot. Programunk segítségével megtudhatja, melyik utat érdemes megtennie az utasítások betartásával.

FELÉPÍTÉSE – hogyan csináltuk?

A projektet a Visual Studio Code nevű alkalmazásban hoztuk létre. A weboldal alapját HTML (HyperText Markup Language) programozási nyelvvel írtuk meg, ezzel csináltuk meg az oldalon található gombokat, ezen kívül ez adta meg a vázát és struktúráját az oldalnak. Ehhez szorosan kötődik egy CSS (Cascading Style Sheets) nyelvű rész, ami főleg a külalak szerkesztésében játszik szerepet. Az algoritmusokat JavaScript használatával írtuk meg és építettük a programba. Maga a programozási folyamat is igényel matematikai tudást és logikát, hiszen a változók és zárójelek használata az egyik alapja minden számítógépes kódírásnak. A kész oldalunkon meg lehet adni egy kiinduló és egy célpontot, ezek közé pedig lehet különféle blokkokat vagy falat építeni. A „visualize” gomb megnyomásával elindul az animáció, ami sárga vonallal fogja jelezni a végleges legrövidebb összeköttető utat. Tettünk bele egy olyan gombot is, amivel egy útvesztőt lehet randomizálva generálni.

DIJKSTRA ALGORITMUSA

Dijkstra algoritmusa a súlyozott gráfok csúcspontjai közötti útvonalkereséssel foglalkozik, kitalálójáról, Edsger W. Dijkstra-ról kapta nevét.

Ez az algoritmus megmutatja a legrövidebb utat két csúcs között, minden lehetőséget figyelembe véve. Amikor lefuttatjuk, a képen látható táblázatban lévő információt kaphatjuk meg. Az első oszlopban a célponti csúcsot jelölő betű [[1]](#footnote-1)szerepel, a másodikban pedig a legrövidebb lehetséges út értéke[[2]](#footnote-2). A harmadik azt mutatja, hogy melyik csúcson kellet áthaladnunk utoljára[[3]](#footnote-3) ahhoz, hogy a célpontba érkezzünk. Ezeknek a vizsgálatához először két listát kell néznünk, egyet, ami azt jegyzi, hogy melyik csúcson mentünk már keresztül, és egy másikat, hogy melyiket nem látogattuk még meg. Ez azért szükséges, hogy ne vizsgáljuk kétszer ugyanazt az útvonalat. Először vegyük az „A” csúcsot A képen asztal látható

Automatikusan generált leíráskezdőpontnak. Az A csúcsnak saját magától való távolsága 0. A többi csúcsnak a távolsága ismeretlen, ezért végtelen értéket adunk nekik. Megvizsgáljuk a szomszédos csúcsokhoz vezető utak értékeit (azaz összeadjuk a jelenlegit, ami 0, a szomszédos értékével:

pl.: A -> B: 0 + 6 = 6), majd kiválasztjuk azt, amelyik a legkisebb, és ezt kicseréljük az előzőleg megadott végtelen értékkel, mivel ez jóval kevesebb annál. A program így ismétlődve végigfut a megadott és az előzőleg megismert adatokon, és mindig kicseréli a legkisebb értékre, amit talál, egészen addig, ameddig nem talál újabb kisebbet. A képen lévő példa esetében a végeredmény az alábbi táblázatban látható:

Ahhoz, hogy ebből működő algoritmust csináljunk, annyit kell tennünk, hogy mindezt beírjuk az adott programozási nyelven egy „*repeat until*” (ismételje addig) ciklusba, melynek a vége az a paraméter, hogy az összes csúcs a már érintett csúcsok listájába került.

Egy dologra viszont figyelnünk kell: hogyha a gráfunkban kialakul egy olyan kör, melyben az utak összeadott értéke negatív, az algoritmusunk nem fog működni, mert beragad. Elvégre, negatív távolságú utat képtelenség menni.

BFS (BREADTH – FIRST SEARCH) ALGORITMUS

A képen számítógép, beltéri, fából készült, faanyag látható

Automatikusan generált leírás A BFS algoritmus ugyanúgy a lehető legrövidebb út megtalálását hajtja végre, viszont sima, nem súlyozott gráfoknál. A mechanika hasonlóan kezdődik, mint a Dijkstra algoritmusnál: először megvizsgáljuk a kezdőpont két szomszédját, majd csak utána haladunk a következőkre egy réteges rendszerrel. A következőnek vizsgálandó szomszédos csúcsokat sorba tesszük, és így haladunk egyről egyre. Hogyha egy csúcsnak olyan szomszédja van, amit már egyszer meglátogattunk, szimplán nem rakjuk be a sorba. Hogyha egy csúcsot megvizsgáltunk, kivesszük a sorból. Így haladunk végig a megadott csúcsokon a kezdő és a végpont változói között. Ezután az algoritmusunk kidobja azt az utat, amelyikhez a legkevesebb ponton kellett átmennie ahhoz, hogy a célhoz eljusson.

DFS (DEPTH – FIRST SEARCH) ALGORITMUS

A képen ülő, számítógép, beltéri, halom látható

Automatikusan generált leírásA DFS egy olyan algoritmus, amelynek igazából nincsen megadott célpontja. Működése arra épül, hogy minél tovább utazzon a gráf csúcsain anélkül, hogy kétszer ugyanazon a ponton menne át. Hogyha olyan pontba akad, ami már vizsgálva volt vagy éppen vizsgálatban van, vagy esetleg zsákutcához ér el, akkor visszamegy ahhoz a legközelebbi elágazáshoz (erre azt a fogalmat használjuk, hogy *backtracking*), ahol tud még másik irányba menni, ezzel elkerülve az utak metszését vagy ütközését.

forrásaink a projekthez:

<https://www.youtube.com/watch?v=msttfIHHkak>

* magához a kód struktúrájához

<https://www.youtube.com/watch?v=7fujbpJ0LB4>

<https://www.youtube.com/watch?v=oDqjPvD54Ss>

<https://www.youtube.com/watch?v=pVfj6mxhdMw>

* algoritmusok

<https://github.com/clementmihailescu/Pathfinding-Visualizer-Tutorial>

1. vertex oszlop = célponti csúcsot jelölő betű [↑](#footnote-ref-1)
2. shortest distance from A = az A ponttól való legrövidebb távolság [↑](#footnote-ref-2)
3. previous vertex = az a csúcs, amelyiken utoljára haladtunk át a célpont előtt [↑](#footnote-ref-3)