Doodling és algoritmika

Osztián Pálma Rozália[1], Kátai Zoltán[2], Osztián Erika[3]

{[1]osztian.palma, [2]katai_zoltan, [3]osztian}@ms.sapientia.ro
Debreceni Egyetem, Sapientia EMTE

Absztrakt. A jegyzetkészítés vagy a piszkozat használatának fontossága számos tudományterületen megmutatkozik. Legyen szó matematikáról, irodalomról, történelemről vagy bármi más szakterületről a jegyzetek használata és elkészítése sokszor elengedhetetlen folyamata a tanulásnak. A tanulás mellett, az úgynevezett „doodling" (doodle) jelenség jelentősen hozzájárul a diákok problémamegoldó képességének fejlesztéséhez, és a feladatok megértéséhez is.

A különféle szakterületek közül az informatika és a számítógépes gondolkodás sem képeznek kivételt. Előzetes kutatások során bebizonyosodott, hogy a doodling nem csupán a problémamegoldó képességekre összpontosít, hanem kiemelt hangsúlyt helyez, sőt részese a feladat megértésének és elemzésének, főként kezdők esetén.

Dolgozatunk célja, hogy vizsgáljuk a doodling számítógépes gondolkodásra gyakorolt hatását három különböző algoritmus vizualizáció esetén. Továbbá, kíváncsiak vagyunk arra is, hogy miként befolyásolja a megértést a résztvevők előzetes programozási tapasztalata.

Kulcsszavak: doodling, problémamegoldó képesség, algoritmus, vázlat, nyomon követés, piszkozat

1. Bevezető

1.1. Algoritmika

Különféle algoritmikai feladatokkal akár már egészen kiskorban találkozhatunk. Minden kihívást jelentő matematikai-, fizikai-, kémiai–, informatikai- és hasonló szakterületekhez tartozó kérdésben fellelhetőek algoritmikai vonatkozások. De talán nem is kell szakterületekről, iskoláról és oktatásról beszélnünk, hiszen a mindennapi élet során felmerülő események és teendők is sokszor azonosíthatók különféle algoritmusokkal. Ezek közül a legismertebbek közé tartoznak a keresési és rendezési algoritmusok, melyekkel mindenki találkozhat élete során, akár tudatában van, akár nem. A rendező és kereső algoritmusok szemléltetéséhez és megtanításához, az adott korosztály függvényében, különféle módszereket alkalmaznak a pedagógusok.

1.2. Vizualizációk

A Sapientia Erdélyi Magyar Tudományegyetemen működő AlgoRythmics kutatócsoportunk egyik legjellegzetesebb kutatási témája a különféle algoritmusvizualizációkkal való oktatás [1], [2], [3], [4]. A módszer különlegessége abban rejlik, hogy a különféle kereső-, és rendező algoritmusok emberi mozgás által, a tánc nyelvén kerülnek szemléltetésre. Mostanáig, összesen 10 tánckoreográfiával szemléltetett algoritmus látott napvilágot, melyek három különböző táncstílus révén kerültek bemutatásra (néptánc, flamenco, ballet). A videók repertoárját idén az előzőekhez hasonló, mégis új stílust képviselő vizualizációval sikerült bővíteni. A tánc helyét ezúttal a leegyszerűsített emberi mozgás, pontosabban szólva a színészi játék vette át. Mindez lehetőséget adott arra, hogy további kutatásokat végezzünk az emberi mozgás által szemléltetett vizualizációk hatékonyságának felmérése kapcsán.

1.3. Doodling

A vizualizációk mellett még számos tényező befolyásolhatja az algoritmusok megértését. Több kutatás is amellett érvel, hogy a különféle programozási feladatok kihívást jelentenek a diákok számára [5], [6]. Sokszor az informatikai alapfeladatok ismerete és megértése, akár egy év programozás tanítás után sem bizonyul elegendőnek, főként kezdők esetén [7], [8]. Ennek egyik oka a diákok problémamegoldó

képességének hiánya lehet és az, hogy nem osztják részfeladataira az adott problémát [9]. A probléma részfeladatokra való elosztása és megértése nagymértékben elősegíthető a különféle papír alapú levezetési stratégiák által [7], [5], [6], [10]. A „doodling" egyik típusa a vázlatkészítési stratégiáknak, mely jelentősen elősegíti egy algoritmus lépéseinek levezetését, valamint egy adott kérdéshez tartozó helyes válasz meghatározását.

Jelen tanulmány három központi témája közé az algoritmusok, a vizualizáció típusok és a firkálás („doodling") tartozik. Az algoritmusok tekintetében két rendező stratégián (beszúró-, shell rendezés) mértük a hallgatók teljesítményét, melyet három különböző vizualizációval (animáció, tánc és színészi játék) szemléltettünk. Továbbá, a mérés során a résztvevőknek lehetőségük volt piszkozatlapot használni, melyre lejegyezhették a feladathoz kapcsolódó megoldásmenetet. Kíváncsiak voltunk, hogy a hallgatók hogyan teljesítenek a két algoritmus esetén, és arra is, hogy az eredményeket miként befolyásolja az előzetes programozási tapasztalat, a vizualizációk típusa, valamint a piszkozat használata.

2. Szakirodalmi áttekintő

2.1. A vázlatkészítéstől a doodle-ig

Az algoritmusok megértésének elősegítése érdekében a programozási feladatok megoldását a hallgatók gyakran különféle reprezentációkkal egészítik ki, amelyek úgynevezett annotációként szolgálnak és igen hasznosnak bizonyulnak [10]. Ezeknek a technikáknak három nagy csoportját különböztethetjük meg:

- **sketching** (vázlat: a programozó által meghatározott programállapotról vagy bármely más számítási folyamatról készített írásbeli vizualizáció leírása [7]);
- **tracing** (nyomon követés: egy algoritmus végrehajtási folyamatának a teljes vagy részleges emulálása, utánzása [11]) és;
- **doodling** (firkálás: diagramok és megjegyzések, amelyeket általában tapasztalt programozók írnak vagy rajzolnak, amikor egy algoritmus működésének meghatározásával szembesülnek [5]).

A feladatok megértése és megoldása szoros összefüggésben van a diákok problémamegoldó képességével. Ennek a képességnek az öt alapvető lépését a McCracken kutatócsoport [9] is meghatározta: (1) a probléma elvonatkoztatása a leírásától, (2) al-problémák generálása, (3) az al-problémák al-megoldásokká való átalakítása, (4) újra összeállítás vagy rekompozíció és (5) értékelés és iterálás. Ugyancsak az ITiCSE 2004-es „McCracken kutatócsoport" munkájához fűződik a doodling kérdésköréhez tartozó egyik legjelentősebb kutatás, akik szerint a rutinnak számító programozási feladatok végrehajtásának gyenge ismerete hatással van a hallgatók problémamegoldó képességének és programozási ismeretének hiányosságaira. Ilyen rutinnak nevezhető feladatnak számít a **nyomon követés** (tracing) is, amely nagyon sok hallgatónak, főként a kezdőknek ismeretlen [9]. Hasonló következtetéshez jutott Fitzgerald és kutatótársai is [11], akik szerint a problémamegoldó képességnek a hiánya szoros kapcsolatban áll a hallgatók bizonytalan vagy éppenséggel kevés tudásával. Kutatásában ő is megfogalmazza azt, hogy mindez fejleszthető, ha külön figyelmet fordítunk a doodlingre vonatkozó stratégiákra. Ő említi meg többek között a "gondolkodj hangosan" ("think aloud") elvet, amely kapcsán arra próbálja ösztönözni a hallgatókat, hogy egy probléma vagy feladat megoldása esetén legyenek tudatában annak, hogy melyik lépés következik, vagy miért cselekedtek úgy, ahogy ("What are you thinking?, Why did you do that?").

Figyelembe véve a diákok előzetes programozási tapasztalatait minden esetben találkozhatunk hiányosságokkal, mégis érzékelhető egy látványos szakadék a kezdők és a haladók gondolkodása között. Több kutató is megfogalmazza azt, hogy míg a haladó programozási tapasztalattal rendelkező diákok részfeladataira osztanak egy-egy nagyobb problémát, a kezdők ritkán, vagy egyáltalán nem alkalmazzák ezt a stratégiát és teljes egészében próbálják megoldani a feladatot [9]. Mindez kihatással van arra is, hogy nem alakul ki bennük az a rutin, hogy az egyszerűtől a bonyolult fele haladjanak. Lister és kutatótársai a sakkal szemlélteti és helyezi párhuzamba ezt a jelenséget [12], ahol arról számolnak be, hogy haladó és

kezdő játékosok két külön módszert alkalmaznak a sakkfigurák elhelyezkedésének memorizálására. Chase és Simon [13] kutatása rávilágít arra, hogy a kezdők olyan módszert alkalmaznak, amellyel minden sakkfigura helyzetét külön-külön próbálják megjegyezni, elszigetelten a többitől (izolált gondolkodás), míg ezzel szemben a haladók a támadó és védekező lehetőségek alapján próbálják felidézni a sakkfigurák pozícióit (absztrakt gondolkodás). Mindez az algoritmikai feladatok megoldásában is tükröződik, ahol kezdők esetén különösen érzékelhető a „nem látja a fától az erdőt" jelenség [12]. Lister és kutatótársai azt is kihangsúlyozzák, hogy ennek elősegítésére érdemes figyelmet fordítani egy koherens struktúra kidolgozására, melyhez a feladat lépésről lépésre való nyomon követése jelentősen hozzá tud járulni.

2.2. A „vak számítógéppel" való azonosulás

Egy másik oka a problémamegoldó képesség hiányának lehet az, hogy a diákok nem tudnak azonosulni a „vak számítógéppel" [14]. Cunningham és kutatótársai úgy tartják, hogy a vázlatkészítés olyan, mint egy elosztott megismerés, és az, ami a vázlatot (a lapon végzett jelek, a folyamat, amely során ezek változnak) és a vázlat készítőjét (a diák) összeköti nem más, mint a diák munkájának megismerése, hogy egy közös válasz szülessen egy adott feladatra [7]. Kutatásukban azt is megfogalmazzák, hogy ez a nyomon követés elősegítheti, hogy a "gép" működése nyilvánvalóvá váljon. Több kutató is azonosul ezzel az elvvel, miszerint a hallgatónak meg kell ismernie, kell azonosulnia a „vak számítógép" [14], a „képzeletbeli számítógép" (notional machine) [15], [16] működésével, hiszen ezáltal elősegíthető egy mentális modell kialakítása [7]. Mindezt, a kód olvasáshoz, -íráshoz, -hibajavításhoz kapcsolódó feladatok megismerése és lépésről lépésre való követése jelentősen elő tudják segíteni. Néhány kutató úgy hivatkozik a „képzeletbeli számítógéppel" való azonosulásra, mint „emberi fordító" (human compiler) vagyis az az aktív résztvevője a folyamatnak, aki elemzi, és lépésről lépésre kiértékeli a részeredményeket [10].

2.3. Interaktivitás

Azt gondolnánk, hogy vázlatot készíteni, "firkálni", egy önálló munkafolyamat, amely nem tartalmaz interaktív elemeket, sokszor mégis azt tapasztalhatjuk, hogy az interaktivitás egy további pozitív hozadéka lehet a doodlingnek. Feltevődik a kérdés azonban, hogy mitől interaktív egy vázlat elkészítése. Kirsh az olvasást hozza fel példának a téma kapcsán, amelyről úgy vélekedik, mint önmagában egy nem interaktív folyamat, mely mégis aktívan hathat az olvasóra. Olvasni, aláhúzni, vagy összefoglalni egy szöveget már igencsak interaktív lehet hiszen kialakul egy oda-vissza kapcsolat az olvasó és az olvasott szöveg között [10]. Éppen ezért a rendkívüli szemléltetések nagymértékben hozzájárulhatnak a megértéshez és a tanulási folyamat érdekesebbé tételéhez. Mivel egy feladat nyomon követése az a teljes folyamat, amely magával vonhatja a doodling jelenséget [17] azt is elmondhatjuk, hogy ennek alkalmazása az oktatásba sokkal vonzóbbá teheti a tananyag ismertetését és magát a tanulást is, hiszen közvetlen módon vonja be a diákokat az oktatási folyamatba [7].

2.4. Kérdés- és doodling típusok

Gyakran feltevődik azonban a kérdés, hogy miért jó a doodling, miért nem elégséges az, ha csak ülünk és gondolkodunk a helyes megoldáson. A doodling, a gondolatok papírra való leírása nagymértékben függ a feladat típusától is. Gyakran tapasztalhatjuk, hogy amennyiben a megértés automatikus, nem foglalkozunk a megoldásmenet leírásával, azonban komplexebb kérdések esetén szinte minden esetben benne van a zsigereinkben, hogy papírt és ceruzát ragadjunk [10]. Mindez nem meglepő, hiszen az emberi gondolkodás egy „kognitív operációs rendszernek" tekinthető amely formákra, állapotokra és struktúrákra van tervezve [18].

Az a tény, hogy egy hallgató hogyan dolgoz fel egy problémát vagy hogyan közelíti meg az adott feladatot, nagymértékben befolyásolhatja a doodle típust, amit választ. Ennek kapcsán a Leeds kutatócsoport 12 doodle kategóriát határozott meg: váltakozó válasz (alternate answer), üres (blank page), számítás (computation), idegen nyomok, jelzések (extraneous marks), számvetés (keeping tally), szám (number), furcsa nyom (odd trace), pozíció (position), (szinkronizált nyomkövetés) synchronized trace, (nyomkövetés) trace, (aláhúzás) underlined, kizáró (ruled out) [9].

A kiválasztott doodle típust tehát nagymértékben befolyásolja a kérdések típusa is. Ahogy azt McCartney és kutatótársai is megfogalmazzák tanulmányukban, adott kérdések esetén bizonyos megjegyzések hatékonyabbak, mint mások, de bármilyen problémáról is legyen szó a megjegyzések jelenléte mindenképp jobb, mint az, ha egyáltalán nincsenek jelen annotációk. Ugyancsak ők tapasztalták azt, hogy kód-olvasással kapcsolatos problémák esetén a konkrét kóddal (fixed-code) kapcsolatos kérdéseknél, amelyek egy adott kódrészlet eredményét várták megoldásul a diákok többet "firkáltak", mint a kód-vázlatokkal (skeleton-code) kapcsolatos kérdések során [19].

Az említett szakirodalmi kutatások rávilágítanak arra, hogy egy algoritmus stratégia megértése nagymértékben függ a hallgatók problémamegoldó képességétől, mely szoros összefüggésben van azzal, hogy a hallgatók milyen mértékben követik nyomon papíron is az algoritmus lépéseit. Jelen kutatásunkban arra voltunk kíváncsiak, hogy hogyan teljesítenek a hallgatók két rendező algoritmus esetén, figyelembe véve a vizualizációk típusát, a doodling mértékét és a résztvevők előzetes programozási tapasztalatát is.

3. Kutatási kérdések

Az előzetes szakirodalmi kutatások alapján azt feltételeztük, hogy az algoritmusok típusa és a feladatok nehézsége befolyásolja a hallgatók doodlingre való hajlamát. Továbbá, úgy gondoltuk, hogy eltérő lehet az erre való hajlam és annak minősége kezdő és haladó programozási tapasztalattal rendelkező résztvevők esetén.

Ezek kapcsán a következő kutatási kérdéseket fogalmaztuk meg:

- Hogyan befolyásolja az algoritmusok típusa és a kérdések nehézségi szintje a hallgatók doodling technikáit?
- Milyen hatással van a hallgatók előzetes programozási tapasztalata a doodlingre?
- Mi az, ami befolyásolja a hallgatók "firkálásra" való hajlamát?

4. Módszertan

A kísérlet megvalósítására a 2022/2023-as tanév első félévének regisztrációs (előkészítő) hetén, a Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhelyi karán került sor. A felmérésen összesen 229 elsőéves egyetemi hallgató vett részt. A kutatás egy előzetes kérdőívet, két algoritmus bemutatását és ezekhez tartozó utótesztet foglalt magába.

4.1. Résztvevők

A kísérleten összesen 239 hallgató vett részt, melyek közül 12 nem töltötte ki megfelelően mindkét kérdőívet vagy nem vett részt a kísérlet második fázisán, így a végső eredmények meghatározásához 227 (30% lány) hallgató válaszait dolgoztuk fel. A résztvevők csoportját a Sapientia Erdélyi Magyar Tudományegyetem Marosvásárhelyi Kar elsőéves hallgatói alkották, a következő egyetemi szakokról: Automatika és alkalmazott informatika, Fordító és tolmács, Gépészmérnöki, Informatika, Kertészmérnöki, Kommunikáció és közkapcsolatok, Közegészségügyi szolgáltatások és politikák, Mechatronika, Számítástechnika, Tájépítészet és Távközlés.

A résztvevőket az előzetes kérdőív alapján három kategóriába soroltuk az előzetes programozási tapasztalatot illetően: **nincs előzetes programozási tapasztalat** (PT_0: egyáltalán nem tanult programozást a középiskolás évek során), **alap programozási tapasztalattal** ($PT_{1\text{-}3}$: 1, 2 vagy 3 évet tanult programozást a középiskolás évek során, természettudomány osztály diákjai; heti 1-2 programozás óra; vagy osztály változtatás esete), és **magas programozási tapasztalattal** (PT_4: 4 évet tanult programozást a középiskolás évek során, matematika informatika osztály diákjai; heti 5-7 programozás óra) rendelkező diákok.

A kísérlet során három csoportot határoztunk meg a vizualizáció típusa függvényben: Animáció, Tánc, Színészi játék. A három csoportba véletlenszerűen osztottuk el a hallgatókat úgy, hogy a fent említett programozási kategóriák mindegyikéből egyforma arányba kerüljenek résztvevők.

4.2. Kutatási eszközök

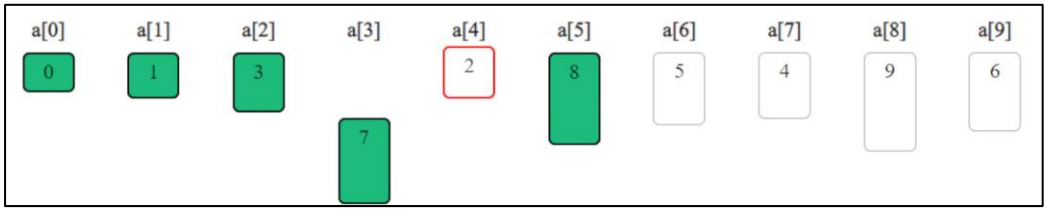
A kísérlet három nagy részre osztható fel: egy előzetes kérdőív, tanulási fázis és egy utóteszt. A kutatás során kitöltendő előzetes kérdőív és utóteszt is Google kérdőív segítségével került megvalósításra, míg a tanulási fázis videó szemléltetésével.

4.2.1. Előteszt

Az előzetes kérdőív összesen 10 kérdésből állt: 1 személyes adatok feldolgozására vonatkozó kérdés (GDPR), 2 demográfiai adatok feldolgozására vonatkozó kérdés, 2 szakterületre vonatkozó kérdés, 4 érettségi eredményekre vonatkozó kérdés és 1 előzetes programozási tapasztalatra vonatkozó kérdés (Hány évet tanultál programozást középiskolában?).

4.2.2. Tanulási fázis

A tanulási fázis két algoritmus bemutatását foglalta magába. A bemutatás videók vetítése segítségével valósult meg. Minden vizualizáció típushoz (animáció, tánc, színészi játék) algoritmusonként (beszúró-, shell rendezés) tartozott egy felvezető videó (tanár által szemléltetett bemutatás), egy algoritmus stratégiát bemutató videó, és végül egy záró videó (tanár által szemléltetett bemutatás). A második algoritmus jellegénél fogva, mivel egy nehezebb rendezési stratégiát mutatott be még kiegészült egy segítségként szolgáló videóval is.



1. ábra: Algoritmus reprezentáció animációval

2. ábra: Algoritmus reprezentáció tánccal

3. ábra: Algoritmus reprezentáció színészi játékkal

4.2.3. Utóteszt

Az utóteszt kérdései 4 felvonásra voltak osztva melyre összesen 18 pontot lehetett összegyűjteni:

Az első felvonás 1 demográfiai adatokra vonatkozó kérdést és egy szubjektív véleményhez tartozó kérdést („Te melyik vizualizációval tanulnál legszívesebben?") foglalt magába.

A második felvonás az első rendezési algoritmusra, a beszúró rendezésre vonatkozó kérdéseket tartalmazta: 2 alapműveletre, 3 belső műveletre és 4 algoritmus bonyolultságra vonatkozó kérdést.

A harmadik felvonás a második rendező algoritmushoz, a shell rendezéshez tartozó kérdéseket tartalmazta. Az előző algoritmushoz hasonlóan ebben az esetben is 2 alapműveletre, 3 belső műveletre és 4 algoritmus bonyolultsági kérdésre vártuk a hallgatók válaszát.

Végezetül, a negyedik felvonás néhány záró kérdést tartalmazott, amelyek során a résztvevők szubjektív véleményére voltunk kíváncsiak az algoritmus megértésével, az ábrázolásmód érdekességével, valamint a tanulási élménnyel kapcsolatosan. Ezek a kérdések nem számítottak bele a végső pontozásba.

A felvonások és az azokhoz tartozó kérdések részletes leírását és típusát az alábbi táblázat szemlélteti.

Kérdés száma	**Kérdés leírása**	**Kérdés típusa**
Első felvonás kérdései		
1	Név	nyílt
2	Melyik vizualizációval tanulnál a legszívesebben? (animáció, tánc, színészi játék)	egyválaszos
Második felvonás kérdései (beszúró rendezés)		
A következő 7 elemű számsorozaton: 1, 19, 7, 8, 12, 11, 9		
1	Melyik két szám kerül először összehasonlításra?	nyílt
2	Melyik két szám kerül először kicserélésre?	nyílt

3	Milyen művelet következik a 19-es és 12-es számok összehasonlítása után?	Egyválaszos
4	Melyik két szám kerül összehasonlításra a 19-es és 11-es számok összehasonlítása után?	nyílt
5	Melyik két szám kerül összehasonlításra a 11-es és 9-es számok összehasonlítása után?	nyílt
Algoritmus bonyolultsági kérdések		
6	Ha már eredetileg szigorúan növekvő sorrendben van egy 7 elemű számsorozat, hány összehasonlításra kerül sor abban az esetben, ha a számsorozatot növekvő sorrendbe rendezzük?	nyílt
7	Ha már eredetileg szigorúan növekvő sorrendben van egy 7 elemű számsorozat, hány cserére kerül sor abban az esetben, ha a számsorozatot növekvő sorrendbe rendezzük?	nyílt
8	Ha már eredetileg szigorúan csökkenő sorrendben van egy 7 elemű számsorozat, hány összehasonlításra kerül sor abban az esetben, ha a számsorozatot növekvő sorrendbe rendezzük?	nyílt
9	Ha már eredetileg szigorúan csökkenő sorrendben van egy 7 elemű számsorozat, hány cserére kerül sor abban az esetben, ha a számsorozatot növekvő sorrendbe rendezzük?	nyílt
Harmadik felvonás kérdései (shell rendezés)		
A következő 7 elemű számsorozaton: 1, 19, 7, 8, 12, 11, 9		
1	Melyik két szám kerül először összehasonlításra?	nyílt
2	Melyik két szám kerül először kicserélésre?	nyílt
3	Milyen művelet következik a 7-es és a 11-es számok összehasonlítása után?	egyválaszos
4	Melyik két szám kerül összehasonlításra a 8-as és 9-es számok összehasonlítása után?	nyílt
5	Melyik két szám kerül összehasonlításra a 7-es és 8-as számok összehasonlítása után?	nyílt
Algoritmus bonyolultsági kérdések		
6-9	A beszúró rendezéshez hasonlóan 4 algoritmusra vonatkozó bonyolultsági kérdés.	nyílt kérdések
Negyedik felvonás kérdései (szubjektív vélemények)		
1	Milyen mértékben járult hozzá az ábrázolási mód az algoritmus megértéséhez?	skála
2	Mennyire kötötte le az ábrázolási mód a figyelmedet?	skála
3	Milyen mértékben járult hozzá az ábrázolási mód a tanulási élményhez?	skála

1. táblázat: Utóteszt kérdései

4.3. A kutatás menete

A kutatás menetét tekintve az előzetes kérdőív kitöltésére a felmérés előtti nap került sor. A résztvevőknek 8 óra állt rendelkezésükre kitölteni a kérdőívet. Ezt követően a beírt adatoknak megfelelően a

diákokat három, a programozási előismereteknek megfelelően kiegyensúlyozott csoportba osztottuk. Másnap a résztvevők ennek a beosztásnak megfelelően helyezkedtek el az egyetem három különböző előadótermében, ahol egy-egy oktató fogadta őket. A három terem három különböző vizualizációval tanult: animáció, tánc, színészi játék.

A kísérlet mindhárom teremben egyforma módon zajlott. A résztvevőknek két algoritmus vizualizációt kellett megtekinteniük, a típusnak (animáció, tánc, színészi játék) megfelelő módon. Mindkét algoritmus stratégia bemutatása kétszer került vetítésre. Minden algoritmusvizualizáció egy bevezető videóval kezdődött, és egy záró videóval végződött, mely minden csoport esetén egyforma volt. Az utóteszt különböző felvonásainak kérdéseire a diákoknak megszakításokkal kellett válaszolniuk. Minden felvonás, az adott videó megtekintése után került megválaszolásra. A shell rendezési stratégia esetén minden csoport megtekinthetett egy kiegészítő videót is, lévén egy nehezebb rendezési stratégiáról szó.

A kutatás menetének részletes beosztását az alábbi táblázat szemlélteti:

Cs_1: Animáció	**Cs_2: Tánc**	**Cs_3: Színészi játék**
Utóteszt - Első felvonás kérdéseinek megválaszolása		
Bevezető videó - beszúró rendezés		
Beszúró rendezés videójának megtekintése **animációval** X2	Beszúró rendezés videójának megtekintése **tánccal** X2	Beszúró rendezés videójának megtekintése **színészi játékkal** X2
Utóteszt - Második felvonás kérdéseinek megválaszolása (beszúró rendezés)		
Bevezető videó - shell rendezés		
Shell rendezés videójának megtekintése **animációval** X2	Shell rendezés videójának megtekintése **tánccal** X2	Shell rendezés videójának megtekintése **színészi játékkal** X2
Kiegészítő videó - shell rendezés		
Utóteszt - III. felvonás kérdéseinek megválaszolása (shell rendezés)		
A kísérlet záró videója		
Utóteszt - IV. felvonás kérdéseinek megválaszolása (szubjektív vélemények összegyűjtése)		

2. táblázat: A kutatás menete

A kísérlet elején a résztvevők tudtára adtuk, hogy a kérdések során bármikor kérhetnek piszkozat lapokat, vagyis ezeket nem adtuk oda mindenkinek a kérdőív kitöltése előtt, csak abban az időpillanatban amikor igényelte az illető személy.

5. Eredmények

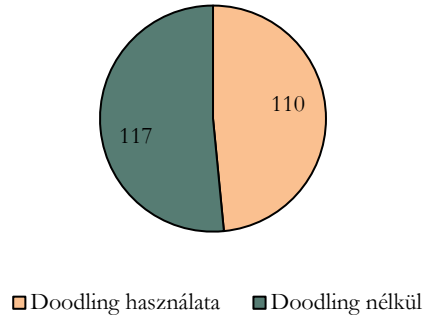
A kísérlet során több szempontot is vizsgáltunk, ami a résztvevők algoritmikus gondolkodását és problémamegoldó képességét illeti. Ebben a tanulmányban a piszkozat használatára, vagyis a doodling hatására fókuszáltunk, melyet a következő szempontok szerint vizsgáltunk: a résztvevők előzetes programozási tapasztalata, neme, a vizualizáció típusa és az alkalmazott doodling technikák.

5.1. A doodling hatása a teljesítményre

Elsőként kíváncsiak voltunk arra, hogy miként befolyásolja a hallgatók eredményét, vagyis végső pontszámát az, hogy használnak piszkozatot vagy sem. Amint azt már korábban is említettük, a kísérlet

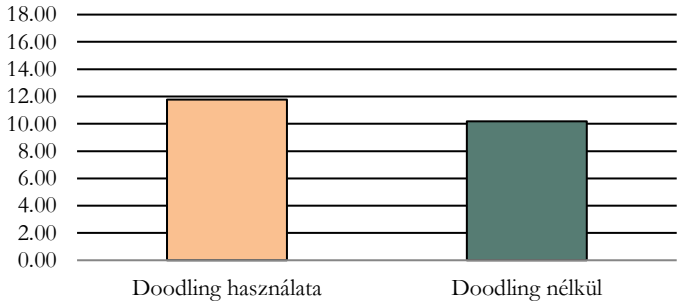
során teljesen fakultatív dolognak számított a piszkozatlap kérése. Ezeket csak azon hallgatóknak osztottuk ki, akik igényelték.

Ahogy azt a **4. ábra:** Piszkozatok használatának eloszlása is szemlélteti, a résztvevők szinte fele-fele arányban kértek piszkozatot (110 hallgató) vagy anélkül dolgoztak (117 hallgató).



4. ábra: Piszkozatok használatának eloszlása

Az eredmények feldolgozását követően a két csoport látszólag szinte teljesen egyformán teljesített (ábra). Azok a résztvevők, akik piszkozatot használtak a maximális 18 pontból átlagosan 11.77 pontot gyűjtöttek össze, míg azok, akik piszkozat nélkül, átlagosan 10.18 pontot (**5. ábra**). Páros T próba alkalmazása után arra a következtetésre jutottunk, hogy a piszkozatot igénylő hallgatók szignifikánsan jobban teljesítettek, mint a piszkozat nélküliek ($p=0<0.005$). Ez a jelenség várható volt, hiszen számos kutató arról számol be, hogy bár a piszkozatra leírt megjegyzések típusa eltérő lehet, egy dolog bizonyos: a doodling alapjáraton elősegíti a helyes válasz meghatározását [5], [19]. Hasonlóképpen, további kutatócsoportok [7], [9] is kihangsúlyozzák azt a jelenséget kód olvasási problémák kapcsán, hogy azok, akik különféle nyomkövetési technikákat alkalmaznak, mint például a doodling, jobb teljesítményt érnek el.

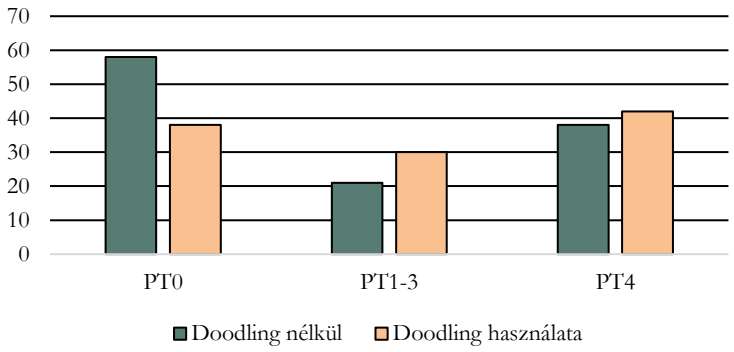


5. ábra: Doodling hatása az eredményekre

5.2. A doodling és az előzetes programozási tapasztalat közötti összefüggés

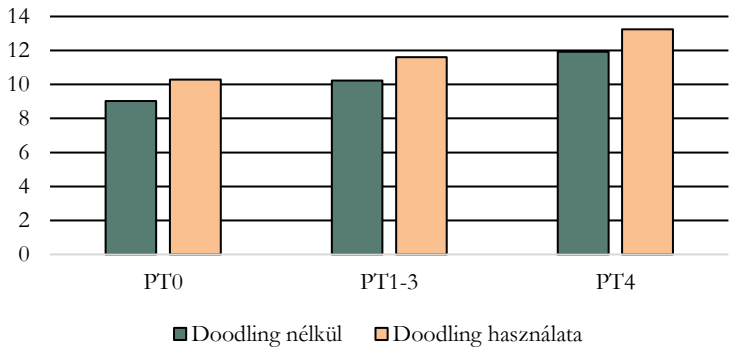
A továbbiakban kíváncsiak voltunk arra is, hogy milyen mértékben befolyásolja a résztvevők előzetes programozási tapasztalata a doodlingre való hajlamot. Az eredmények feldolgozását követően azt tapasztaltuk, hogy a több programozási tapasztalattal rendelkező hallgatók kategóriája a piszkozatok igénylési arányát tekintve felülmúlta a kezdőkét (PT_0: 96 diákból 38, $PT_{1\text{-}3}$: 51 diákból 30, PT_4: 80 diákból 42

igényelt piszkozatot). Érdekes módon csak a PT_0 kategóriára volt igaz az a jelenség, hogy többen nem kértek piszkozatot, mint ahányan kértek (**6. ábra**).



6. ábra: Doodling használata programozási tapasztalat szerint

A kérdésekre adott válaszok feldolgozását követően itt is várható volt az eredmény: azon diákok, akik piszkozatot igényeltek minden esetben (PT_0, PT_{1-3} és PT_4 esetén is) jobban teljesítettek, mint azon társaik, akik nem (**7. ábra**). Mi több, a kezdő (P_0) és haladó (PT_4) programozási tapasztalattal rendelkezők kategóriájában is szignifikáns különbséghez jutottunk (P_0: $p=0.04<0.05$; P_4: $p=0.03<0.05$).



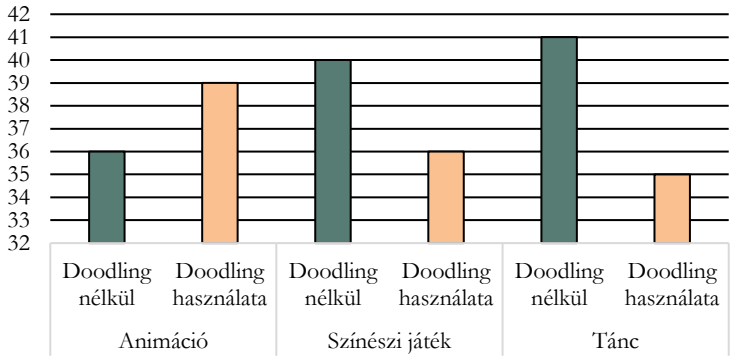
7. ábra: A doodling hatása az eredményekre programozási tapasztalat szerint

Eredményeink arra engedtek következtetni, hogy a haladók sokkal nagyobb valószínűséggel kezdenek el piszkozatot használni, mint a kezdők, melyet Whalley és kutatótársai [5] is megfogalmaztak. Ennek a jelenségnek Perkins és kollégái szerint akár több oka is lehet, amely a doodlinghez kapcsolható, mint például az, hogy sok hallgató nem érti miért is lehet hasznos számára az ilyen jellegű nyomon követése a feladatnak, vagy egyszerűen nem magabiztosak abban, hogy helyesen tudnának jegyzetelni [20]. Úgy véljük ez a bizonytalanság a mi esetünkben is hatással lehetett a kezdők piszkozat igénylési hajlamára.

5.3. A vizualizáció típusának hatása a doodlingre

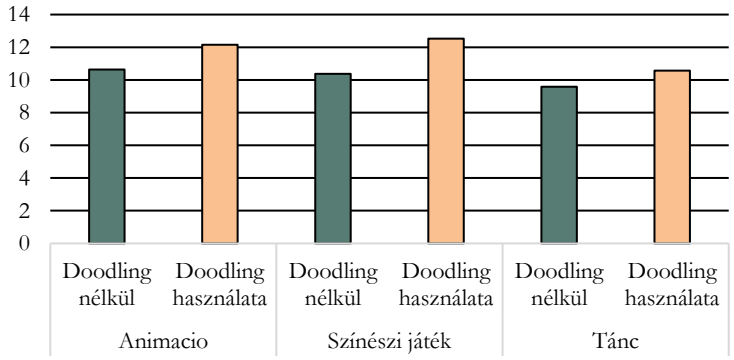
Arra is kíváncsiak voltunk, hogy miként befolyásolja az adott vizualizáció típusa azt, hogy a diákok használnak-e piszkozatot vagy sem. Hasonlóképpen, először összesítettük, hogy hány piszkozat volt igényelve külön-külön a három csoportban (**8. ábra**). Az animációval tanuló hallgatók közül a 75 hallgatóból összesen 39-en, a színészi játék esetén a 76 hallgatóból 36-on, míg a tánccal bemutatott algoritmusvizualizáció esetén 76 hallgatóból 35-ön alkalmaztak doodlinget a feladatmegoldás során. Érdekes módon, csak az animációs csoportra volt igaz az a kijelentés, hogy többen használtak piszkozatot, mint sem.

Ennek oka az is lehet, hogy az animáció absztrakt jellege miatt jobban ösztönzi a diákokat abban, hogy piszkozatot ragadjanak és nyomon kövessék az algoritmus lépéseit. Ehhez hasonló okot fogalmazott meg Cunningham és munkatársai is, akik szerint az ábrák, formák több tantárgy esetén is, mint például matematika, fizika, kémia, magukkal vonják és ösztönzik a diákokat a piszkozatírásra [7].



8. ábra: Doodling használata az algoritmusvizualizáció típusa szerint

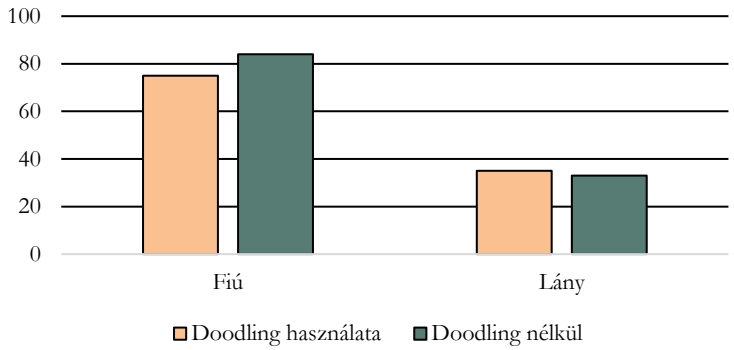
A résztvevők pontszámát figyelembe véve a következőképpen alakultak az átlagok: animációs csoportban (doodling használata: 12.15, doodling nélkül: 10.63), színészi játék csoportban (doodling használata: 12.52, doodling nélkül: 10.37) és tánccal bemutatott vizualizáció csoportban (doodling használata: 10.57, doodling nélkül: 9.58). Várható módon, mindhárom csoport esetén a piszkozatot használt hallgatók átlagos teljesítménye jobb volt, mint azon diákoké, akik nem használtak piszkozatot. Az animáció és színészi játék csoportokban a doodling szignifikánsan magasabb átlageredményekhez (animáció: $p=0.04<0.05$, színészi játék: $p=0.002<0.005$) vezetett (**9. ábra**).



9. ábra: Doodling hatása az eredményekre az algoritmusvizualizáció típusa szerint

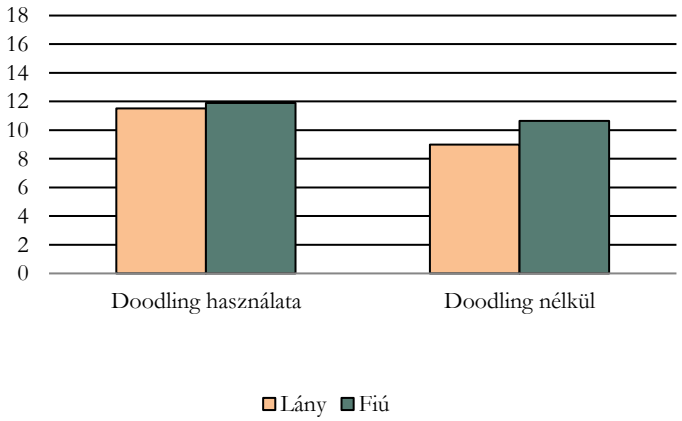
5.4. Nemek

Bár a diákok nemek szerinti eloszlását tekintve jelentősen kevesebb lány vett részt a kísérleten (30%) mégis kíváncsiak voltunk, hogy milyen mértékben jellemző a lányokra, illetve a fiúkra az, hogy doodlinget alkalmazzanak a feladatmegoldás során. A piszkozatok számát tekintve szinte fele-fele arányban kértek piszkozatot úgy a lányok (doodling: 51%), mint a fiúk (doodling: 47%).



10. ábra: Doodling használata nemek szerint

A pontszámokat tekintve (**11. ábra**) a doodlinget alkalmazó lányok 11.51pontot szereztek átlagosan, mellyel szignifikánsan jobban teljesítettek ($p=0.003<0.005$), mint azok a lányok, akik nem használtak piszkozatot (9 pont). A fiúk esetén is hasonló eredményhez jutottunk, hiszen a doodlinget alkalmazó fiúk (11.89 pont) szignifikánsan jobban teljesítettek ($p=0.01<0.05$), mint társaik (10.64 pont). Az előző eredményeink és a szakirodalmi kutatások alapján mindez várható volt, ezért azt is szerettük volna megvizsgálni, hogy milyen különbségekhez vezet az, ha összemérjük a fiúk és lányok eredményeit. Bár külön elemezve a fiúk és lányok eredményeit szignifikáns különbségekhez jutottunk, összehasonlítva ezeket lényegesen eltérő különbség csak a piszkozat nélküli kategóriákban volt észlelhető ($p=0.01<0.05$), míg doodlinget alkalmazó lányok nem maradtak le lényegesen az ugyancsak doodlinget alkalmazó fiúktól ($p=0.29>0.05$).

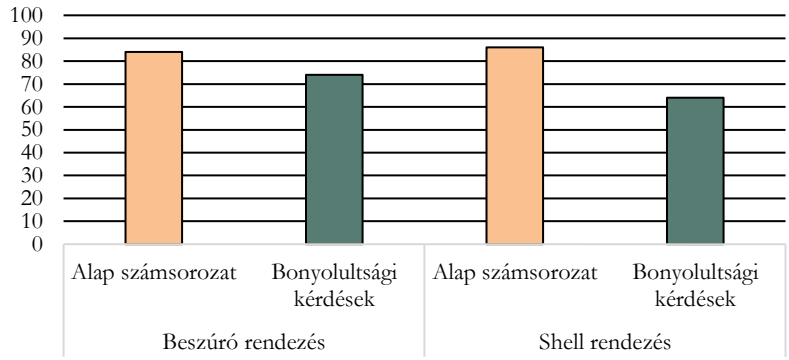


11. ábra: Doodling hatása az eredményekre nemek szerint

Mindez arra tud ösztönözni, hogy bár nyilvánvalónak bizonyul a piszkozat hasznossága, ennek kettős pozitívuma is lehet, ha a nemek eloszlását tekintjük. A lányoknak kifejezetten segítségére lehet az, ha levezetik az algoritmusok lépéseit, mely által csökkenthető a gyakran észlelhető szakadék fiúk és lányok teljesítménye között.

5.5. A doodling és a kérdéstípusok közötti összefüggés

Végül, de nem utolsó sorban szerettük volna közelebbről is megvizsgálni a piszkozatokat és elemezni a hallgatók által írt megjegyzéseket és firkákat algoritmusok, valamint kérdések szerint is (**12. ábra**). A kérdések összetételét tekintve mindkét algoritmus (beszúró- és shell rendezés) esetén két kategóriát vettünk figyelembe: (1) alap számsorozattal kapcsolatos kérdések, (2) algoritmus bonyolultsági kérdések. A doodling számát tekintve a 227 hallgatóból összesen 44 hallgató használta minden feladat esetén a piszkozatot. Külön-külön a két algoritmus esetén pedig összesen 158 (beszúró rendezés: alap számsorozat - 84, bonyolultsági kérdések - 74), illetve 150 (shell rendezés: alap számsorozat - 86, bonyolultsági kérdések - 64) hallgató használt piszkozatot.



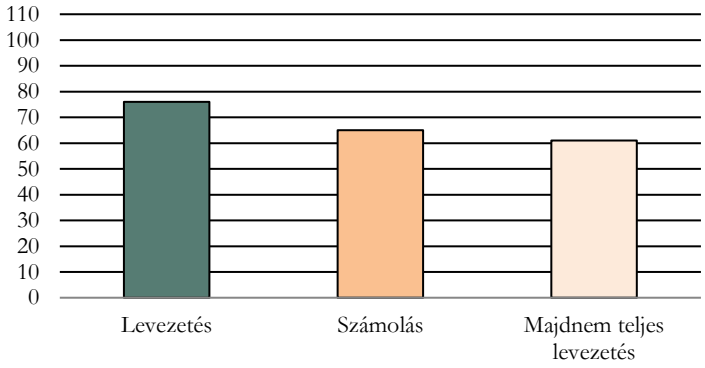
12. ábra: Doodling használata kérdéstípusok szerint

Bár Kirsh és kutatótársai úgy tapasztalták, hogy a komplex problémák nagyobb eséllyel vonzzák a piszkozat használatát [10], ez a jelenség nálunk nem igazolódott be. Érdekes módon, a beszúró rendezés kapcsán többen firkálták le a megoldásmenetet, mint a shell rendezés esetén, amely egy jellegében nehezebb algoritmusnak számít. Ennek ellenére sem az alap számsorozattal kapcsolatos feladatok, sem az algoritmus bonyolultsági kérdések esetén nem volt szignifikáns különbség a két algoritmusra kapott pontszámok között (alap számsorozat: $p=0.32>0.05$, bonyolultsági kérdések: $p=0.08>0.05$, összességében: $p=0.22>0.05$).

Annak oka, hogy kevesebb doodling volt a shell algoritmus során az is lehet, hogy a hallgatók nem értették az algoritmust és ez a tudáshiány, bizonytalanság ahhoz vezetett, hogy nem is kezdték el az algoritmus lépéseit papíron ábrázolni [19]. Az általunk mért eredmények bizonyos mértékben harmonizálnak McCartney és társai következtetésivel, akik kutatásukban megfogalmazták, hogy a piszkozat hiányának a hallgatók bizonytalansága mellett számos más okozója is lehet, mint például az, hogy a hallgatók nem rendelkeznek elegendő ismerettel a probléma megoldásához, vagy gyenge a résztvevők problémamegoldó képessége [19].

A fent említett jelenség az alap számsorozattal és bonyolultsági kérdések kapcsán is érzékelhető volt. Bár a bonyolultsági kérdések jelentősen több ismeretet és jó problémamegoldó képességet igényelnek, a piszkozathasználat ebben az esetben is háttérbe szorult, míg az alap feladatok esetén nem. Ennek egy másik oka a kérdés típusából adódó jelleg is lehet. Ahogy azt McCartney és kutatótársai is megfogalmazták a kérdések milyensége jelentősen befolyásolhatja azt, hogy a hallgató hogyan teljesít, illetve azt is, hogy milyen megjegyzéseket, firkákat jegyez le a papírra [19]. Hasonlóképpen, Whalley és társai is kihangsúlyozták tanulmányukban azt, hogy bizonyos kérdéstípusok, mint például a "fixed-code" jellegű kérdések jobban előídézik a doodle használatát [5]. Ehhez hasonlóan, a mi kísérletünk során is több megjegyzés és firka érkezett az alap számsorozatra vonatkozó kérdésekre ("fixed-sequence"), mely azonosítható a "fixed-code" típusú feladatokkal.

Ezt a jelenséget a **13. ábra** is megerősíti, mely segítségével a levezetések és számolások közötti összefüggéseket szerettünk volna szemléltetni. A levezetések (76 hallgató) rendszerint az alap számsorozatra, míg a számolások az algoritmus bonyolultsági kérdésekre (65), vagyis az általánosításra vonatkozó feladatoknál voltak hasznosak.

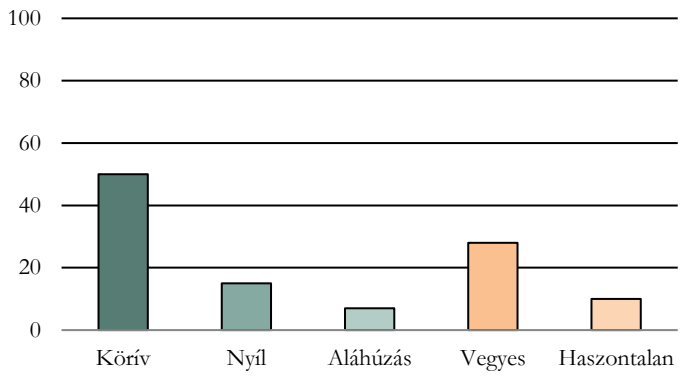


13. ábra: Levezetések és számolások

Érdekességképp, az is megemlítendő, hogy a 110 piszkozatot használó hallgató közül több, mint fele (61 hallgató) vezette le teljesen, vagy majdnem teljes egészében a két algoritmus lépéseit a piszkozatlapra.

A McCartney kutatócsoporthoz hasonlóan [19], mi is meghatároztunk 5 doodling kategóriát, melyet vizsgáltunk a résztvevők piszkozatlapjain: (1) körív: rendszerint két szám összehasonlításának/cseréjének szemléltetésére, (2) nyíl: rendszerint az algoritmus egy új állapotának meghatározására, (3) aláhúzás: rendszerint a számsorozatban szereplő aktív értékek szemléltetésére, (4) vegyes: az előzőek közül több együttes használata, (5) zavaros: rendezetlen, következetlen piszkozat, kihúzásokkal, összefüggéstelen jelekkel (**14. ábra**).

Érdekes módon, bár a résztvevők többségének a különböző algoritmus lépéseit szemléltető jelek ismeretlenek voltak, előszeretettel használták a körív (50 hallgató) jelet két elem összehasonlításakor/cseréjekor. Az aláhúzás (7 hallgató), lévén egy ritkább jelölési forma, mely inkább azok számára volt ismert, akik tanultak már programozást jelentősen kevesebb piszkozatlapon volt látható, míg a nyíl jelzés 15 hallgató piszkozatán. Emellett, összesen 28 résztvevő használta vegyesen a fent említett jeleket, míg 10 hallgató piszkozata zavarosnak, következetlennek bizonyult. Fontos megemlíteni, hogy jelen kísérlet során a hallgatóknak igényelni kellett a piszkozatot. Ennek értelmében, összesen 117 hallgató nem is használt doodlinget a felmérés során, melyet asszociálhatunk az úgynevezett üres megjegyzésekkel.



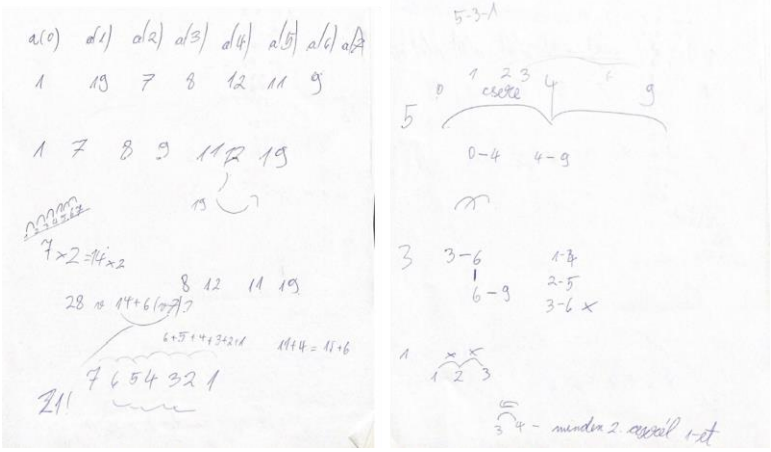
14. ábra: Jelek használata a piszkozaton

5.6. Különleges - Haszontalan piszkozatok

Ebben a részben szeretnénk néhány számunkra különlegesnek, illetve zavarosnak minősített piszkozatot szemléltetni. Fontos megjegyezni, hogy a 110 piszkozat közül több is különlegesnek nevezhető, ezek közül azonban csak néhányat fogunk bemutatni.

Az alábbi doodling technikákat azért tartottuk különlegesnek, mert felhasznált jelzések (nyíl, körív stb) mellett, fellelhetőek további rendszerezettségre és következetességre utaló jelek is.

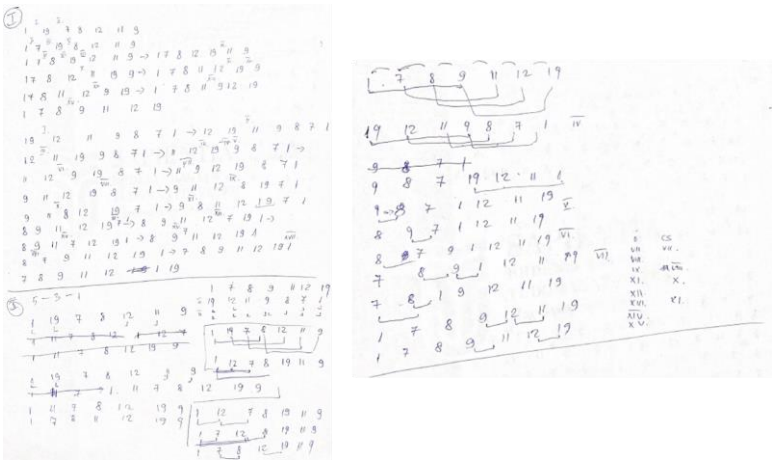
Az 1. piszkozat esetén (**15. ábra**) a hallgató szemléltette a tömb (a[0], a[1].. a[9]) adatszerkezetet is annak ellenére, hogy 0 év programozási tapasztalattal rendelkezett. Emellett, tömör megjegyzéseket is írt a lapjára. Ennek ellenére érdekes módon összesen 3 pontot kapott a tesztre.



15. ábra: 1. piszkozat - PT_0, pontszám: 3

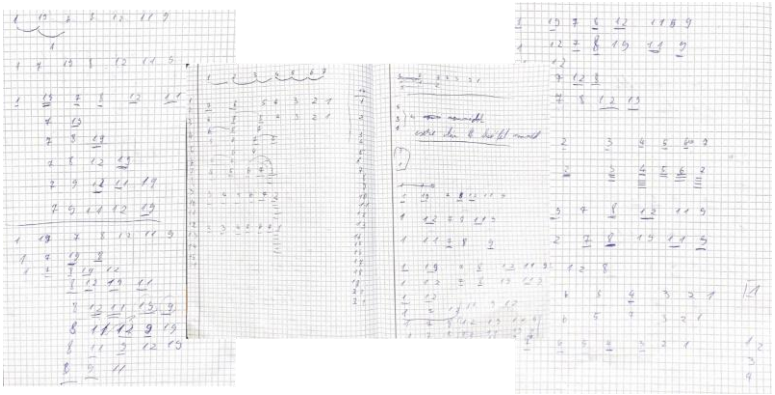
A 2. piszkozat (**16. ábra**) az arab számok mellett római számokat is használt a két algoritmus elkülönítésére, valamint az algoritmus lépéseinek nyomon követésére is. Az is megfigyelhető, hogy az összehasonlítások és cserék nyilak, valamint körívek felhasználásával egyaránt szemléltetve vannak.

Hasonlóképpen, ez a hallgató is 0 programozási tapasztalattal rendelkezett a teszt kitöltésekor, ennek ellenére 13 pontot sikerült összegyűjtenie a teszten.



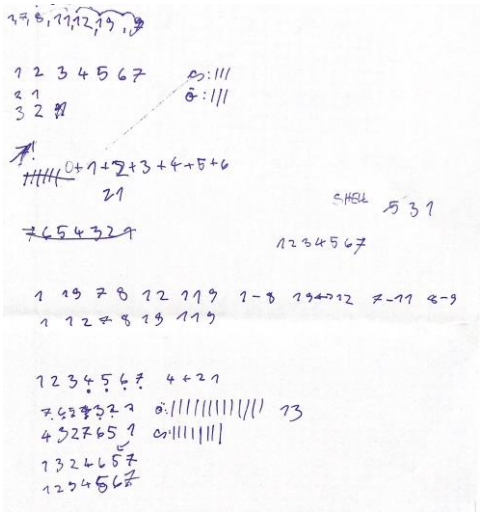
16. ábra: 2. piszkozat - PT_0, pontszám: 13

A 3. piszkozat (**17. ábra**) az egyik leghosszabbnak nyilvánított piszkozat volt, ahol részletesen le volt vezetve az algoritmus minden lépése. A piszkozat különlegessége abban is megmutatkozik, hogy egyszeres, illetve többszörös aláhúzásokkal vannak jelölve a számsorozatban szereplő értékek és a megjegyzések sem maradnak el. A piszkozat tulajdonosa 2 év előzetes programozási tapasztalattal rendelkezett a felmérés idején, és 16 pontot kapott a tesztre, amely igen jó eredménynek számít a maximálisan megszerezhető 18 pontból.



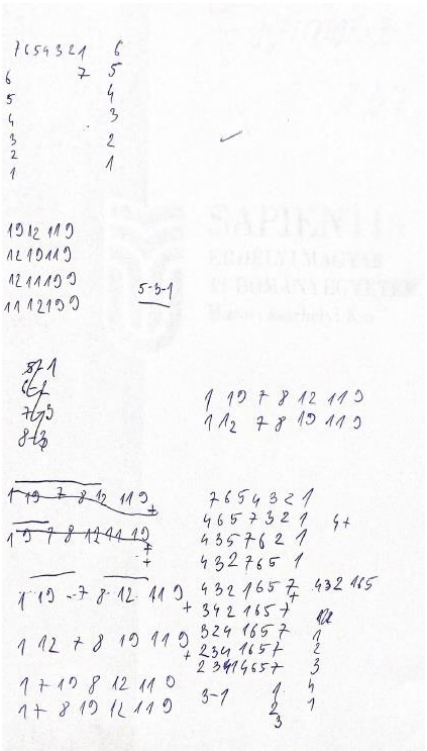
17. ábra: 3. piszkozat – $PT_{1\text{-}3}$, pontszám: 16

A 4. piszkozat (**18. ábra**) az előző háromhoz hasonlóan kevés, pontosabban 0 előzetes programozási tapasztalattal rendelkező hallgató tulajdona, aki a teszten ennek ellenére 15 pontot gyűjtött össze. A tömör doodling ellenére, sok helyes fázisa fedezhető fel az algoritmusnak, mely minden bizonnyal segítette a hallgatót a válaszadásban.

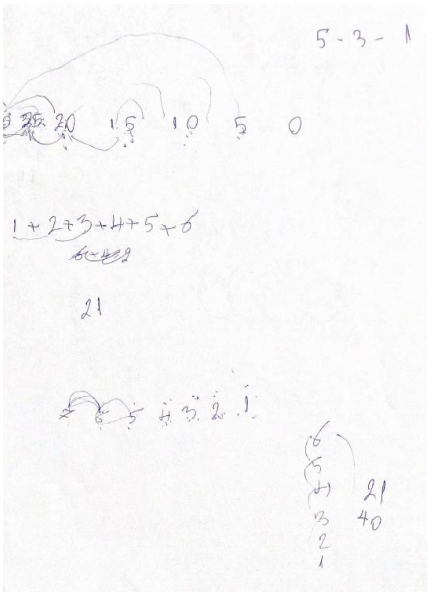


18. ábra: 3. piszkozat - PT_0, pontszám: 15

A zavaros vagy nem követhető piszkozatok kapcsán három példát szeretnénk szemléltetni. Az első két piszkozat mindegyike egy-egy 4 év előzetes programozási tapasztalattal rendelkező hallgató tulajdona (**19. ábra** és **20. ábra**). Érdekes módon, a piszkozat alapján kevés algoritmus stratégiájára vonatkozó rész fedezhető fel, a megoldásmenet is nehezebben követhető, sok esetben a számsorozat sem megfelelő. Ennek ellenére a hallgatók összesen 12, illetve 14 pontot szereztek a teszten.

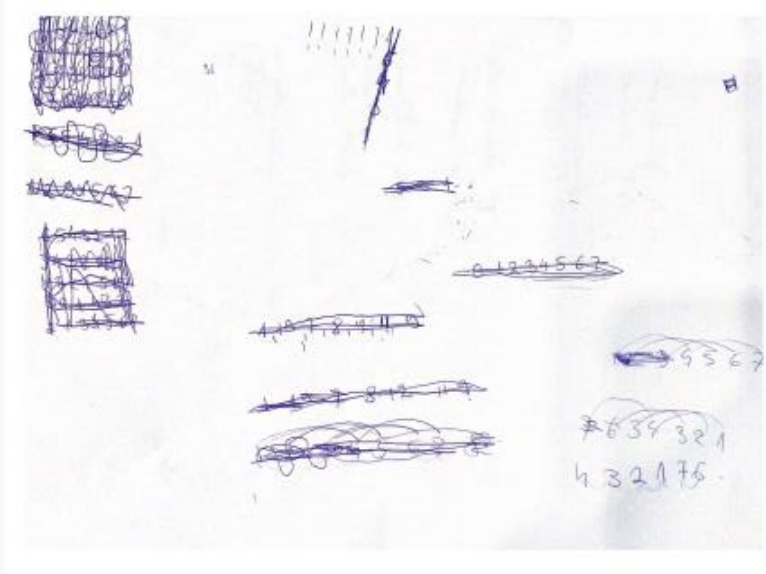


19. ábra: 1. piszkozat – PT_4, pontszám: 12



20. ábra: 2. piszkozat – PT_4, pontszám: 14

A harmadik zavarosnak nevezhető piszkozat (**21. ábra**) tulajdonosa ismeretlen, így az előzetes programozási tapasztalat mértékét és a pontszámot nem volt lehetőségünk meghatározni. Az azonban bizonyos, hogy a résztvevőnek minden próbálkozása ellenére nem sikerült megoldáshoz jutnia, vagy legalábbis a piszkozat szintjén nem volt látható az algoritmus lépéseinek nyomon követése.



21. ábra: 3. piszkozat – ismeretlen adatok

Bár a fent bemutatott piszkozatok csak töredékét képezik az összképnek, érdekesnek mondható az a tény, hogy sok 0 programozási tapasztalattal rendelkező hallgató készített rendszerezett piszkozatot, míg a magas programozási tapasztalattal rendelkező résztvevők közül sokan felületesen, zavarosan vetik papírra ötleteiket.

A pontszámok ennek ellenére nem feltétlen azonosíthatók a gondosan és rendszerezetten elkészített piszkozatokhoz. Sok esetben a felületes, zavaros és kódolt piszkozatok tulajdonosai szereztek nagyobb pontszámot.

6. Limitációk

A kísérlet lehetőséget biztosított arra, hogy felmérjük és megvizsgáljuk a résztvevők doodling technikáit különböző szempontok szerint. Kutatásunk egyik limitációja, hogy a kiosztott piszkozatlap nem volt feladatonként külön szegmensekre osztva. Ennek segítségével sokkal pontosabban és egyszerűbben tudtuk volna összesíteni a hallgatók eredményeit. Ugyancsak ehhez kapcsolódhat az a limitáció, miszerint a hallgatók utótesztje Google kérdőív keretén belül volt megvalósítva. Minden bizonnyal, egy papír alapú kérdőív újabb befolyásoló tényező lehet, ami a résztvevők piszkozatírási hajlamát illeti.

7. Összegzés

Jelen tanulmányban azt vizsgáltuk meg, hogy milyen a hallgatók doodling technikájának minősége, illetve milyen tényezők befolyásolják a piszkozatírásra való hajlamot.

Az eredmények és az előzetes szakirodalmi kutatások is egyértelműen arra utalnak, hogy a piszkozat önmagában jobb eredményekhez vezetnek. Legyen nemekről, előzetes programozási tapasztalatról vagy épp vizualizáció típusokról szó, azon diákok, akik használnak piszkozatot, összességében jobb eredményt érnek, mint azok, akik egyáltalán nem használnak.

A hallgatók doodlingre való hajlamát számos tényező befolyásolhatja. Az egyik ilyen tényező lehet a hallgatók előzetes programozási tapasztalata. Eredményeink arra engedtek következtetni, hogy a haladó programozási tapasztalattal rendelkező diákok nagyobb valószínűséggel ragadnak papírt és ceruzát annak érdekében, hogy nyomon kövessék az algoritmus lépéseit. Ennek egyik oka a kezdő programozási tapasztalattal rendelkező hallgatók bizonytalansága, vagy akár a probléma megoldó képesség hiánya is lehet.

A kísérlet során arra is választ kaptunk, hogy a feladatok és kérdések milyensége is nagy hatással lehet a doodlingre való hajlamra. Az utóteszt kérdései alapján azt tapasztaltuk, hogy az alapfeladatoknak számító, levezetést igénylő feladatok jobban igénylik a piszkozat használatát, mint a jellegüknél fogva elvont kategóriába tartozó kérdések.

Az ábrázolás tekintetében, arra utalnak az eredmények, hogy a leginkább az animációval bemutatott algoritmus vizualizáció idézi elő a piszkozatlap használatát. Ennek oka az animációban megtalálható ábrák, formák jelenléte és annak absztrakt jellege lehet, mely sokkal inkább társítható a szabad kézzel elkészített firkákhoz, mint a másik kép vizualizáció.

A tanulmány számos érdekes jelenségre is rávilágított, mint például arra, hogy sok résztvevő esetén az összehasonlítások és cserék műveletét szinte tudat alatt is a körívekkel azonosítják. Ennek kapcsán is érzékelhető, hogy a diákok mondhatni automatikusan kialakítják önmagukban piszkozatírási technikáikat, ez azonban nagyon sokszor felületesnek bizonyul.

Jelen dolgozat arra is felhívta figyelmünket, hogy fontos a hallgatók doodling technikáira is fókuszálni, hiszen ez nagymértékben befolyásolhatja a továbbiakban a programozás oktatásban való előrehaladásukat. Véleményünk szerint ezek a technikák kellőképpen fejleszthetők, ha a mindennapi informatika tanórákba helyet keresünk a debuggolás, nyomkövetés, program olvasás, program felismerés és hasonló jellegű feladatoknak.

8. Köszönetnyilvánítás

A jelen munkát Magyarország Collegium Talentum programja támogatta. Köszönjük továbbá a Kutatási Programok Intézetének támogatását is.

9. Irodalom

1. Kátai, Z., & Osztián, E. (2021). Improving AlgoRythmics Teaching-Learning Environment by Asking Questions. International Journal of Instruction, 14(2), 27-44.
2. Kátai, Z. (2021). Algorythmics. Technologically and artistically enhanced computer science education.
3. Nagy, E. J., Osztián, P. R., Cosma, C., Kátai, Z., & Osztián, E. (2019, June). Looking for the Optimal Interactivity Level in the AlgoRythmics Learning Environment. In EdMedia+ Innovate Learning (pp. 106-114). Association for the Advancement of Computing in Education (AACE).
4. Osztián, P. R., Kátai, Z., & Osztián, E. (2020, October). Algorithm Visualization Environments: Degree of interactivity as an influence on student-learning. In 2020 IEEE Frontiers in Education Conference (FIE) (pp. 1-8). IEEE.
5. Whalley, J., Prasad, C., & Kumar, P. A. (2007, January). Decoding doodles: novice programmers and their annotations. In ACM International Conference Proceeding Series (Vol. 239, pp. 171-178).
6. Hertz, M., & Jump, M. (2013, March). Trace-based teaching in early programming courses. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 561-566).
7. Cunningham, K., Blanchard, S., Ericson, B., & Guzdial, M. (2017, August). Using tracing and sketching to solve programming problems: replicating and extending an analysis of what students draw. In Proceedings of the 2017 ACM Conference on international computing education research (pp. 164-172).
8. Xie, B., Nelson, G. L., & Ko, A. J. (2018, February). An explicit strategy to scaffold novice program tracing. In Proceedings of the 49th ACM technical symposium on computer science education (pp. 344-349).
9. Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., ... & Thomas, L. (2004). A multi-national study of reading and tracing skills in novice programmers. ACM SIGCSE Bulletin, 36(4), 119-150.
10. Kirsh, D. (2010). Thinking with external representations. AI & society, 25(4), 441-454.
11. Fitzgerald, S., Simon, B., & Thomas, L. (2005, October). Strategies that students use to trace code: an analysis based in grounded theory. In Proceedings of the first international workshop on Computing education research (pp. 69-80).
12. Lister, R., Simon, B., Thompson, E., Whalley, J. L., & Prasad, C. (2006). Not seeing the forest for the trees: novice programmers and the SOLO taxonomy. ACM SIGCSE Bulletin, 38(3), 118-122.
13. Chase, W. G., & Simon, H. A. (1973). Perception in chess. Cognitive psychology, 4(1), 55-81.
14. Katai, Z. (2014, June). Selective hiding for improved algorithmic visualization. In Proceedings of the 2014 conference on Innovation & technology in computer science education (pp. 33-38).
15. Sorva, J., Karavirta, V., & Malmi, L. (2013). A review of generic program visualization systems for introductory programming education. ACM Transactions on Computing Education (TOCE), 13(4), 1-64.
16. Du Boulay, B. (1986). Some difficulties of learning to program. Journal of Educational Computing Research, 2(1), 57-73.
17. Fitzgerald, S., Simon, B., & Thomas, L. (2005, October). Strategies that students use to trace code: an analysis based in grounded theory. In Proceedings of the first international workshop on Computing education research (pp. 69-80).
18. Giere, R. N. (2004). How models are used to represent reality. Philosophy of science, 71(5), 742-752.
19. McCartney, R., Moström, J. E., Sanders, K., & Seppälä, O. (2004). Questions, Annotations, and Institutions: observations from a study of novice programmers. In the Fourth Finnish/Baltic Sea Conference on Computer Science Education, October 1–3, 2004 in Koli, Finland (pp. 11-19). Helsinki University of Technology, Department of Computer Science and Engineering, Laboratory of Information Processing Science, FINLAND.

20. Perkins, D. N., Hancock, C., Hobbs, R., Martin, F., & Simmons, R. (1986). Conditions of learning in novice programmers. Journal of Educational Computing Research, 2(1), 37-55.