Algoritmikus gondolkodás fejlesztése keresési és rendezési algoritmusokon keresztül

Mahler-Lakó Viktória[1], Mahler Attila[2]

[1] viktoria.lako@gmail.com
ELTE IK
[2] mahler.attila@berzsenyi.hu
ELTE IK

Absztrakt. Az algoritmikus gondolkodás fejlesztése a 21. századi informatika oktatás kiemelt feladata. Cikkünkben egy szakkör tapasztalatait mutatjuk be, melynek témája az algoritmus fogalmának megértése különböző keresési és rendezési algoritmusok megismerésén keresztül. A szakkör eredményességét tesztek segítségével vizsgáltuk, melynek konklúzióit szintén bemutatjuk.

Kulcsszavak: algoritmikus gondolkodás, rendezési algoritmusok, lineáris és bináris keresés, közoktatás

1. Bevezető

Egy korábbi tanulmányunkban már bemutattuk egy 6 órás szakkör tervét [1], amelyet kipróbáltunk egy középiskolai csoportban, ennek tapasztalatait mutatjuk be cikkünkben. A szakkör célja a tanulók algoritmikus gondolkodásának fejlesztése [2, 3] különböző játékok [4] és szemléltető programok [5, 6] segítségével.

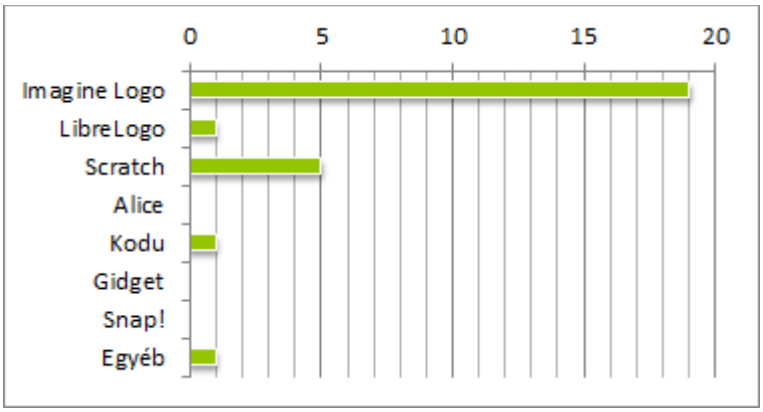
A szakkörön két keresési algoritmussal és három rendezési algoritmussal ismerkedtünk meg részletesebben [7]. Ezeken a problémákon keresztül ismertettük meg a résztvevőket az algoritmus fogalmával, az algoritmus készítés lépéseivel, az algoritmus alapvető szerkezeti elemeivel (szekvencia, elágazás, ciklusok), valamint az algoritmus pszeudo kódos leírásával.

2. Szakkör résztvevőinek bemutatása

A szakkörön a budapesti Berzsenyi Dániel Gimnázium 9. C osztályos tanulói közül jelentkező 19 tanuló vett részt. A diákok 6 évfolyamos speciális matematika tagozatra járnak, hetente 7 matematika és 2 informatika órájuk van. A csoportról tehát elmondható, hogy erős matematikai érdeklődésűek, szeretnek gondolkodni, problémákat megoldani és érdekli őket a programozás is.

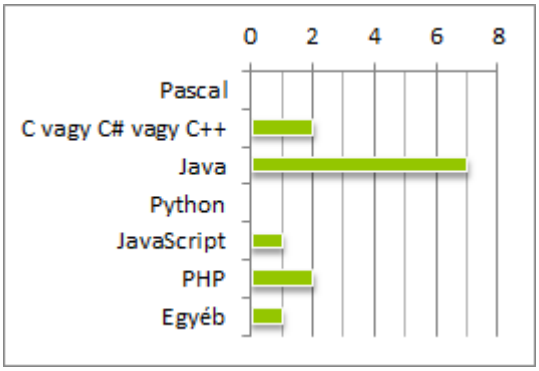
A nemek aránya a vizsgált csoporton belül megegyezik az osztálybeli aránnyal 2:1 a fiúk javára.

A tanulókat megkérdeztük az előzetes programozási ismereteikről. Az osztály az előző tanévben Imagine Logo nyelven tanult programozni, ezért ezt minden résztvevő ismeri. Az Imagine-en kívül még a Scratch fejlesztő környezetet ismerték még számottevően (1. ábra).



1. ábra: Mely játékos programozási környezetben készítettél már saját programot?

A professzionális programozási nyelvek közül a Java volt a legmeghatározóbb, ennek magyarázata, hogy a csoportból többen járnak egy végzett diák által tartott algoritmus szakkörre, amelyen a Java programozási nyelvben dolgoznak (2. ábra). Közülük a Java-n kívül még a C-szerű illetve a weboldalkészítéshez kapcsolódó nyelveket használták már a tanulók, de a többiek egyáltalán nem használtak még semmilyen professzionális programozási környezetet.



2. ábra: Mely professzionális programozási nyelven készítettél már saját programot?

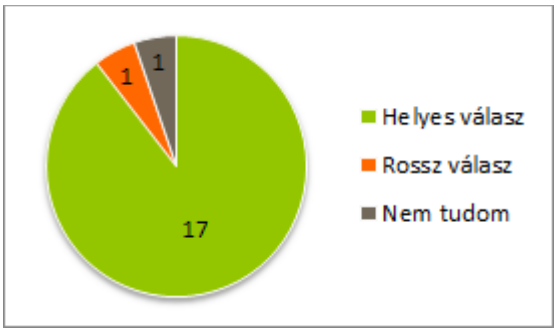
3. Bemeneti teszt

A résztvevők előzetes ismereteit egy 5 kérdésből álló feladatsorral mértük fel, amely egy könnyű, három közepes nehézségű és egy nehéz feladatból állt. A feladatokban a pszeudo kódhoz hasonló (a ciklus kifejezés helyett az ismétlést alkalmaztuk, mivel nem volt előfeltétel a ciklus ismerete a Logo nyelvből pedig ismerik az `ismétlés` utasítást) algoritmus leíró nyelven írt algoritmusokat kellett a diákoknak értelmezni egy-egy konkrét bemenetre.

3.1 Teszt kiértékelése

1. Mi lesz az eredmény értéke az algoritmus végén, ha a=10 és b=5?

```
ha a<b akkor
          eredmény=a+b
különben
          eredmény=a*b
elágazás vége
```

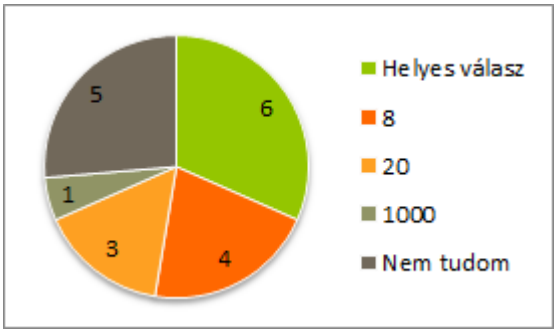


3. ábra: 1. feladatra adott válaszok eloszlása

Az első feladat egy elágazás értelmezése volt, amelyet a diákok többsége (17 fő) helyesen értelmezett, egy tanuló hibásan válaszolt, a számok összegét adta meg válaszként a szorzatuk helyett, egy tanuló pedig nem tudott válaszolni a kérdésre (3. ábra).

2. Mi lesz az eredmény értéke az algoritmus végén, ha n=4?

```
eredmény=2
ismétlés i=1-től n-ig
          eredmény=eredmény*i
ismétlés vége
```



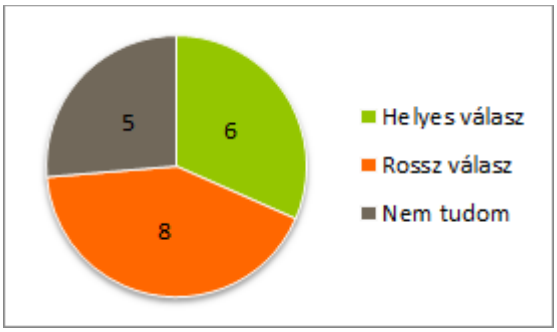
4. ábra: 2. feladatra adott válaszok eloszlása

A második feladatban egy számlálós ciklust kellett értelmezniük a tanulóknak, erre a kérdésre már csak 6-an tudtak helyesen válaszolni, és 5-en úgy ítélték, hogy nem tudják megoldani a feladatot (4. ábra).

A leggyakoribb helytelen válasz a 2. kérdés esetében a 8 volt, amely azt sugallja, hogy a tanuló úgy vette, hogy az `eredmény=2` utasítás a ciklusmagban található. A második leggyakoribb válasz a 20, amely úgy állhat elő, ha 1-től n-ig összegzi a 2*i értékeket.

3. Mi lesz az eredmény értéke az algoritmus végén, ha n=35?

```
i=2
ismétlés amíg n nem osztható i-vel
          i=i+1
ismétlés vége
eredmény=i
```



5. ábra: 3. feladatra adott válaszok eloszlása

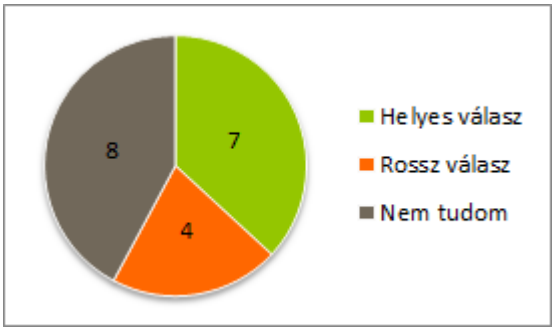
A 3. feladatban a feltételes ciklus értelmezését kértük számon, erre a kérdésre szintén 6-an adtak helyes választ (5. ábra). Érdekesség, hogy közülük csak 2-en válaszoltak a 2. kérdésre is helyesen. A helytelen válaszok közül a 3 fordult elő többször is, összesen háromszor, közülük ketten a második kérdésre a 8-at adták meg helyes válaszként. Ez a válasz arra enged következtetni, hogy a ciklusmagot nem ismételték meg, tehát gyakorlatilag egy elágazásként fogták fel ezt a programszerkezetet.

4. Mi lesz az alábbi példában az eredmény táblázat 5 eleme az algoritmus végén? Sorold fel őket szóközökkel elválasztva!

sorozat[1]	sorozat[2]	sorozat[3]	sorozat[4]	sorozat[5]
5	0	-3	-8	2

eredmény[1]	eredmény[2]	eredmény[3]	eredmény[4]	eredmény[5]

```
ismétlés i=1-től 5-ig
          ha sorozat[i]≥ 0 akkor
            eredmény[i]=sorozat[i]
          különben
            eredmény[i]= -(sorozat[i])
          elágazás vége
ismétlés vége
```



6. ábra: 4. feladatra adott válaszok eloszlása

A 4. feladatban rejtetten megjelent a tömb fogalma, és itt már elágazást és ciklust is kellett egy időben értelmezniük a diákoknak, mégis ezt a feladatot 7 diáknak is sikerült helyesen megoldania (6. ábra). A helyesen válaszolók közül 6-an az előző két kérdésből legalább egyre helye-

sen válaszoltak, a hetedik pedig az első három kérdés mindegyikére helytelenül válaszolt. Bár erre a kérdésre többen válaszoltak helyesen, mint az előző két kérdésre, nőtt azok száma, akik nem tudták megoldani a feladatot.

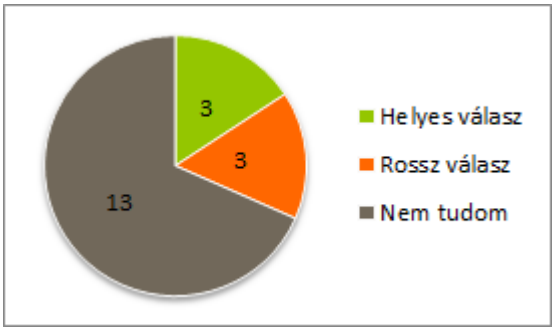
5. Mik lesznek az alábbi példában az eredmény táblázat elemei az algoritmus végén? Sorold fel őket szóközökkel elválasztva!

A-sorozat[1]	A-sorozat[2]	A-sorozat[3]	A-sorozat[4]	A-sorozat[5]
9	1	3	7	5

B-sorozat[1]	B-sorozat[2]	B-sorozat[3]	B-sorozat[4]
5	4	3	8

eredmény[1]	eredmény[2]	eredmény[3]	...	eredmény[db]

```
db=0
ismétlés i=1-től 5-ig
           j=1
           ismétlés amíg (j≤4 és A-sorozat[i]≠B-sorozat[j])
             j=j+1
           ismétlés vége
           ha j≤4 akkor
             db=db+1
             eredmény[db]=A-sorozat[i]
           elágazás vége
ismétlés vége
```



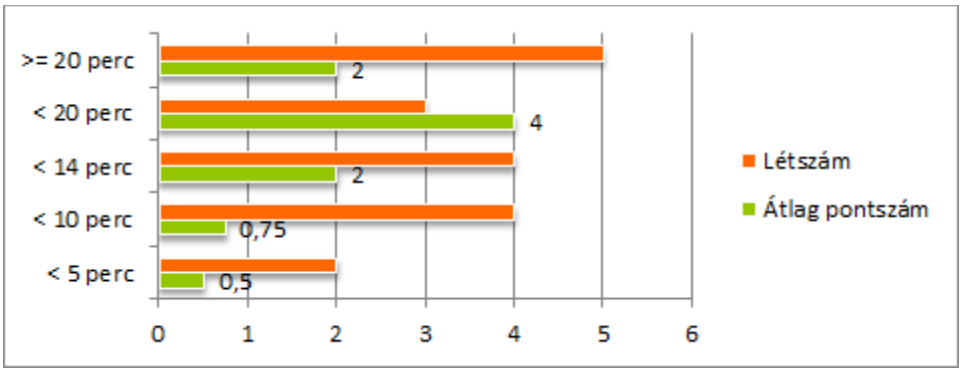
7. ábra: 5. feladatra adott válaszok eloszlása

Az 5. feladatnak a metszet programozási tétel algoritmusát választottuk, amely már mindkét fajta ciklust tartalmazza, ráadásul egymásba ágyazva, megjelenik benne a tömb fogalma és az elágazás is. Ez a kérdés azért került bele, hogy a már tapasztaltabb diákokat is kihívás elé állítsuk, az ő fejlődésüket is mérni tudjuk.

Erre a kérdésre már csak hárman tudtak helyesen válaszolni. Rajtuk kívül összesen hárman próbálkoztak a feladat megoldásával, a többiek elrettentek az algoritmus bonyolultságától (7. ábra).

3.2 Tesztből levont tanulságok

A bemeneti tesztet a tanulók otthon töltötték ki a szakkört megelőzően, megkértük őket, hogy a megoldás során semmilyen segédeszközt ne vegyenek igénybe és mérjék a tesztre fordított időt.



8. ábra: A tesztre fordított idő és a kapott pontszám közötti kapcsolat

A tesztre fordított időt és a tesztre kapott átlagpontszámokat vizsgálva (8. ábra) azt látjuk, hogy, aki kevesebb, mint 5 percet foglalkozott a feladattal, nem is gondolkodott a feladatokon, az algoritmust látva egyből rávágta, hogy „Nem tudom”. Az is kiderül, hogy ahhoz, hogy érdemben foglalkozzanak a feladatokkal legalább 14 percet kellett volna foglalkozniuk a teszttel. Érdekes, hogy 4-en több mint 20 percig gondolkodtak a megoldásokon, mégis elég rosszul teljesítettek a bemeneti teszten.

A diákok átlagosan 14 percet töltöttek a teszt kitöltésével, a legkevesebb idő 1 perc (csupa „Nem tudom” válasz), a legtöbb 31 perc volt. A tesztet 3 tanuló töltötte ki hibátlanul, ők 14, 16 illetve 30 percet fordítottak a tesztre. A három hibátlanul kitöltő mindegyike jár a másik algoritmus szakkörre, tehát voltak előzetes ismereteik.

A kérdésekre adott helyes válaszok számából arra következtetünk, hogy sikerült a feladatok nehézségi szintjét eltalálnunk. Valamint az is kiderült, hogy a ciklus értelmezése könnyebben ment a tömbelemekre hivatkozva, mint csupán egy ciklusváltozó értékének segítségével. De az is lehetséges, hogy a változó önmagára való hivatkozása okozott gondot a 2. és 3. feladat esetében.

4. Szakkör bemutatása

1. óra

Az első alkalommal kétféle torpedó játékot játszattunk a résztvevőkkel [4]. Mindkét játékot párban játsszák a diákok, mindegyiküknek van egy hajóflottájuk, a hajókat betűkkel lehet azonosítani, és mindegyik hajón van egy számozott láda. Mindkét diák elhelyezi a saját titkos fegyverét az egyik ládában, ennek a ládának a számát elárulja az ellenfélnek. A játékosok feladata megtalálni azt a hajót, amelyen a titkos fegyvert tartalmazó láda van, oly módon, hogy felváltva megkérdezhetik egy adott hajón lévő láda sorszámát. Az első esetben a ládák sorrendje véletlenszerű, míg a másodikban a ládák a sorszámuk szerint növekvő sorrendben helyezkednek el a hajókon (ezt a tanulóknak maguknak kell észrevenniük). Mindkét játék során számolniuk kellett, hogy hány lépés után sikerült megtalálnia a nyertesnek a titkos fegyvert.

Az egyes játékok után megbeszéltük a tapasztalatokat, mint a lépések számát és, hogy milyen stratégiát volt érdemes követni a játékosoknak. Az első játékkal a lineáris, a másodikkal pedig bináris keresés stratégiájához sikerült együtt eljutnunk, valamint mindkét algoritmus esetében kiszámítottuk a lehetséges legkevesebb és legtöbb lépés darabszámát a hajók számának függvényében, a legtöbb lépés esetén azt a lehetőséget is megengedtük, hogy a keresett láda nincs a megadott hajókon.

2. óra

A következő órán a diákok LibreLogo program segítségével kipróbálhatták az előző órán megismert algoritmus ötleteket számítógéppel megvalósítva. A programban a játékhoz képest néhány dolog módosult: a keresett érték nem feltétlenül volt benne a megadott sorozatban (ezt már az előző alkalommal is érintettük). A sorozat elemeit véletlenszerűen generáltuk, ami azt eredményezte, hogy azonos értékek is lehetnek a sorozatban, a gyerekek azt is észrevették, hogy ekkor is megáll mindkét algoritmus, ha megtalált egy keresett elemet. A programban kipróbálhatták, hogy a sorozat elemszámának változtatása, illetve a keresett elem helyzete a sorozatban, milyen hatással van az algoritmus lépésszámára.

A kísérletezés után az algoritmus fogalmát beszéltük meg közösen, valamint az algoritmuskészítés főbb lépéseit (algoritmus ötlet, algoritmus, leírása (pszeudo kód), program kód, gépi kód). Ezután megismerkedtünk a szakkörön használni kívánt algoritmus leíró nyelv (pszeudo kód) elemeivel és a főbb algoritmus szerkezetekkel: szekvencia, elágazás, ciklusok (feltételes, számlálós). Ezen az órán a tervekkel ellentétben csak a lineáris keresés algoritmusát tudtuk részletesen megbeszélni és leírni.

3. óra

A harmadik óra elején felelevenítettük a bináris keresés algoritmus ötletét, egy konkrét példán közösen végigjátszottuk az algoritmus lépéseit. Ezután először vázlatosan, majd pszeudo kóddal is leírtuk az algoritmust.

Az óra további részében ismét egy játék következett: a csoport tagjait háromfős csapatokba szerveztük és minden csapat kapott 6 db számozott kártyát, egy játéktáblát és egy feladatleírást. A játék során a tanulóknak sorba kellett rendeznie a számkártyákat, a játéktáblán, úgy hogy csak a következő lépéseket alkalmazhatták:

- A mérleg segítségével két kártya értékének összehasonlítása.
- A két összehasonlított kártya megcserélése. (Az eredeti sorba a kártyák csak értékükkel lefelé fordítva kerülhetnek!)
- Egy kártya félretevése a segéd négyzetbe (lefordítva).

A játék során vezetniük kellett a végrehajtott lépéseket. Az algoritmus ötletek közösen való megbeszélésére nem maradt idő, de összegyűjtöttük az általuk leírt rendezési lépéseket, amelyeket a következő alkalom előtt elemeztünk. A 6 csapat közül négynek sikerült a kártyák sorba rendezése, két csapat az egyszerű cserés, két csapat a buborékos rendezés szerint rendezte a kártyákat, ezért a későbbiekben ezt a két algoritmus beszéltük meg részletesen.

4. óra

A negyedik alkalommal a tanulók által kitalált rendezési algoritmusokat beszéltük meg közösen, elsőként a két csoport által kitalált egyszerű cserés rendezését. Ismét kaptak egy LibreLogo programot, amelynek segítségével tesztelhették az algoritmust különböző bemeneti adatokra (véletlenszerű, rendezett, visszafelé rendezett), különböző elemszámokra, a program jelezte az összehasonlításokat és cseréket, valamint ezek darabszámát.

Az egyszerű cserés rendezés algoritmusát pszeudo kóddal is leírtuk, kiszámítottuk az összehasonlítások és a lehetséges cserék számát a sorozat elemszámának függvényében. A buborékos rendezésről csak az alapötletet sikerült megvitatnunk.

5. óra

Folytatván az előző órai gondolatmenetet, a diákok másik rendezési ötlete alapján megnéztük a buborékos rendezés algoritmusát, a LibreLogo segédanyag segítségével az összehasonlítások és cserék számát, majd pszeudo kóddal le is írtuk az algoritmust. Ezután a javítási ötletet is megbeszéltük és leírtuk a javított buborékos rendezés algoritmusát is, valamint megnéztük az ehhez tartozó segédanyagot is.

Így előkerült a hatékonyság kérdése is, mert míg a korábbi algoritmusok mindenképpen $n \cdot (n+1)/2$ összehasonlítást végeztek, addig a javított buborékos nem feltétlenül. Például rendezett elemek esetén n-1 összehasonlítás után végzett. Ezt az észrevételt is megfogalmaztuk.

6. óra

Az utolsó foglalkozáson a diákok ismét háromfős csoportokban dolgoztak. Kaptak egy újabb rendezési algoritmust (minimum kiválasztásos), azt kellett eljátszaniuk, értelmezniük. Végül ezt közösen megbeszéltük és nevén neveztük ezt az algoritmust is, majd a LibreLogo segédanyaggal ezt is szemléltettük, elemeztük. Újra felmerült a hatékonyság kérdése, hiszen ennél az algoritmusnál is biztosan $n \cdot (n+1)/2$ a lépésszám, de a cserék száma egy visszafele rendezett tömbben csupán $n/2$. Így az is felismerést nyer, hogy több szempontból lehet a hatékonyságot vizsgálni, illetve adott helyzettől függ, hogy melyik rendezési algoritmus a hatékony.

A szakkör zárásaként pedig megnéztünk egy rendezési algoritmusokat animációk segítségével összehasonlító honlapot [8], majd videókat, amelyen magyar néptáncosok eltáncolnak [9] különböző rendezési algoritmusokat. Ezeket nagyon látványosnak találták és nagyon szerették a gyerekek, jól összegezték és lezárták a szakkört.

5. Kimeneti teszt

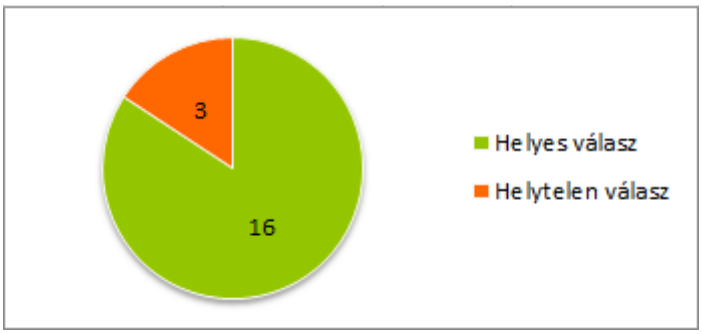
A szakkör után is felmértük a diákok tudását egy teszt segítségével. Ebben már teljes egészében a szokásos pszeudo kódos jelölést használtuk, beleértve a ciklus szóhasználatot és az értékadó illetve összehasonlító egyenlőségek megkülönböztetését is.

Terveink szerint 2 könnyű, 2 közepes és 2 nehéz feladatot tűztünk ki, amelyeket az alábbiakban bemutatunk.

5.1. Teszt kiértékelés

1. Meg szeretnénk cserélni az a és b változók értékeit, de egy parancs hiányzik. Mi az?

```
segéd:=a
a:=b
...
```



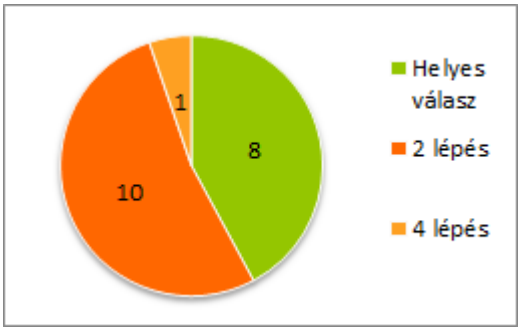
9. ábra: 1. feladatra adott válaszok eloszlása

A rendezési algoritmusokban megkerülhetetlen a csere, így természetesen ezt is algoritmizáltuk, ezt könnyű kérdésnek szántuk és ezt az eredmények is igazolták (9. ábra).

2. Hány lépésben találja meg az alábbi bináris keresési algoritmus a keresett=15-öt, ha n=10 és sorozat=[10, 12, 15, 18, 20, 23, 27, 31, 35, 40]?

```
eleje:=1
vége:=n
közepe=[(eleje+vége)/2]
ciklus amíg (keresett ≠ sorozat[közepe]) és (eleje ≤ vége)
          ha keresett > sorozat[közepe] akkor
            eleje:=közepe+1
          különben
            vége:=közepe-1
          elágazás vége
          közepe=[(eleje+vége)/2]
ciklus vége
ha keresett=sorozat[közepe] akkor
          ki: „Van”, közepe
különben
          ki: „Nincs”
elágazás vége
```

A második feladatban, egy megbeszélt algoritmus (bináris keresés) lépésszámára kérdeztünk rá konkrét példában. Itt az algoritmus csak emlékeztetőként szerepelt, az volt a feltevésünk, hogy tudják a stratégiát és az alapján oldják meg. Bár könnyűnek gondoltuk (mivel alaposan átbeszéltük a szakkörön) a többség elrontotta és azt írta, hogy 2 lépésből találja meg (10. ábra).



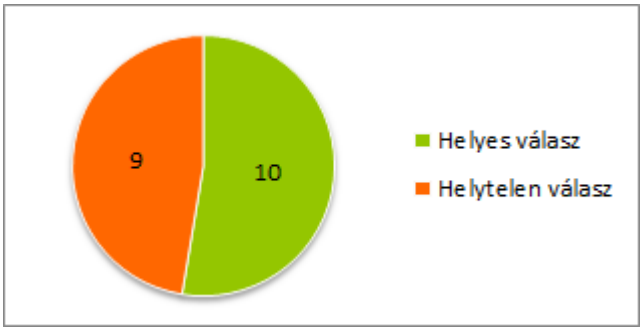
10. ábra: 2. feladatra adott válaszok eloszlása

Két ötletünk van, hogy miért gondolhatták ezt. Az egyik, hogy első lépésben vették az 5. elemet, majd a következő lépésben nem csak az előtte lévőket nézték, hanem még az 5.-et is, így a 3.-at vették középsőnek, ami a keresett. A másik lehetőség, hogy első lépésben a 6. elemet vették középsőnek (egészrész helyett, kerekítettek) és utána az 1-5. elemek közül már valóban a 3. a középső. Meglepve tapasztaltuk, hogy a bemeneti tesztet hibátlanul kitöltők mindegyike a 2-t adta meg válaszként.

3. Mi lesz az x értéke az algoritmus végén, ha a=9 és b=12?

```
x:=a
y:=b
ciklus amíg x ≠ y
          ha x < y akkor
            x:=x+a
          különben
            y:=y+b
          elágazás vége
ciklus vége
```

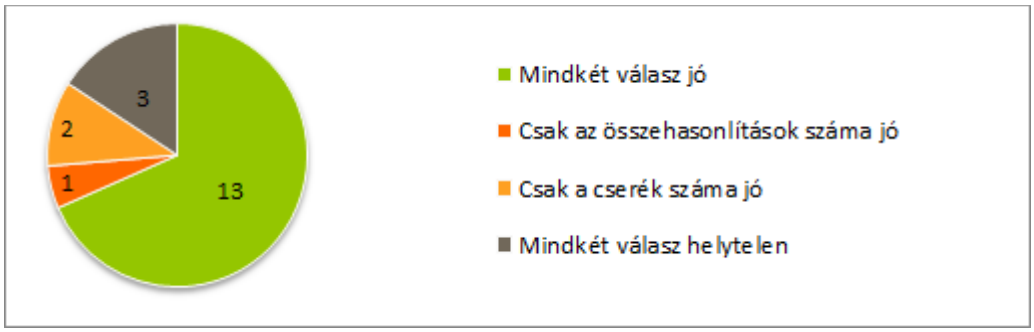
Ezt a feladatot közepesnek minősítettük, azért, mert összességében nem egy bonyolult algoritmus, de nincs közvetlen kapcsolata a szakkörön megbeszélt algoritmusokkal. A diákok éppen több mint fele (10-en) tudták helyesen megoldani (11. ábra).



11. ábra: 3. feladatra adott válaszok eloszlása

4. Hány összehasonlítást és hány cserét végez az alábbi rendezési algoritmus, ha sorozat=[10, 15, 8, 11, 12] (n=5)?

```
ciklus i:=1-től n-1-ig
          ciklus k:=i+1-től n-ig
            ha sorozat[i] > sorozat[k]
              s:=sorozat[i]
              sorozat[i]:=sorozat[k]
              sorozat[k]:=s
            elágazás vége
          ciklus vége
ciklus vége
```

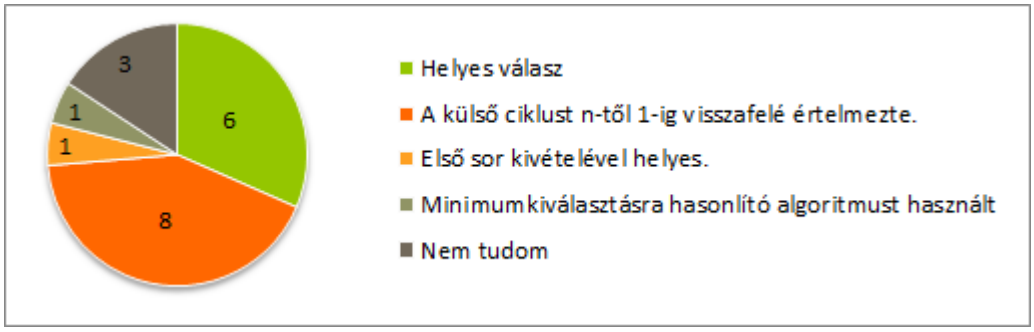


12. ábra: 4. feladatra adott válaszok eloszlása

Ezt a feladatot szintén közepes nehézségűnek szántuk, mert a szakkörön megbeszélt (egyszerű cserés) rendezési algoritmus megértésére kérdez rá. Ellentétben a 2-es feladattal itt nem adtuk meg, hogy melyik megbeszélt algoritmus szerepel, arra csak a pszeudo kódból lehetett ráismerni. A többség jól oldotta meg a feladatot: 13-an mindkét kérdésre jól válaszoltak, további 1, illetve 2 diák csak az egyik kérdésre adott helyes választ, és csak 3 olyan válasz született, amelynek mindkét fele hibás (12. ábra). A válaszból nem derül ki, hogy valóban a megadott algoritmussal dolgozott-e a diák, mert a másik két megismert rendezési algoritmus is azonos lépésben és csere számmal oldja meg a feladatot. Célszerű lett volna ebben a feladatban is a sorozat cserék utáni állapotait is megkérdezni.

5. A következő algoritmussal sorba szeretnénk rendezni a sorozat=[13, 17, 30, 10, 22] (n=5)-ot. Írd le a sorozat állapotát minden csere után!

```
ciklus i:=2-től n-ig
          j:=i-1
          ciklus amíg (j ≥ 1) és (sorozat[j] > sorozat[j+1])
            s:=sorozat[j]
            sorozat[j]:=sorozat[j+1]
            sorozat[j+1]:=s
            j:=j-1
          ciklus vége
ciklus vége
```



13. ábra: 5. feladatra adott válaszok eloszlása

Ebben a feladatban egy a szakkörön nem tárgyalt rendezési algoritmust (beillesztéses rendezés) kellet alkalmazniuk egy konkrét sorozatra, a helyes lépéseket 6 tanulónak sikerült felismernie, érdekesnek találtuk, hogy a tanulók több mint harmada a belső ciklust jól értelmezte és

szomszédos elemeket cserélt a rendezés során, de a külső ciklusban visszafelé gondolkodott (13. ábra).

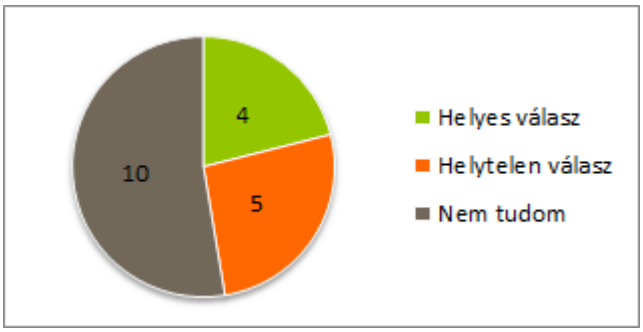
Az utolsó órán a minimum kiválasztásos rendezéssel kellett átgondolniuk egy hasonló feladatot, ezért lehet, hogy egy tanuló egy ehhez hasonló módszerrel próbálta rendezni a kapott sorozatot.

6. Mik lesznek az alábbi példában az eredmény sorozat elemei az algoritmus végén, ha A_sorozat=[11, 19, 12, 17, 18, 10] (n=6) and B_sorozat=[14, 18, 13, 11] (m=4)? Sorold fel őket szóközökkel elválasztva!

```
db:=0
ciklus i:=1-től n-ig
          j:=1
          ciklus amíg (j ≤ m) és (A_sorozat[i] ≠ B_sorozat[j])
            j:=j+1
          ciklus vége
          ha j ≤ m akkor
db:=db+1
            eredmény[db]:=A_sorozat[i]
          elágazás vége
ciklus vége
```

A 6. feladatban a már bemeneti tesztben is kérdezett metszet programozási tételt kérdeztük egy másik bemenetre. Ezt az algoritmust egyáltalán nem érintettük a foglalkozásokon, azért került bele a kimeneti tesztbe, hogy lássuk egy más jellegű, összetettebb algoritmus megértését mennyire segítette a szakkör.

Az eredmények kis mértékben, de javultak, még egy tanulónak sikerült helyesen megoldania a feladatot, és kettővel többen próbálkoztak a megoldással, mint a bemeneti teszt esetében (14. ábra).

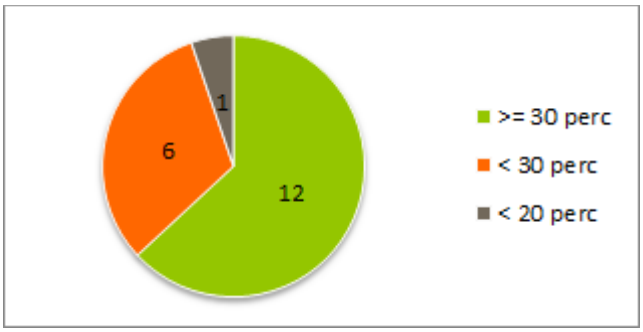


14. ábra: 6. feladatra adott válaszok eloszlása

5.2. Tesztből levont tanulságok

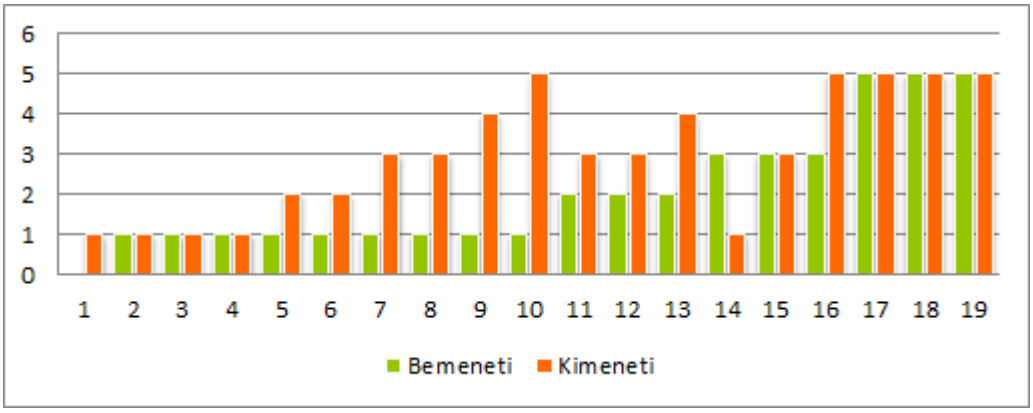
A bemeneti teszt esetében azt tapasztaltuk, hogy sokan nem hagytak maguknak elegendő időt a feladatok alapos értelmezésére, ezért a kimeneti tesztnél arra kértük őket, hogy alaposan gondolják át a válaszokat, szánjanak rá időt, és legalább 30 percet fordítsanak a teszt kitöltésére.

A kimeneti teszt több és nehezebb feladatot tartalmazott a bemeneti tesztnél, ezért a tanulók valóban több időt foglakoztak a feladatok megoldásával (15. ábra). Érdekesség, hogy mindhárom kategóriába tartozó diákok átlagpontszáma megegyezik a csoport átlagpontszámával, ebből azt feltételezzük, hogy mindenki kellő időt gondolkodott a feladatokon.



15. ábra: Az időráfordítás alakulása a kimeneti tesztnél

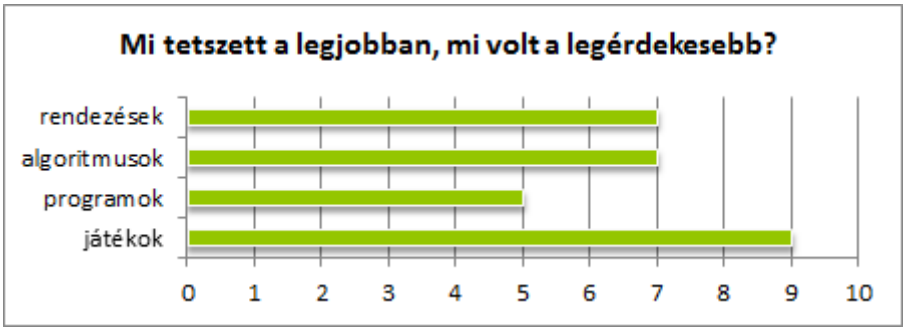
Érdekelt minket a tanulók egyéni fejlődése is, a résztvevők pontszámainak változását a 16. ábra mutatja. A csoportban egy diák volt, aki rosszabb eredményt ért a szakkör után, mint előtte, 9-en javítottak a korábbi eredményükön. Mivel a kimeneti teszt közepes és nehéz feladatai jóval nehezebbek voltak a bevezető feladatoknál, ezért a bemeneti teszten legalább 3 pontot szerzők esetében már akkor is az eredmény javulásáról beszélhetünk, ha a kimeneti pontszám megegyezik az előzetes pontszámmal.



16. ábra: A tanulók teljesítményének változásai

6. A tanulók személyes benyomásai

A szakkör végén a tanulók személyes véleményére is rákérdeztünk, ezeket is közöljük.



17. ábra

Elsőként azt kérdeztük meg tőlük, hogy mi tetszett nekik a legjobban: a legtöbben a játékokat emelték ki, valamint sokaknak tetszettek az algoritmusok azon belül főként a rendezések. Öten a szemléltető programokat emelték ki (17. ábra).

A résztvevők kétharmada mindennel elégedett volt, hármójuk szerint kevesebbet kellett volna játszani és több algoritmust kellett volna megbeszélni, ketten az órák időpontjával voltak elégedetlenek (nehéz volt közös időpontot találni, legtöbbször a 8. órában tudtuk a foglalkozásokat tartani és előfordult, hogy egy nap két alkalom is volt.) Egy diáknak a pszeudo kódos leírás nem tetszett. Egy diák pedig a szakkör rövidségére panaszkodott (18. ábra).

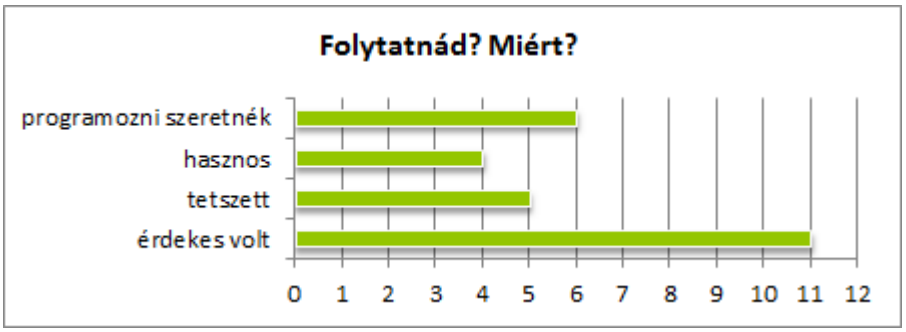
18. ábra

Arra is kíváncsiak voltunk, hogy a tanulók szerint a szakkör mely elemei segítették jobban az algoritmusok megértését, valamint, hogy hogyan értékelik saját fejlődésüket (19. ábra).

19. ábra

A diákok szerint mind a játékok, mind a szemléltető programok meghatározó szerepet játszottak az algoritmusok megértésében, szerintük a játékoknak volt erősebb hatása. Állításuk szerint ugyanannyira értették meg magukat az algoritmusokat, mint az azok leírására szolgáló nyelvet. Az algoritmusok értelmezésére kapott osztályzatukat ennél kicsit rosszabbra értékelték.

Minden résztvevő folytatná a szakkört, indokaikat a 20. ábra szemlélteti. A legtöbben azért folytatnák, mert érdekesnek találták és élvezték a foglalkozásokat. A diákok egyharmada szeretne a későbbiekben programozni.



20. ábra

7. Összegzés

Összességében a diákok és mi is hasznosnak és eredményesnek ítéltük meg a szakkört és az azon elvégzett munkát. Komoly előkészületek előzték meg a szakköröket, a szakirodalom tanulmányozása, az óratervek alapos átgondolása, valamint nagy könnyebbséget jelentett, hogy az órák többségén két tanár volt jelen. Ez lehetőséget biztosított arra, hogy az apróbb fennakadásokon az óramenet megszakítása nélkül tovább lehetett lépni, valamint a csoportok munkáját is jobban nyomon lehetett követni, gyorsabban meg lehetett nézni, hallgatni az ötleteket, eredményeket.

Ez a kérdőíveztetéssel kiegészített hat alkalom számos tanulság levonására is alkalmas volt. Az egyik, hogy jó képességű, érdeklődő, a szakkörre önként jelentkező gyerekek esetében is a tervezettnél időigényesebb volt az algoritmusok átbeszélése, megértése és leírása. Így azt gondoljuk, hogy egy átlagos iskolában, átlagos képességű tanulók esetében legalább kétszer ekkora óraszám (12 óra) lenne szükséges. Továbbá célszerű lenne a bemeneti és a kimeneti tesztet is helyben, a szakkörön megíratni, ami újabb két órát ölel fel. Sőt akár bővíteni is lehetne, még több olyan feladattal, ahol ismeretlen algoritmust kell értelmezni. Ez az igény megjelent a diákok válaszai között is, és mi is úgy látjuk, hogy mélyebbé tenné a megértését az algoritmusoknak. Főleg, mert a szakkör eredeti céljai közt szerepelt, hogy a tanulók képessé váljanak ismeretlen algoritmusok értelmezésére.

A szakkörnek kevés az eszközigénye, ezért szinte minden iskolában kivitelezhető, nyomtatott papíron és néhány számítógépen kívül nincs szükség másra és a LibreLogo is ingyenesen és több platformra is elérhető program, tehát szoftverekre sem kell költeni.

A jelenlegi kerettanterv kötelezően 9 órát szán az algoritmikus gondolkodás fejlesztésére, ami a szakköri tapasztalatok alapján (is) nagyon kevés, de a foglalkozások játékos és szemléltető elemei, akár az algoritmusok precíz leírása nélkül is bekerülhetnek az informatika tananyagba. Ahol az idő engedi a teljes 12 órás anyag is beemelhető, illetve akár további játékokkal, feladatokkal bővíthető.

Kérdés, hogy miért éppen a rendezési algoritmusokat emeljük ki egy ilyen tág témakörből? Sokak szerint nem is szükséges rendezési algoritmusokat tanítani, hiszen minden programozási nyelv tartalmaz valamilyen rendező függvényt. Pedagógiai szempontból azonban mégis nagy szerepük van a rendezési algoritmusoknak: mert a hétköznapi életben is találkozunk a sorba rendezés problémájával; jó példával szolgálnak arra, miért keressünk egy problémára egy új megoldást az után, hogy már találtunk egyet; valamint jól lehet vele szemléltetni velük a hatékonyság sokféle aspektusát.

Irodalom

1. MAHLER-LAKÓ V., MAHLER A.: Improving computational thinking skills via sorting algorithms, Budapest (2016), XXIX. DIDMATTECH 2016
2. WING, J. M.: Computational Thinking, In Commun, ACM, 49:33-35., 2006
3. WING, J. M.: Computational thinking, 10 years later, On Microsoft Research Blog, 2016. https://blogs.msdn.microsoft.com/msr_er/2016/03/23/computational-thinking-10-years-later/ (utoljára megtekintve: 2016.11.05.)
4. BELL, T., WITTEN, I.H. AND FELLOWS, M.: CS Unplugged: An enrichment and extension programme for primary-aged students (2015) http://csunplugged.org/wp-content/uploads/2015/03/CSUnplugged_OS_2015_v3.1.pdf (utoljára megtekintve: 2016.11.05.)
5. LAKÓ,V.: LibreLogo oktatási segédanyag, 2013 http://mek.oszk.hu/12700/12781/12781.pdf (utoljára megtekintve: 2016.11.05.)
6. MAHLER-LAKÓ, V.: LibreLogo és a rendezési algoritmusok, Budapest (2016), Linux az oktatásban konferencia 2016 http://lok.hu (utoljára megtekintve: 2016.11.05.)
7. GREGORICS, T., HEIZLERNÉ BAKONYI, V., HORVÁTH, GY., MENYHÁRT, L., PAP, G. S., PAPP-VARGA, ZS., SZLÁVI, P. AND ZSAKÓ, L.: Programozási alapismeretek eTananyag: Rendezések és hatékonyságmérések, ELTE http://progalap.elte.hu/downloads/seged/eTananyag/lecke30_lap1.html (utoljára megtekintve: 2016.11.05.)
8. Rendezési algoritmusok összehasonlítása: http://www.sorting-algorithms.com/ (utoljára megtekintve: 2016.11.05.)
9. Bubble-sort with Hungarian ("Csángó") folk dance: https://www.youtube.com/watch?v=lyZQPjUT5B4 (utoljára megtekintve: 2016.11.05.)