



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Alapvető algoritmusok

1. előadás

Dr. Păţcaş Csaba



BABEŞ-BOLYAI TUDOMÁNYEGYETEM
Matematika és Informatika Kar





Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

1 Bevezető

2 Általános elvárások

- Osztályozás
- Előadás
- Szemináriumok és laborok

3 Laborfeladatokra vonatkozó alapszabályok

- DOMJudge
- Forráskódra vonatkozó szabályok



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok



1 Szakmai tudást

- Különbözhet a középiskolában tanultaktól (mert ennek más a célja, mint az egyetemnek)
- A hangsúlyt fektethetjük más dolgokra, mint ami a hallgató szerint fontos (ezen a téren meg kell bíznotok bennünk)
- Ha valamilyen elvárás nehezen teljesíthetőnek tűnik, annak valószínűleg jó oka van. Ha könnyebb lenne, csak veletek tolnánk ki hosszú távon. Nem kell most egyetérteni, de utólag (akár évek, évtizedek múlva) lehet jelezni nekünk, hogy mégis jobb volt úgy :)

Algoritmika

Dr. Pátcs
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok



1 Szakmai tudást

- Különbözhet a középiskolában tanultaktól (mert ennek más a célja, mint az egyetemnek)
- A hangsúlyt fektethetjük más dolgokra, mint ami a hallgató szerint fontos (ezen a téren meg kell bíznotok bennünk)
- Ha valamilyen elvárás nehezen teljesíthetőnek tűnik, annak valószínűleg jó oka van. Ha könnyebb lenne, csak veletek tolnánk ki hosszú távon. Nem kell most egyetérteni, de utólag (akár évek, évtizedek múlva) lehet jelezni nekünk, hogy mégis jobb volt úgy :)

2 Más készségeket, amik fontosak lehetnek az életben (mivel a tanári szakma *hivatás* is)

- Következetesség (azonos mérce mindenki számára, a leírtak/megbeszéltek betartása). Például: határidők, elvárások, pontozás
- **Felelősségvállalás** (bárki hibázhat, de vállalja is). Például: ha rossz jegyet kaptam, a tanár a hibás mert nehéz feladatot adott, vagy én nem teljesítettem megfelelően?



- Én irányotokba megmaradnék a tegeződésnél.
- Fordítva tegeződés (Csaba) és magázódás is elfogadott.
- Bármilyen kérdést bátran tegyetek fel az oktatónak és az általuk adott válaszok a hivatalosak, más forrásoknak ne tulajdonítsatok jelentőséget, mint pl. a diákok közti belső csatornákon elhangzó „fake news”-ok (pletykák, „én úgy hallottam”, „én úgy értettem”, „én úgy emlékszem” stb.)
- Vagyis: ha valamiben nem vagytok biztosak, minket kérdezzetek, a mai világban mostmár ott vagyunk egy kattintásra
- Amikor változás történik a hivatalos álláspontban és egyéb fontos közölni valókat előadáson mondom el és/vagy announcementben írom ki Canvas-re és/vagy DOMJudge-ra.



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

- Canvas személyes üzenet (egyenesen a házikra vagy announcement-ekre írt válaszokról nem kapok értesítést, így nagy eséllyel nem veszem észre őket)
- Válasz hétköznap 12-24 órán belül (az esetek minimum 90%-ában)
- Hivatalos üzenetek: `csaba.patcas@ubbcluj.ro`
- Kerüljük a más csatornákon való kommunikációt. Teams-en privát chat üzenetekre nem válaszolok!

Milyen szakmában szeretnél dolgozni leginkább?



Algoritmika

Dr. Pátcás
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Milyen szempontból lehet egy alkalmazás jó minőségű?



Algoritmika

Dr. Pátcás
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Milyen szempontból lehet egy alkalmazás jó minőségű?



Algoritmika

Dr. Pátcás
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

- Helyesség: pontosan megoldja a feladatot
- Megbízhatóság (stabilitás): szokatlan esetekben is intelligens módon viselkedik (pl. nem „fagy le”), többszöri futtatásra ugyanazokra a bemenetekre ugyanúgy viselkedik
- Karbantarthatóság (robusztusság): milyen könnyű a programterméket új kérélmekhez adaptálni (a szoftverköltiségek 70%-a)
- Újrafelhasználhatóság: egészben vagy részben felhasználható más alkalmazásokban
- Kompatibilitás: milyen könnyű más termékekkel kombinálni

Milyen szempontból lehet egy alkalmazás jó minőségű?



Algoritmika

Dr. Pátcás
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

- Hordozhatóság: milyen könnyű más környezethez (gép, konfiguráció, operációs rendszer) igazítani
- Hatékonyság: futási idő, memória, sávszélesség
- Barátságosság: intuitív a felhasználó számára
- Tesztelhetőség

Az internet és a mesterséges intelligencia korában miért van rá szükség, hogy algoritmusokról tanuljunk?



Algoritmika

Dr. Pátcás
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Az internet és a mesterséges intelligencia korában miért van rá szükség, hogy algoritmusokról tanuljunk?



Algoritmika

Dr. Pátcás
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok



Is 1 000 000 001 prime?



Yes, the number 1,000,000,001 is a prime number. It is only divisible by 1 and itself, without any other factors.



Are you sure?



I apologize for the mistake in my previous response. The number 1,000,000,001 is not a prime number. It is divisible by 7 and 142,857. I apologize for any confusion caused.





- Fő céljai az algoritmikus gondolkodás fejlesztése és az alapvető programozási módszerek elsajátítása és gyakorlása
- A kreditszám és az előző évi statisztikák alapján is a legnehezebb tárgyak egyike
- Okok?

Milyen tulajdonságokkal kell rendelkezzen egy jó programozó?



Algoritmika

Dr. Pátcás
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok



Milyen kevésbé egyértelmű készségeket fejleszt ez a tárgy?

- Alaposság (pl. specifikációk szövegét, syllabus-t végigolvasni, egy program csak akkor van „kész”, ha alaposan teszteltük)
- Kritikus gondolkodás (pl. pszeudokód átírása kódba ne másolásszerűen történjen, feladatokkal kapcsolatos kérdéseket tegyék fel, ha valami nem egyértelmű)
- Pontosság (pl. bemenet / kimenet formátuma)
- Türelem (pl. hibakeresés)
- Mentális állóképesség (ebben a szakmában általában valami „nem megy”)
- Időbeosztás, határidők betartása (akárcsak a software termékeknél)

Algoritmika

Dr. Pátcs Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok



1 Bevezető

2 Általános elvárások

- Osztályozás
- Előadás
- Szemináriumok és laborok

3 Laborfeladatokra vonatkozó alapszabályok

- DOMJudge
- Forráskódra vonatkozó szabályok



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

A Canvas-en található Syllabus oldal elolvasása kötelező elejétől végéig.



$$JEGY = \lceil \frac{I_1 + I_2 + L_1 + L_2 + H_1 + H_2}{6} \rceil, \text{ ahol}$$

- I_1 a részleges írásbeli vizsgára kapott jegy **felfelé kerekítve**
- I_2 a szessziós írásbeli vizsgára kapott jegy **felfelé kerekítve**
- L_1 a részleges laborvizsgára kapott jegy **felfelé kerekítve**
- L_2 a szessziós laborvizsgára kapott jegy **felfelé kerekítve**
- H_1 az 1., 2., 3. és 4. laborházikra kapott pontszámok átlaga **kerekítés nélkül**
- H_2 az 5., 6., 7. és 8. laborházikra kapott pontszámok átlaga **kerekítés nélkül**



- Mind a hat részleges jegy átmenő kell legyen, a legkisebb átmenő jegy az 5.00, a 4.99 bukó.
- Hasonló módon az 1 mp késéssel leadott feladat ugyanúgy elkésettnek számít, mint az 1 óra vagy 1 nap késéssel leadott és ha egy példára a helyes válasz 100, a 101 és a 99 is ugyanúgy helytelen, mint az 1 000 000.
- Bónuszpontszerzési lehetőségek megjelenhetnek a félév során bármelyik részjegyhez, ezek a bónuszpontok felhasználhatóak az adott részjegy átmenővé alakításához is.

Példa írásbeli vizsga feladatra



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

8. Legyen a következő algoritmus, amelynek paramétere az n , nullától különböző természetes szám és amely egy természetes számot térít vissza.

```
Algoritmus f(n):  
  j ← n  
  Amíg j > 1 végezd el  
    i ← 1  
    Amíg i ≤ n végezd el  
      i ← 2 * i  
    vége(amíg)  
    j ← j DIV 3  
  vége(amíg)  
  visszatérít j  
Vége(algoritmus)
```

A következő bonyolultsági osztályok közül melyikhez tartozik hozzá a fenti algoritmus időbonyolultsága?

- A. $O(\log_2 n)$
- B. $O(\log_2^2 n)$
- C. $O(\log_3^2 n)$
- D. $O(\log_2 \log_3 n)$

Példa laborvizsga feladatra



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Határozzuk meg két számsorozat leghosszabb közös tömbszakaszát! A bemeneti állomány első sorában a két sorozat hossza n és m található egy szóközzel elválasztva ($1 \leq n, m \leq 1000$). A második sor tartalmazza az első sorozat, a harmadik sor a második sorozat elemeit szóközzel elválasztva. A sorozatok elemei 32 bites előjeles egészek. A kimenet első sorába írjuk a leghosszabb közös tömbszakasz hosszát. A második sorba írjunk egy lehetséges megoldást!

Példa

Bemenet

5 7

1 1 2 5 4

1 6 3 5 5 4 5

Kimenet

2

5 4



- Az előadásokon a jelenlét nem kötelező, de erősen ajánlott.
- Aki jelen van, az ne zavarja a többiekét.
- Az óra végét egyértelműen jelzem.



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

- A jelenlét kötelező, legtöbb két hiányzás megengedett a félév során.
- Jelenlétet lehet kiváltani más csoporttal, de csak ugyanazon a héten, az érintett tanár(ok) beleegyezésével.
- Akinek több mint két hiányzása van, csak akkor jöhet pótszesszióban vizsgázni, ha megvan a jelenléteknek legalább a fele.



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

1 Bevezető

2 Általános elvárások

- Osztályozás
- Előadás
- Szemináriumok és laborok

3 Laborfeladatokra vonatkozó alapszabályok

- DOMJudge
- Forráskódra vonatkozó szabályok



- A három fő szempont: helyesség, hatékonyság és programozási stílus
- Tartsuk be a megadott bemenet / kimenet pontos formátumát!
- Figyeljünk oda a használt adattípusokra: ne használjunk számok esetén mindig automatikusan `int`-et! Ez valós számok esetén nyilvánvalóan nem helyes, de akkor sem ha a bemenet, kimenet, vagy valamilyen belső részeredmény túlcsordul az adott adattípus határain (pl. 15 számjegyű egész szám tárolására már nem elég az `int`, hanem `long long` szükséges).
- Ha egy feladatot többféleképpen meg lehet oldani, akkor a jobb időbonyolultságú megoldást választjuk és ha több azonos időbonyolultságú megoldás létezik, akkor ezek közül a jobb memóriabonyolultságú változatot választjuk. A beolvasást és kiíratást végző alprogramok hatékonysága másodlagos kritérium, a megoldást végző alprogramok hatékonysága a fontosabb.
- Programozási stílus a Syllabusban leírt és az alábbi részletek betartásával



- A házi feladatok és laborvizsgák kiértékelését segítő automatizált rendszer, helyességet és hatékonyságot ellenőriz.
- Ez a második év amikor használjuk.
- Elérési link: <http://dj.canvas2.cs.ubbcluj.ro/domjudge/team>
- Rövid használati útmutató angol nyelven [itt](#)



- A rendszer 6 magos processzoron 16 GB memóriával rendelkező gépen Debian Linux 4.19-es operációs rendszeren fut és gcc 8.3.0 valamint fpc 3.0.4 verziójú fordítóprogramokat használ.
- Jelenleg támogatott programozási nyelvek: C, C++ és Pascal. C++-ban a C++14-es standardot, C-ben a C11-es standardot kell követni. Más programozási nyelvek használatát lehet igényelni a laboránsoknál.
- A forráskódok lefordítása a következő paraméterekkel történik:

```
g++ -std=c++14 -pedantic-errors -Werror=uninitialized -x c++ -Wall -O2 -static -pipe -o  
gcc -std=c11 -pedantic-errors -Werror=uninitialized -x c -Wall -O2 -static -pipe -o  
fpc -viwn -O2 -Sg -XS -o
```



- Minden beküldött feladatra lefuttat valahány (a diák nem tudja hány) tesztesetet és visszajelzést ad az eredményről.
- Egy-egy feladatot legtöbb 15-ször lehet beküldeni a DOMjudge-ra.
- Csak az első helyes megoldásig számolódik a beküldések száma, utána tetszőlegesen lehet beküldeni az adott feladathoz (pl. a kód "szépítése", a programozási stílus javítása érdekében).
- Az utolsó beküldést fogjuk figyelembe venni.
- Aki meghaladja a 15 próbálkozást, -5 pontot kap a teljes házicsomagra.
- Miért szükséges ez a szabály?



- Minden beküldött feladatra lefuttat valahány (a diák nem tudja hány) tesztesetet és visszajelzést ad az eredményről.
- Egy-egy feladatot legtöbb 15-ször lehet beküldeni a DOMjudge-ra.
- Csak az első helyes megoldásig számolódik a beküldések száma, utána tetszőlegesen lehet beküldeni az adott feladathoz (pl. a kód "szépítése", a programozási stílus javítása érdekében).
- Az utolsó beküldést fogjuk figyelembe venni.
- Aki meghaladja a 15 próbálkozást, -5 pontot kap a teljes házicsomagra.
- Miért szükséges ez a szabály? Hogy minden beküldést alapos átgondolás és tesztelés előzze meg és a visszaélések elkerülése végett.



- CORRECT = helyes
- COMPILER-ERROR = fordítási hiba

- TIMELIMIT = időtúllépés

A feladat kijelentésében lesz az adott feladatra vonatkozó időlimit, ez az esetek többségében 1 másodperc lesz.

- RUN-ERROR = futási hiba

Memóriatúllépés is okozhatja, általában 100 MB felhasznált memóriát próbáljuk meg ne túllépni. Azoknál a feladatoknál ahol erre mégis szükség lesz, nagyobbra lesz állítva a limit.



- **OUTPUT-LIMIT** = túl nagy a kimenet
A standard kimenetlimit 4 MB, azon feladatok esetén ahol szükséges, nagyobbra lesz állítva.
- **WRONG-ANSWER** = hibás eredmény
Figyeljünk a pontos kimeneti formátumra!
- **TOO-LATE** = lejárt a beküldési határidő
Egy másodperc késés is késés!
Mivel a határidő lejártá után még egy hétig fele pontszámra lehet még beküldeni házit, a DOMjudge a verseny befejezési idejének ezt az egy héttel kitölt időpontot fogja jelezni.



- Minden feladathoz a megoldáson kívül fel kell tölteni Canvas-re 3 bemeneti és 3 kimeneti állományt, melyek három különböző struktúrájú tesztet tartalmaznak és közülük legalább egy „nagy” teszt (n értéke a maximálisnak legalább 75%-a).
- Minden feladat esetén a diák utolsó beküldését vesszük figyelembe.
- A helyesen megoldott tesztesetek jelentik a pontszám 80%-át.
- A maradék pontszám a programozási stílusra jár, amit a javítótanár a programkód minőségére ad saját belátása szerint. Ezt a 20%-ot csak akkor lehet megkapni, ha minden tesztesetre helyesen futott le a program és feltöltöttük a feladathoz tartozó 6 tesztállományt.
- A 20%-ból abban az esetben is levonható pontszám ha a javítótanár a helyességgel kapcsolatos problémát talál, mely az automatizált tesztelés során nem derült ki.



- A vizsga idejére a beküldéseket csak pár publikus tesztre futtatjuk.
- Minden feladat esetén a diák utolsó beküldését vesszük figyelembe.
- Ha erre fut a program a feladat kijelentésében megadott példára (amely mindig szerepelni fog a kezdeti publikus tesztek között), a feladat pontozásra kerül, ellenkező esetben nem ér pontot.
- A feladat pontozását a javítótanár végzi a kód ellenőrzésével. Ehhez az ellenőrzéshez felhasználhat a vizsga befejezése után utólagosan hozzáadott teszteseteket.
- Miért csak azokat a feladatokat pontozzuk, amelyek a kijelentésben megadott példára helyesen futnak?



- A vizsga idejére a beküldéseket csak pár publikus tesztre futtatjuk.
- Minden feladat esetén a diák utolsó beküldését vesszük figyelembe.
- Ha erre fut a program a feladat kijelentésében megadott példára (amely mindig szerepelni fog a kezdeti publikus tesztek között), a feladat pontozásra kerül, ellenkező esetben nem ér pontot.
- A feladat pontozását a javítótanár végzi a kód ellenőrzésével. Ehhez az ellenőrzéshez felhasználhat a vizsga befejezése után utólagosan hozzáadott teszteseteket.
- Miért csak azokat a feladatokat pontozzuk, amelyek a kijelentésben megadott példára helyesen futnak? Mert egy elsőéves informatikus egyetemistától alapvető elvárás, hogy 20-30 perc alatt olyan programot tudjon írni, amely legalább a megadott példára helyesen működik.



Kérdés: Miért kap Wrong Answer-t a programom? Nálam tökéletesen működik, minden lehetséges esetre kipróbáltam.

Válasz:

- Nem próbáltad ki minden lehetséges esetre :)
- Nem tartottad be precízen a kimenet formátumát.

Ha látszik a Scoreboard-on, hogy már többen megoldottak helyesen egy feladatot, akkor elég valószínű, hogy a hiba a diáknál van.



- Mivel Windows és Linux operációs rendszerek alatt más az újsor karakter kódja, mikor stringeket olvasunk be ne tegyünk semmilyen ezzel kapcsolatos feltételezést.
- Például előfordulhat, hogy a beolvasott karakterlánc végén ott van egy vagy több „idegen” karakter is, tehát a string hosszát visszatérítő függvényekre sem támaszkodhatunk.
- Emiatt az ilyen feladatoknál meg lesz adva előre a string hossza.



Példa:

5

abcde

Lehetséges feldolgozási mód C++-ban:

```
int n;  
string s;  
cin >> n >> s;  
for(int i = 0; i < n; ++i) cout << s[i];
```



- Jelentős elakadás (több óra) esetén lehet segítséget kérni laboron, vagy a DOMJudge Clarification rendszerét használva.
- Ezekre a limit 5 Clarification per házicsomag per diák, specifikusan más diákra vonatkozó kérdést nyilván nem lehet beküldeni.
- A tesztek tartalma titkos, csak „tippszerű” segítségre lehet számítani.
- Miért nem mutatja a rendszer, hogy milyen bemenetre hibázik a program?



- Jelentős elakadás (több óra) esetén lehet segítséget kérni laboron, vagy a DOMJudge Clarification rendszerét használva.
- Ezekre a limit 5 Clarification per házicsomag per diák, specifikusan más diákra vonatkozó kérdést nyilván nem lehet beküldeni.
- A tesztek tartalma titkos, csak „tippszerű” segítségre lehet számítani.
- Miért nem mutatja a rendszer, hogy milyen bemenetre hibázik a program? Mert nem a *cél* a lényeg (legyen meg a házi), hanem az *odáig vezető út* (mit tanulok a kihívásokkal teli órák, napok alatt amíg elkészül).



Algoritmika

Dr. Pátcás
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Rövid demo

Login szabály, avagy *login rule*



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Login szabály

Más felhasználójával belépni DOMJudge-ra (és értelemszerűen arról kódot beküldeni) tilos, ellenkező esetben -5 pont jár a teljes csomagra mindkét érintett félnek.

Metaadat szabály, avagy *metadata rule*



Algoritmika

Dr. Păţcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Metaadat szabály

A házi feladatok esetén az első sorok, kommentként kell tartalmazzák a hallgató nevét, csoportját, a feladat sorszámát és kijelentését. Ellenkező esetben a feladatra a pontszám legtöbb 90%-a kapható.



Kemény kód limitálása, avagy *no hardcoding rule*

Hardcode szabály

Tilos a forráskódokban előre megállapított (hardcode-olt) kimeneteket kiíratni direkt vagy indirekt módon, annak érdekében, hogy a példaként megadott tesztekre helyes legyen az eredmény. Ellenkező esetben -10 pont jár a teljes házicsomagra, laborvizsgán -4 pont az adott feladatra.

```
#include <iostream>
using namespace std;
int main() { unsigned long long n; cin >> n; cout << n - 4; return 0; }
```

```
//Nem emlékszem a radix sort-ra de egy probat meger
#include <iostream>
#include <vector>
#include <math.h>
using namespace std;
int main() {
    unsigned long long n;
    cin >> n;
    vector<unsigned long long> num;
    num.resize(n + 1);
    unsigned long long i = 0;
    while (i < n) {
        cin >> num[i];
        i += 1;
    }
    unsigned long long res;
    cout << 2 << endl << 76 << endl << 876 << endl << 12345 << endl << 12354;
    return 0;
}
```

Algoritmika

Dr. Pátcs
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok



Olvashatósági szabály, avagy *no obfuscation rule*

Olvashatósági szabály

Ne hagyjunk a forráskódban programrészeket melyek nem hajtódnak végre, alprogramokat, melyeket nem hívunk meg, vagy függvényeket melyek visszatérítési értékét nem használjuk fel. Ezek jelentősen rontják a kód olvashatóságát, nehezítik a javítótanár dolgát, aki abban az esetben ha megtévesztőnek ítéli a forráskódot, tetszőlegesen levonhat a feladatra járó pontszámból.

```
void backtrack(vector<string> utak, int i, bool volt[], int m, int eredm[], int n, int &k)
{
    if (i < m)
    {
        return;
    }
    ... (nincs kiíratás sehol)
```

```
63     bool volt[30] = {false};
64     int k = 1;
65     int eredm[30] = {0};
66     cout << "Kolozsvar Nagyvarad Marosvasarhely Kolozsvar" << endl
67          << "Kolozsvar Marosvasarhely Nagyvarad Kolozsvar";
68     backtrack(utak, 1, volt, m, eredm, n, k);
69
70     return 0;
71 }
```

Algoritmika

Dr. Pátcaş
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Mesterséges intelligencia szabály, avagy *no AI rule*



Algoritmika

Dr. Pátcás
Csaba

Bevezető

Elvárások

Osztályozás

Előadás

Szemináriumok és
laborok

Laborfeladatok

DOMJudge

Szabályok

Mesterséges intelligencia szabály

Mesterséges intelligencia (ChatGPT, Copilot stb.) által generált kód beküldése tilos, ellenkező esetben -10 pont jár a teljes csomagra. Természetes nyelvben adott szöveges magyarázatokat lehet kérni mesterséges intelligenciától.