Feladatok Algoritmus Vizualizációval

Bende Imre

beiraai@ludens.elte.hu
ELTE IK

Absztrakt. Arra szeretnék választ adni a következőkben, hogy algoritmus animációval, vizualizációval milyen feladatokat, módszereket lehet felhasználni a tanórákon, valamint a számonkérés során. Illetve milyen feladatok segíthetik a programozásoktatást, amelyeket mintákkal, példákkal is szemléltetek.

Kulcsszavak: algoritmus animáció, algoritmus vizualizáció, programozásoktatás, számonkérés

1. Bevezetés

Az algoritmus vizualizációs eszközök alapkövei lehetnek az informatikaoktatásnak. Hatékonyságát, helyét az oktatásban már számos helyen bizonyították (például: Karavirta disszertációjában összegyűjtött több kutatást is, amelyek erre az eredményre jutottak [3]). Ha a lehetséges problémákra választ, megoldást tudunk adni, akkor nem lehet kérdés, hogy érdemes-e felhasználniuk a tanároknak a módszert a tanóráikon. Azonban hiába szeretnénk használni, kérdés lehet az, hogy érdemes-e, és ha igen milyen feladatokat, módszereket lehet felhasználni a számonkérés során. A következőkben erre szeretnék választ adni, úgy, hogy leírom én milyen módszereket tudnék elképzelni, majd ezeket példákkal is bemutatom. Megjegyezném, hogy az algoritmus animáció és a vizualizáció között van különbség, azonban a cikk során ezeket nem különítem el és mindkettőt felhasználom a feladattípusok létrehozása közben.

2. Feladattípusok

2.1. Algoritmus leírása papíron

Papír alapú számonkérésként azt tudom elképzelni, hogy le van írva az algoritmus maga, vagy csak annak megnevezése, majd egy minta teszteset segítségével le kell írniuk a diákoknak, hogy az egyes lépésekben mi történik (gondolok itt arra, hogy egy-egy közbenső utasításnak mi a célja; mik a változók értékei, hogyan módosulnak), illetve az is egy lehetséges feladat emellett, hogy valamilyen módon a papíron ábrázolják a főbb lépéseket vizuálisan. Az értékelés főbb szempontja ebben az esetben az, hogy minden lényegesebb utasításnál tett-e valamilyen féle megjegyzést, illetve a főbb változók (lehet olyan feladat, melyben mondjuk egy ciklusváltozót nem kötelező jelölni, mivel nem játszik nagyobb szerepet) értékei leírásra kerültek-e.

Ha nem is az online világot nézzük, szorosan az algoritmus vizualizációhoz köthetjük az egyes feladatok életben vett reprezentációit. Ezekre nagyon jó mintákat, konkrét feladatokat kínál a CSUnplugged című könyv [1], mely feladatonkénti leírással, felhasználhatósággal, anyagokkal rendelkezik. Mindemellett a feladatok valósághűvé teszik az algoritmus működését, használatát, így az értelmezése is könnyebbé válik a fiatalabbak számára, mindezt játékosan próbálták megalkotni a könyv készítői. A kötetben találhatunk többek között kereséses (lineáris, logaritmikus, hasheléses), rendezéses, gráfalgoritmusos gyakorlatokat is.

Az Algorithms Unplugged [7] hasonló feladatokkal rendelkezik, mint az előző bekezdésben említett CSUnplugged, viszont komplexebb algoritmusokat (keresés, rendezés, gráfalgoritmusok, dinamikus programozás, hatékonyság javítás) mutat be, amelyek idősebb, több alapismerettel rendelkező diákoknak lehet megfelelő. A megközelítés hasonló: a könyv motivációt akar adni a diákoknak a

programozástanulásra offline feladatokkal. Részletes magyarázattal és többfajta vizualizációs példával mutatja be az algoritmusokat. Több esetben találkozhatunk olyan tananyagokkal is, amik csak egyetemen fordulhatnak elő, viszont megfelelő illusztrációkkal ezeket is jól és érthetően mutatja be.

Egy másik érdekes megközelítés a Dynamicland [I5] környezete, ahol a Lua-ban megírt programoknak fizikai megjelenítése van. A papírlapok funkcionálnak programként, illetve fizikai tárgyakat tudunk bemenetként, módosító tényezőként használni. Jelenleg ez még elég új, de erre a technológiára a későbbiekben mindenképpen érdemes odafigyelni, illetve amennyiben elterjedtebb lesz, mindenképpen érdemes lehet kipróbálni, használni az informatika órákon.

2.2. Algoritmus vizualizáció gyűjtése

Előre kiszabott, vagy saját érdeklődés alapján kiválasztott algoritmusról lehetne algoritmus vizualizációkat keresni gyűjtőmunkaként, majd ezeket a diákság osztályozza, értékeli, s végül összehasonlítja őket egymással. A keresést lehet támogatni azzal, hogy vizualizációs gyűjteményeket, kollekciókat, tárházakat mutatunk nekik, így már eleve tudják hol kell kezdhetik el a keresést (példa kedvéért: AlgoViz [I1], VisuAlgo [I8]). A tanórákon elmondjuk mely szempontok lehetnek fontosak egy vizualizációval kapcsolatban (gondolok itt Myller által meghatározott pontokra [4]), melyek alapján tudják értékelni az egyes eszközöket. A végtermék egy dokumentum, linkgyűjtemény lenne, amely összehasonlít egy algoritmushoz tartozó vizualizációkat, sőt akár végeredmény lehet egy vizualizáció is, amelyet aztán az osztály többi tanulójának is meg lehetne mutatni tanulás céljából, ezáltal a tanár feladatainak száma is csökkenhet.

2.3. Algoritmus vizualizáció gyűjtése

Lehetséges otthoni feladat egy algoritmus vizualizáció megtervezése, elkészítése, lefejlesztése. Első feladatként a tanár feladhatja, hogy készítsenek a diákok egy dokumentumot, amelyen részletesen megfogalmaznak egy algoritmust (lehet szó olyanról, amit órán már vettek, vagy olyanról is amit máshonnan ismernek és érdekesnek találtak), illetve egy hozzá tartozó animációt, vizualizációt, bemutatva azt, hogy mely esetekben, mikor, mi történik rajta. Ha vannak jobb képességű, tehetségesebb tanulók, akkor pedig folytatásként, akár szorgalmi feladatként lehet az, hogy a dokumentum alapján elkészítik ténylegesen is az AV-t (opció lehet, hogy ekkor más diákok által elkészített tervezetek közül is választhatnak, ha nekik az jobban tetszik, szívesebben foglalkoznak vele). Könnyítés lehet, hogy ha mutatunk nekik egy olyan keretrendszert, library-t, mely részletes, jó leírást, dokumentációt tartalmaz, így egyszerűbb lesz a helyzetük a feladat elkészítésében. Mivel ezt főként otthoni, szorgalmi feladatnak szánnám ezért, ha a munkán és annak részletességén látszik a megfelelő vele eltöltött időmennyiség, akkor azt valamiféle jutalmazással értékelném. Azáltal, hogy maguk készítenek egy vizualizációt sokkal jobban megértik az általuk választott algoritmus működését, illetve plusz programozói tudásukat, kompetenciáikat is fejlesztik környezettől függően (például egy webes felület esetén HTML, CSS, JavaScript készségeiket is fejlesztik).

Algoritmus vizualizáció készítése lehetséges lenne más eszközökkel is. Lehet egyszerűen állapotokat készíteni, rajzolni akár papíron, akár gépen, ezeket lefotózva diavetítésszerűen lehetne animálni, bemutatni az algoritmus egyes lépéseit. Minél részletesebben mutat be egy-egy lépést, állapotot (ami fontos az algoritmus működése szempontjából), minél több lépést tartalmaz az animáció, annál jobban magyarázza el, mutatja be az algoritmust. Ennél összetettebb egy Flash készítése, ami hasonló elven működhet, de közben a Flash készítést is gyakorolják a diákok. Flash alapúhoz hasonló megoldásokat találhatunk például Végh Ladislav által alkotott weblapon is, mely referenciaként szolgálhat hasonló animációkhoz [6], de az AlgoViz oldalán is találhatunk ilyeneket [I1].

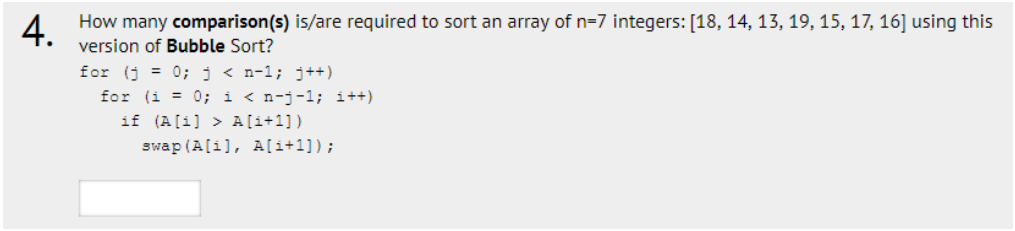
Emellett persze csak a képzelet szab határt annak, hogy egy-egy diák milyen megközelítésből is tud, vagy szeretne megcsinálni egy vizualizációt. Itt mindenképpen meg szeretném említeni Kátai Zoltán és Tóth László által rendezett néptáncokat, melyek egy-egy algoritmust mutatnak be táncosokkal (példa kedvéért buborékos rendezés, gyors rendezés, összefésülés rendezés stb.) [I7]. De nagyon

látványos animációkat is lehet találni az interneten, ilyen például egy kis robot által bemutatott részletes videó, mely a jelentősebb rendezéseket mutatja be, majd hasonlítja össze őket [I6].

2.4. Algoritmikus feladatok megoldása AV segítségével

Számonkérés lehetséges lenne gépen is, online, interaktív tesztek formájában. Ebben az esetben fel lehetne tenni működésbeli kérdéseket az algoritmussal kapcsolatban (például: egy konkrét bemenet alapján a buborékos rendezés hány cserét igényel, adott állapotban a változók milyen értékekkel rendelkeznek, mely algoritmus a leírt pszeudokód). A válaszokat megadhatjuk kérdés típusától függően sima bemeneti mezőbe, feleletválasztósként, illetve igaz/hamis formában is. Az algoritmus működését érintő kérdéseknél a bemeneti értékeket random függvénnyel is generálhatjuk, így nem kell több különböző kérdéssort összeállítani, hiszen a tanulóknak nagy valószínűséggel különböző feladatok jönnek létre. Mivel a teszt online környezetben jönne létre, így a tanárnak az értékeléssel egyáltalán nem kellene foglalkoznia, vagy csak kevés feladata lenne vele. Tanártól és iskolától függően pedig állítható, hogy a százalékos eredmények alapján, milyen érdemjegy jár a dolgozatot kitöltő diáknak.

Végh Ladislav hasonló feleletválasztós feladatsorokkal próbálta vizsgálni, hogy a diákok mennyire értették meg az órákon szereplő algoritmusokat [6]. De az interneten is találhatunk erre hajazó, hasonló példákat, ahol a megoldások a teszt kitöltését követően kiértékelésre kerülnek. Ilyenek vannak például az VisuAlgo honlapján [I8], mely nem csak vizualizációkat, hanem a témakörökhöz tartozó feladatsorokat is tartalmaz.



1. ábra: Példa VisuAlgo honlapján lévő tesztfeladatra

Emellett el tudok képzelni olyan feladatokat is, amelyeknél egy animációt nézve kell meghatározni, leírni a bemutatott algoritmust (a megoldásoknál pszeudokód alapú algoritmusleírás lehetséges). Alapvetően gondolok itt arra, hogy az algoritmus megjelenik a felületen, de néhány változó, utasítás ki van szedve belőle, melyeket a diákoknak kell kiegészíteni a vizualizáció lépéseinek megfigyelésével, megvizsgálásával. Összetettebb, nehezebb lehet, ha az animáció alapján az egész algoritmust kell meghatározni, de ezt csak részletesebben ábrázolt, könnyebben felismerhető algoritmusoknál lehetne feladni, megoldani.

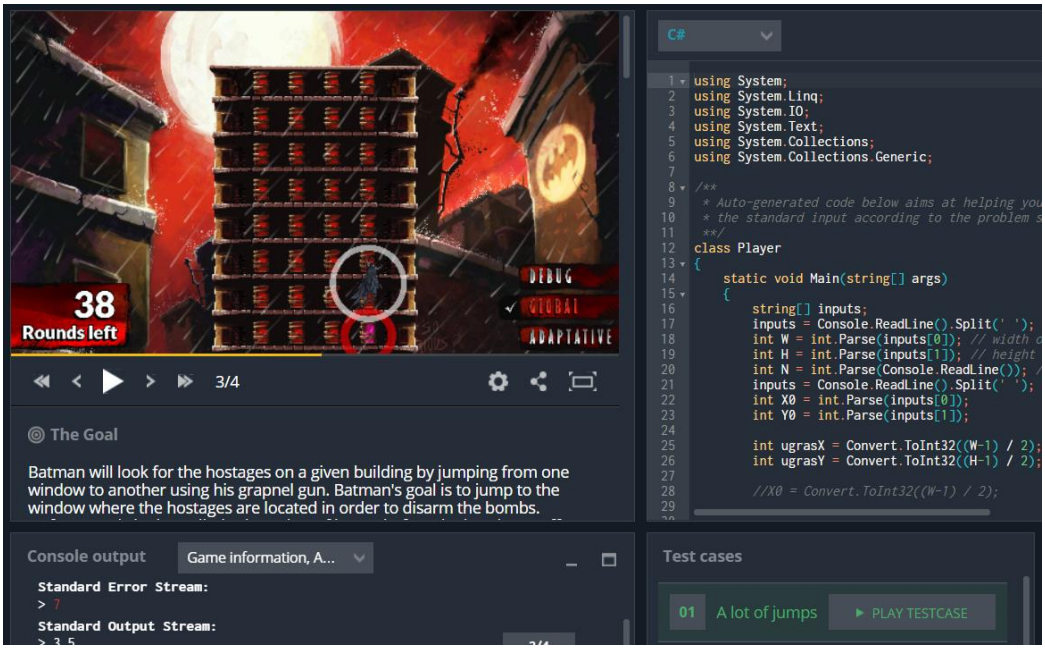
Egy másik megközelítése lehet az előző fejezetben említett feladattípusnak, hogy az algoritmus leírását és/vagy algoritmus nevét ismerjük, majd a feladata az lenne, hogy a diákok az animációt irányítsák pszeudokód, vagy saját tudásuk alapján. Interaktivitási eszközök lehetnek az elemek helyükre húzása, értékek beírása. Ebben az esetben a feladat megoldása sikeres, ha az algoritmus végeredményét megkaptuk, de úgy, hogy az algoritmus működését követtük.

2.5. Programozási feladatok AV támogatással

Alapvetően nem minden diák számára motiváló, hogy amikor programozási feladatokat oldanak meg kódolással, akkor visszacsatolásként, tesztelésként mindösszesen egy konzolt látnak, illetve csak azt tudják használni. Érdekesnek tartom azt az ötletet, hogy ha fogjuk ezeket a hagyományosnak vett programozási feladatokat és a feladat megoldását, tesztelését algoritmus animációval, vizualizációval támogatnánk meg. Ezt úgy tudom elképzelni, hogy az algoritmus eredménye irányít egy animációt, illetve különböző előre megírt tesztesetek léteznek a feladatokhoz, majd a mi algoritmusunk ezeket

próbálja megoldani, mely során látványos eszközökkel ad egyfajta visszaigazolást a program helyességére vonatkoztatva.

Példaként a CodinGame oldalán [I4] ez egy megvalósított eszköz. Többféle típusú, nehézségű feladat van, melyek jelentős része vizualizációval van támogatva. Ez társul azzal, hogy nemcsak a módszert használja, hanem a feladatok is érdekesnek mondhatók, így összeségében ez motiváló erőként hathat azok számára is, akik kevésbé ilyen beállítottságúak, kevésbé szeretnek konzolos programokat írni algoritmus tanulás szempontjából.

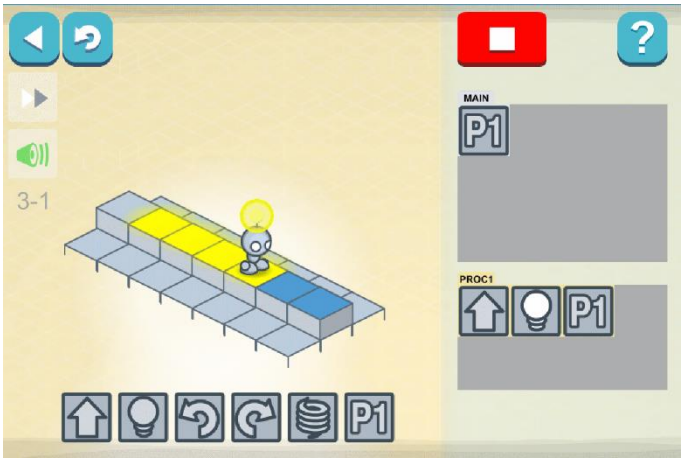


2. ábra: Egy feladat a CodinGame oldalán

Az ábrán jól látszik a felület összes eleme. Bal oldalon helyezkedik el a vizualizáció (léptethetőséggel, videó lejátszó funkciókkal), a feladat szövege és a konzol bemenete/kimenete, míg jobb oldalon a megírt forráskód (több programozási nyelv közül is lehet választani), illetve a tesztesetek elindító gombjai vannak.

Egyszerűbb feladatok mellett mesterséges intelligencia alapú játékok is megjelennek az oldalon, melynek elkészültekor lehetőség van más emberek munkáival versenyezni, összecsapni.

Emellett egyszerűbb utasításalapú feladatokat, „játékokat" is találhatunk az interneten. Többek között ilyen a CodeCombat [I3], vagy a Lightbot [I5]. Míg az elsőben egy harcost kell irányítani a várbörtönben különböző nehézségeket leküzdve, a másikban egy robottal kell végig menni a pályákon úgy, hogy mindeközben a kék mezőket kivilágítja (utasítások, lépések kiválasztásával).



3. ábra: Lightbot egyik feladánya

3. Amőba játék

Pár gondolat, játék a feladattípusok kutatása közben megihletett, így elkészítettem egy egyszerűbb kódolásos játékot. Az alapelgondolás azonos az amőba játékkal azonban ahelyett, hogy a diákok egymással csapnak össze mesterséges intelligenciákat (későbbiekben MI) készítenek, fejlesztenek (jelenleg ezt JavaScript nyelven), melyek helyettük játszák le a játszmákat. Habár az amőba közismert játék, de azért röviden leírom a lényegét: Két játékos van, az egyik az X-szel, a másik a O-rel, majd egy táblázatban egymás után felrajzolják egy-egy négyzetbe a jelüket. Az a játékos nyer, aki vízszintesen, függőlegesen vagy átlóba tud öt azonos jelölést tenni (ennek rövidebb verziója a 3x3-as tábla, ahol csak hármat kell összegyűjteni).

A diákok egy-egy mérkőzés után átgondolhatják mi az, amit másképpen csinálhatna a programjuk, hol, miért is veszített az, ezáltal motiválva vannak a folyamatos fejlesztésre, hogy minél „jobb" amőba MI-t készítsenek. Nem csak másokkal állíthatják szembe kódjukat, hanem saját maguk is játszhatnak ellene (kézi irányítással), így egy „belső" környezetben is tudják tesztelni, mérni annak képességeit, határait, hiányosságait.

Mivel van beépített MI is (mivel ez JS-en íródott, így a forráskódját a diákok elérik, illetve az egész oldal is letölthető a GitHub-ról), amit (direkt) viszonylag egyszerűbb legyőzni ezért általános iskola 7. osztályától felhasználható, akár órai keretek között is bemutatásra. Azonban az egymás közti mérkőzéseket nem célszerű órán is használni, hiszen aki kevésbé ügyes, gyengébb kódot készít demotivált lesz, kevésbé lesz kedve ezzel foglalkozni. Ehelyett azonban órán kívüli tevékenységként, szakkörök alkalmával, illetve akár versenyként is tökéletesen fel lehetne használni ezt a játékot.

4. Feladattípusok kategorizálása

Kérdés lehet, hogy melyik típust, kiknek, milyen körülmények között lehet, érdemes használni. A következőkben különböző szempontok szerint kategorizálom az egyes feladattípusokat. Korosztály, módszer és interaktivitás alapján.

4.1. Koronként vizsgálva

Első lépésben korosztályonként vizsgálom a kérdést, figyelembe véve a jelenlegi Kerettantervet [I6].

- A fiatalabbaknak (általános iskolásoknak) mindenképpen a szórakoztatóbb, egyszerűbb feladatokat ajánlom, gondolok itt a CSUnplugged-ban található mintákra, illetve egyszerűbb utasításalapú játékokra (ezek elkezdése minél előbb lehetséges, hiszen sok esetben olvasni, írni sem kell tudni ahhoz, hogy a robotot vagy az adott figurát irányítani lehessen).
- A középkorosztálynak általános iskola végén, gimnázium elején, mikor tanulnak már programozási tételeket, egyszerűbb rendezéses algoritmusokat, akkor már képesek arra, hogy fényképekből, diákból készítsenek animációkat, melyeket bemutathatnak a többieknek is. Idősebbeknek, gimnazistáknak az összetettebb, de mégsem a legnehezebb AV-alapú programozós játékokat (például CodinGame) lenne célravezető feladni, illetve akár komplexebb algoritmusokról szóló AV-k készítését is rájuk bízhatjuk.
- Végül egyetemistáknak lehetséges módszer, hogy a nehezebb programozási feladatokat feladjunk, illetve ekkor már rendelkeznek olyan szintű programozási, algoritmikus, technológiai tudással, ismerettel, gondolkodással, hogy saját maguk is képesek legyenek algoritmus animációt, vizualizációt létrehozni (gondolok itt Flash alapú animációra, weboldalra, vagy akár egy Unity-vel létrehozott interaktív animációra is).
- A tesztalapú feladatok nehézségét könnyen lehet állítani, így azt kortól függetlenül lehet használni számonkérés során. Animáció mutatásával, annak részletességével, kérdések összetettségével, komplexitásával, bemutatott algoritmusok bonyolultságával lehet állítani a kérdéseken úgy, hogy a kiválasztott csoport, osztály előzetes vagy tanult tudásának, ismereteinek megfelelő legyen. Fiatalabbaknál inkább már tanult algoritmusokról érdemes feladatokat összeállítani, míg később új, eddig nem ismerteket is fel lehet használni, hiszen ekkor a diákok már képesek pszeudokód alapján értelmezni egy algoritmus működését.

Soron kívül pedig megemlíteném még a Logo-t, illetve a Scratch-et melyet főleg kicsiknek, általános iskolásoknak lenne érdemes használniuk. Manapság már a legtöbb helyen beépült a tananyagba, illetve rendelkezik olyan dokumentációval, segédeszközökkel, amelyek segítik a tanárok munkáját.

4.2. Módszer szerint

Ebben a fejezetben aszerint osztom szét az egyes feladattípusokat, hogy mely módszernek, számonkérési formának, melyet tartom ideális választásnak.

- Órai, otthoni gyakorlásra a gép nélküli offline tartalmakat, illetve a programozási feladatokat ajánlom, melyek megoldási menete, hosszúsága bele tud férni a tanórák hosszúságainak.
- Szorgalmi feladatnak a vizualizációkészítés, vizualizáció keresés lehet jó választás, hiszen az órai keretek, óraszámok rövidségébe ez nem férhet be, így pedig annyi időt foglalkozhatnak, tölthetnek vele, amennyit csak szeretnének.

- Számonkérésként érdemjegyért az online tesztfeladatokat és a komplexebb programozási feladatokat érdemes használni, hiszen ezek objektíven értékelhetők és könnyen javíthatók.
- Versenyfeladatként az algoritmus vizualizációval támogatott programozási feladatokat tudnám elképzelni, hiszen komplexebb feladatok is létrehozhatók a módszerrel, illetve segítséget tud nyújtani abban, hogy a diákok megértsék, majd megoldják azt. Az MI (mesterséges intelligencia) megoldást igénylő feladatokra bajnokságszerűen fel lehet építeni egy-egy versenyt, melynek menetét többféleképpen is felépíthetjük (egyenes kieséses, csoportmérkőzéses). Így ezáltal egyértelműen lehet kiválasztani a legjobbakat és nem, vagy csak nagyon ritkán fordulhat elő holtverseny a fontosabb helyezéseken.

Az előbb említett feladattípus nem csak egy megrendezett verseny keretein belül megvalósítható, hanem tanórákon (plusz akár otthoni munkával, továbbfejlesztéssel) is létre lehet hozni egy ilyen környezetet, mely motiválhatja a diákokat, hogy a többiek munkájával versenyezhessen. Ebben az esetben azonban oda kell figyelni azon hallgatókra is, akiket demotiválja a „vereség", lemaradás. Ennek elkerülése végett megoldás lehet az, hogy gyengébb MI-ket, megoldásokat is feltöltünk, melyeket az ő programjaik is le tudnak győzni.

4.3. Interaktivitás szerint

Már több kutatás is kimutatta, hogy minél interaktívabb egy algoritmus vizualizációs eszköz, annál hatékonyabb az a programozásoktatás során (Kavarirta által összegyűjtött eredmények [3]). Így figyelembe véve a Myller által meghatározott kiegészített taxonómiát [4] interaktivitási szint szerint is kategorizáltam az egyes feladattípusokat.

- Válaszadás: A különböző tesztekkel, vizsgafeladatok az interaktivitás ezen szintjét érjük csak el, viszont ezekben az esetekben a cél a tudás vizsgálata, nem pedig annak a megszerzése.
- Változtatás: A játékos feladatokkal a kimenet és maga a vizualizáció módosítható válik. Mindenképpen érdemes ezeket használni a programozással, algoritmizálással való ismerkedés során.
- Összerakás: Dynamicland-del és hasonló eszközökkel a diákok maguk is létrehozhatnak algoritmusokat, valamint a hozzá tartozó vizualizációikat.
- Bemutatás: Az interaktivitás legmagasabb fokán az van, ha a diákok maguk készítik el a vizualizációt, majd bemutatják azt társaiknak. Vagy akár csak az utóbbi, hiszen már ekkor is mélyebben meg kell ismernie a diáknak az algoritmus működését, hogy el tudják azt magyarázni.

5. Zárszó

A cikkben igyekeztem minél többféle feladatot, feladattípust, módszert felsorolni, amiket az informatika órákon, programozásoktatás során fel lehetne használni úgy, hogy mindeközben algoritmus vizualizációs (vagy animációs) eszközöket veszünk igénybe támogatásként. Több esetben is látszik az, hogy a felhasználása ezeknél a feladatoknál hatékonyabbá, élvezhetőbbé teszi a tanulást, illetve motiváló a diákok számára. Érdemes lehet változatosan, időkerettől függően minél több fajtát kipróbálni a diákokkal, hátha megszeretik ezeket az alkalmazásokat, módszereket és így szabadidejükben is fogják használni azokat önfejlesztésre (amelyet sok esetben csak játéknak tekintenek). Természetesen a feladattípusok listája nem teljes, technológiai, módszertani újítások folyamatosan jönnek létre és terjednek el, csak a képzelet szab határt annak, hogy még miként lehetne beépíteni a tanórákba az algoritmus vizualizációs eszközöket és az ehhez kapcsolódó feladatokat, módszereket. Ami biztos: mindenképpen használjuk az informatika órákon probléma-, feladatmegoldásra az AV-t, és folyamatosan

figyeljük azt, hogy mivel lehetne még élvezhetőbbé tenni a diákok számára a programozástanulást a módszer segítségével, mivel egyre könnyebben elérhetőek mindenki számára ezek az eszközök.

Irodalom

1. Mike Fellows et al.: *Computer Science Unplugged*, Computer Science Unplugged, 2015.
2. Christophet D. Hundhausen, Sarah A. Ddouglas Andjohn T. Stasko: *A Meta-Study of Algorthm Visualization Effectiveness*. Journal of Visual Languages and Computing, pp259-290, 2002.
3. Ville Karavirta: *Facilitating Algoritm Vizualization Creation and Adoption in Education*. Helsinki University of Technology, 2009.
4. Niko Myller, Roman Bednarik, Erkki Sutinen, Morechai Ben-Ari: *Extending the engagement taxonomy: Software visualization and collaborative learning*. ACM Transactions on Computing Education, New York, USA, 2009.
5. Törley Gábor: *Vizualizáció a Programozástanításban*, ELTE, 2013.
6. Végh Ladislav: *A Programozás Tanulásának és Tanításának Támogatása Elektronikus Tananyagba beépíthető Interaktív Animációs Modellekkel*, ELTE, 2017.
7. Vöcking B. et al.: *Algorithms Unplugged*, Springer-Verlag Berlin Heidelberg, 2011.

Internetes hivatkozások

I1. The Algorithm Visualization Portal – http://algoviz.org

I2. Algorithm Animations and Visualizations – http://www.algoanim.ide.sk

I3. CodeCombat oldala – https://codecombat.com/

I4. CodinGame oldala – https://www.codingame.com/

I5. Dynamicland oldala – https://dynamicland.org/

I6. Kerettanterv – http://kerettanterv.ofi.hu/

I7. Lightbot oldalat – http://lightbot.com/

I8. Youtube: Merge Sort vs Quick Sort – https://www.youtube.com/watch?v=es2T6KY45cA

I9. Youtube: Merge-sort with Transylvanian-saxon (German) folk dance – https://www.youtube.com/watch?v=XaqR3G_NVoo

I10. VisuAlgo weboldal – https://visualgo.net/training