Nemes Tihamér Sillabusz

A Nemes Tihamér NITV Programozás kategória anyagának kivonata

1 Verzió és státusz

Ez egy nem hivatalos javaslat a 2020/2021. tanévi Nemes Tihamér NITV Programozás kategória második és harmadik fordulójában szereplő feladatok témaköreinek leírására.

Jelen verzió még szerkesztés alatt áll, nem teljes, és nem megosztásra szánt.

A Nemes Tihamér Sillabusz (továbbiakban NT Sillabusz) az IOI Syllabus mintájára készül, és jelenleg még fejlesztés alatt áll. Ha eléri célját, egy hivatalos dokumentum alakul ki belőle, amelynek aktuális verzióját a versenybizottság hagyja jóvá és teszi közzé minden évben. Az évek során a dokumentum változhat, fejlődhet.

2 Szerzők és elérhetőségeik

Szívesen fogadunk bármilyen visszajelzést és javaslatot a sillabusszal kapcsolatban a jelenlegi szerkesztő e-mail címére küldve (`laszlo.nikhazy@gmail.com`).

Azok számára, akik szeretnének hozzájárulni a dokumentum fejlesztéséhez, vagy hozzászólni, javaslatokat tenni a tartalmához, az NT Sillabusz GitHub repository-jában találhatók információk. Minden javaslatot, hozzászólást és segítséget szívesen fogadunk. https://github.com/niklaci/NT-Syllabus.

3 Bevezetés

Az NT Sillabusz az IOI Syllabus-hoz képest erősen rövidített, az áttekinthetőség kedvéért. A konkrét tárgyi ismereteket és módszereket soroljuk fel, míg

a készségekről (például hibakeresés) és eszközök használatáról (például fejlesztőkörnyezet) nem teszünk említést.

Ez a dokumentum kifejezetten a 2. és 3. forduló (gépes) feladatairól szól. Az első fordulóban szélesebb körből fordulhatnak elő feladatok, mert ott elvárt egy-egy (esetleg ismeretlen) számítástechnikai témakör alapszintű megértése a feladatleírás alapján.

A Sillabusz az alábbi célokat szolgálja.

- Meghatározza a versenyre szükséges előzetes tudást.
- Segít a versenyzők felkészülésében, a tanároknak a felkészítés tervezésében.
- A feladat kitűzőknek támpontot ad, hogy egy-egy feladat melyik kategóriában lehet.

A fenti célok elérése érdekében az NT Sillabuszban megtalálható minden olyan téma (algoritmusok, adatszerkezetek, módszerek, nyelvi eszközök), amely egyáltalán szóba jöhet a versenyen, és ezeket kategóriákba soroljuk. A kategóriák tükrözik azt, hogy milyen módon fordulhat elő egy témakör. Az IOI Syllabus-nak kategóriákat alkalmazunk, az alábbiakban felsoroljuk, majd bővebben magyarázzuk őket. A verseny három korcsoportjában különböző kategóriába sorolhatók a témakörök.

✓	Lehet, megkötések nélkül
✓	Lehet, de magyarázandó
✓	Lehet, de a feladatleírásban nem
?	Hatáskörön kívüli
✗	Nem lehet, de nyitott a megvitatásra
✗	Kizárt

4 Kategóriák

This Syllabus classifies a selection of topics into six different categories. Obviously, such a set of topics can never be exhaustive. Instead, the list given in this Syllabus should serve as examples that map out the boundary. Topics not explicitly mentioned in the Syllabus should be classified as follows:

- Anything that is a prerequisite of an Included topic is also Included.
- Anything that is an extension of an Excluded topic or similar to an Excluded topic is also Excluded.

- Anything else that is not mentioned in the Syllabus is considered Outside of focus.

If there is a particular topic for which you are not sure how it should be classified, we invite you to submit a clarification request to the current Syllabus maintainer.

A hat lehetséges besorolás:

✓ Lehet, megkötések nélkül
Ebben a kategóriában lévő témakörök előzetes tudásnak tekintendők. A versenyzőktől elvárt az ismeretük. A feladatleírásban magyarázat nélkül előfordulhatnak.

Példa: *Egész típus* §5.1 PA1

✓🗎 Lehet, de magyarázandó
A versenyzőknek ismerniük kell ezt a témakört, de amikor feladatleírásban szerepel, akkor definiálni kell. Általában akkor használjuk ezt a besorolást, ha egy ✓ témát többféleképpen is lehet értelmezni.

Példa: *Feszítőfa* §6.2 DS5

✓🗎 Lehet, de a feladatleírásban nem
Olyan témakörök sorolhatók ide, amelyeknek ismeretére a versenyzőknek a feladatmegoldás során van szüksége. Tehát a feladat szövegében nem szerepelhetnek.

Példa: *Aszimptotikus felső becslés a komplexitásra* in §5.2 AL1

Ez egy központi téma a verseny kapcsán, hiszen egy helyes program az algoritmus komplexitása alapján kap pontot. Mégsem fordulhat elő a feladatleírásban a komplexitás fogalma, a precíz definíció ismerete nem is elvárt.

It should be noted that this set of topics contains a wide range of difficulties, starting from simple concepts and ending with topics that can appear in problems that aim to distinguish among the gold medallists. It is **not** expected that all contestants should know everything listed in this category.

? Hatáskörön kívül
Bármilyen téma, amit nem említ a Sillabusz, ebbe a kategóriába esik. A versenyzőktől nem elvárt, hogy ismerjék ezeket. A legtöbb versenyfeladat nem kapcsolódik ilyen témakörökhöz. De ez nem zárja ki, hogy

a versenyen legyen olyan feladat, ami ilyen témakörhöz kapcsolódik. A versenybizottság kitűzhet ilyen feladatot a verseny színesítése érdekében, de ebben az esetben is megoldhatónak kell lennie a fenti kategóriákba eső ismeretekkel.

✗ Nem lehet, de nyitott a megvitatásra

A Sillabusz az évek során változik, fejlődik a versennyel együtt, így lehetőség adódik arra, hogy egy (valamely korcsoportban) kizárt téma később bekerüljön a verseny anyagába. Általában ezek a témakörök kapcsolódnak megengedett témakörökhöz, és nehéz határt szabni. Ebben az esetben a megvitatásra nyitott kategóriába kerülnek ezek a témakörök, ezzel is bíztatva a szakmai közösséget a visszajelzésre a kérdést illetően. Ha bekerül az idők során egy témakör, akkor a kizárt kategóriából először ebbe kell kerülnie.

Példa: *Szegmensfa* §5.2 Al4

✗? Kizárt

Ebbe a kategóriákba főként nehéz algoritmikus módszerek, bonyolult matematikai fogalmak tartoznak. Garantált, hogy nem lesz olyan feladat a versenyen, amelynek megoldásához elengedhetetlen egy ilyen témakör ismerete. Szintén kívánatos, hogy ezeknek a témáknak az ismerete ne nyújtson utat egyszerűbb, vagy több pontot érő megoldáshoz. Ugyanakkor előfordulhat, hogy egy feladat témaköre kapcsolódik egy ilyen témához, de ennek ismerete nem segít lényegesen a megoldásban.

Példa: *Heavy-light decomposition* §5.2 AL4

A dokumentum további része a témaköröket és azok korcsoportonkénti besorolását tartalmazza.

5 Számítástechnika, számítástudomány

5.1 Progamozási Alapok (PA)

PA1. Alapvető programozási eszközök

NT1	NT2	OKTV	Leírás
✓	✓	✓	Alapvető szintaxisa és szemantikája egy versenyen megengedett programozási nyelvnek

NT1	NT2	OKTV	Leírás
✓	✓	✓	Változók, típusok, műveletek, kifejezések és értékadás
✓	✓	✓	Elágazások és ciklusok
✓	✓	✓	Függvények és paraméterátadás
✓	✓	✓	Egyszerű beolvasás és kiírás (A standard input/output ismerete kell.)
✗?	✗?	✗?	Fájlba írás és olvasás

PA2. Alapvető adatszerkezetek

NT1	NT2	OKTV	Leírás
✓	✓	✓	Elemi adattípusok (logikai, egész, karakter)
✓	✓	✓	Tömbök
✓	✓	✓	Két vagy több dimenziós tömbök
✓	✓	✓	Karekterláncok (string) és feldolgozásuk
✓📄	✓📄	✓📄	Valós számok használata egyszerű és numerikusan stabil számításokra
✓📄	✓📄	✓📄	A valós számok lebegőpontos reprezentációja, pontossági hibák léte. [1]
✗?	✓📄	✓	Mutatók (pointer) és referenciák
✗	✗?	✓	Láncolt adatszerkezetek
✗?	✗?	✓📄	Törtszámok használata pontos számításokra
?	?	?	Adatok memóriaképe
?	?	?	Dinamikus memóriafoglalás
✗	✗	✗	Nemtriviális számítások lebegőpontos számokon, pontossági hibák kiküszöbölése

PA3. Rekurzió

NT1	NT2	OKTV	Leírás
✓	✓	✓	A rekurzió fogalma

[1] Amikor csak lehetséges, a lebegőpontos számítások teljes elkerülése az előnyösebb megoldás.

NT1	NT2	OKTV	Leírás
✓	✓	✓	Rekurzív matematikai függvények
✓	✓	✓	Egyszerű rekurzív függvények (több függvény kölcsönös rekurziója is)
✗	✓	✓	A visszalépéses keresés rekurzív változata
✗	✗	✓	Oszd meg és uralkodj stratégia

5.2 Algoritmusok és komplexitásuk (AL)

AL1. Algoritmusok elemzése alapszinten

NT1	NT2	OKTV	Leírás
✓	✓	✓	Algoritmus specifikáció, előfeltétel, utófeltétel, helyesség, invariánsok
✓	✓	✓	Aszimptotikus felső becslés a komplexitásra (lehetőleg nem formálisan)
✗	✓	✓	$\mathcal{O}$ (ordó) jelölés a komplexitásra
✓	✓	✓	Szokásos nagyságrendi osztályok: konstans, logaritmikus, lineáris, $\mathcal{O}(n \log n)$, négyzetes, köbös, exponenciális stb.
✓	✓	✓	Algoritmusok idő- és tárigényének optimalizálása
✗	✓	✓	Amortizált komplexitás elemzése
✓	✓	✓	Hatékonyság mérése empirikusan
?	?	?	kis ordó, nagy omega és theta jelölések
?	?	?	Paraméterek hangolása a futási idő vagy memória csökkentése érdekében.
✗	✗	✗	Átlagos komplexitásra aszimptotikus becslések
✗	✗	✗	Rekurziós összefüggések használata komplexitás elemzéskor

AL2. Algoritmikus stratégiák

NT1	NT2	OKTV	Leírás
✓	✓	✓	Egyszerű ciklustervezéses stratégiák
✓	✓	✓	Kimerítő keresés (brute force)
✓	✓	✓	Mohó algoritmusok
✓	✓	✓	Dinamikus programozás
✓	✓	✓	Rekurzív kiszámítás
✓	✓	✓	Két mutató (2 pointers) technika
✗	✓	✓	Bináris keresés szélsőérték meghatározására
✗	✓	✓	Visszalépéses keresés (backtrack, rekurzív és nem rekurzív is)
✗	✓	✓	Elágazás és korlátozás
✗	✓	✓	Oszd meg és uralkodj elv
?	?	?	Heurisztikák
?	?	?	Közelítő algoritmusok
?	?	?	Randomizált algoritmusok
✗	✗	✗	Klaszterező algoritmusok (pl. k-means)
✗	✗	✗	Többváltozós függvények szélsőérték keresése numerikus módszerekkel

AL3. Algoritmusok

NT1	NT2	OKTV	Leírás
✓	✓	✓	Egyszerű számelméleti algoritmusok: számrendszer átváltás, Euklideszi algoritmus (LNKO-ra), prímteszt $\mathcal{O}(\sqrt{n})$ osztókereséssel, Eratosztenészi szita, prímfelbontés (osztókereséssel vagy szitával)
✗	✗	✓	Gyors hatványozás (négyzetre emelésekkel)
✗	✗	✓	Műveletek tetszőlegesen nagy egész számokkal (összeadás, kivonás, szorzás)
✓	✓	✓	Egyszerű programozási tételek tömbökön: összegzés, megszámolás, keresés, minimum/maximum, kiválogatás
✓	✓	✓	Programozási tételek összeépítése, pl. feltételes maximum
✓	✓	✓	Rendezett sorozatok összefésülése, metszet, unió

NT1	NT2	OKTV	Leírás
✓	✓	✓	Egyszerű string algoritmusok (pl. minta keresése naiv módszerrel)
✓	✓	✓	$\mathcal{O}(n^2)$ rendezések (buborék, leszámláló, minimum-kiválasztásos)
✗	✓	✓	Gyorsrendezés (quicksort), *NT1-ben a sort() függvény használata elvárt, de az algoritmus ismerete nem*
✗	✓	✓	$\mathcal{O}(n \log n)$ rendezések (kupac, összefésüléses)
✗	✗	✓	Lineáris idejű rendezések (láda/vödör rendezés, radix rendezés) [2]
✗	✓	✓	Rekurzív fabejárás
✗	✓	✓	Rendezett fák bejárásai (pre, in- és post-order)
✗	✓	✓	Szélességi és mélységi gráfbejárás
✗	✓	✓	Összefüggő komponensek meghatározása
✗	✓	✓	A mélységi feszítőfa alkalmazásai, például topologikus sorrend
✗	✓	✓	Irányított körmentes gráf emeletekre bontása
✗	✓	✓	Euler-séta/körséta keresése
✗	✓	✓	Legrövidebb utak súlyozott gráfokban (Dijkstra, Bellman-Ford, Floyd-Warshall)
✗	✓	✓	Minimális feszítőfa keresése (Prim és Kruskal algoritmus)
✗	✗	✓	Elvágó pontok, hídélek keresése, kétszeres (pont- ill. él-) összefüggőség
✗	✗	✓	Irányított gráfok összefüggősége, erősen összefüggő komponensek
✗	✗	✓	Páros gráfban maximális párosítás keresése magyar módszerrel ($O(VE)$ időben)
✓	✓	✓	Kombinatorikus játékok alapjai, nyerő és vesztő pozíciók
✗	✓	✓	Minimax algoritmus kétszemélyes játékok optimális stratégiájára
✗	✗	✗	Hálózati folyamok, maximális folyam és minimális vágás keresése

[2] A gyakorlatban a futási időt lehetetlen megkülönböztetni a beépített rendezéstől, így csak kifejezetten erről szóló feladatnál kell.

NT1	NT2	OKTV	Leírás
✗	✗	✗	Matroidok és hozzájuk kapcsolódó optimalizálási problémák
✗	✗	✗	Lexikografikus szélességi bejárás, maximum adjacency search
✗	✗	✗?	Haladó string algoritmusok: KMP, Z-algoritmus

AL4. Adatszerkezetek (összetett adatszerkezetek)

NT1	NT2	OKTV	Leírás
✓	✓	✓	Verem és sor
✓	✓	✓	Prefix összeg (kumulatív összeg)
✓	✓	✓	Számláló tömb, hisztogram
✗?	✓	✓	Gráfreprezentációk: mátrix, szomszédsági lista, éllista
✗	✓	✓	Unió-holvan (DSU)
✗	✓	✓	Kupac és hasonló bináris fa reprezentációk
✗	✓	✓	Prioritási sor használata (kupac ismerete nélkül)
✗	✓	✓	Beépített set és map használata
✗	✗	✗?	Szegmensfa, Fenwick-fa, és hasonló statikusan kiegyensúlyozott bináris keresőfák
✗	✗	✗	Kiegyensúlyozott bináris keresőfák (pl. AVL-fa, piros-fekete fa stb.) [3]
✗	✗	✗?	LCA (legalacsonyabb közös ős fában) meghatározása $O(\log n)$ időben.
✗	✗?	✓	Adatszerkezetek egymásba ágyazása (például halmazok halmaza).
✗	✗	✓	Szófa (trie)
✗	✗	✗	Összetett string adatszerkezetek (suffix arrays/tree, suffix automaton, Rabin-Karp hashing)
✗	✗	✗	Perzisztens adatszerkezetek (például perzisztens szegmensfa)
✗	✗	✗	Heavy-light decomposition

[3] Ugyan nem kell ismerni ezeket az adatszerkezeteket, de az erre épülő sztenderd tárolók (set, map) használata szükséges lehet.

NT1	NT2	OKTV	Leírás
✗	✗	✗	Dinamikusan változó fákhoz fejlett adatszerkezetek
✗	✗	✗	Bonyolult kupac-variációk (Fibonacci, binomiális)
✗	✗	✗	Hash-táblák implementálása, illetve használata [4]
✗	✗	✗	Két dimenziós fa alapú adatszerkezetek (pl. 2D Fenwick-fa)

AL4. Distributed algorithms

This entire section is **?**.

AL5. Basic computability

All topics related to computability are ✗. This includes the following: Tractable and intractable problems; Uncomputable functions; The halting problem; Implications of uncomputability.

However, see AL7 for basic computational models.

AL6. The complexity classes P and NP

Topics related to non-determinism, proofs of NP-hardness (reductions), and everything related is ✗.

Note that this section only covers the results usually contained in undergraduate and graduate courses on formal languages and computational complexity. The classification of these topics as ✗ does not mean that an NP-hard problem cannot appear at an IOI.

AL7. Automata and grammars

✓ Understanding a simple grammar in Backus-Naur form

[4] Természetesen lehet használni a beépített implementációkat, de nem lehet olyan feladat, ami enélkül nem oldható meg.

? Formal definition and properties of finite-state machines,
? Context-free grammars and related rewriting systems,
? Regular expressions

✗ Properties other than the fact that automata are graphs and that grammars have parse trees.

AL8. Advanced algorithmic analysis

✓ Amortized analysis.

? Online algorithms
? Randomized algorithms

✗ Alpha-beta pruning

AL9. Cryptographic algorithms

This entire section is ?.

AL10. Geometric algorithms

In general, the ISC has a strong preference towards problems that can be solved using integer arithmetics to avoid precision issues. This may include representing some computed values as exact fractions, but extensive use of such fractions in calculations is discouraged.

Additionally, if a problem uses two-dimensional objects, the ISC prefers problems in which such objects are rectilinear.

✓ Representing points, vectors, lines, line segments.
✓ Checking for collinear points, parallel/orthogonal vectors and clockwise turns (for example, by using dot products and cross products).
✓ Intersection of two lines.
✓ Computing the area of a polygon from the coordinates of its vertices.[5]

[5]The recommended way of doing so is to use cross products or an equivalent formula. TODO url

- ✓📄 Checking whether a (general/convex) polygon contains a point.
- ✓📄 Coordinate compression.
- ✓📄 $\mathcal{O}(n \log n)$ time algorithms for convex hull
- ✓📄 Sweeping line method

- ✗ Point-line duality
- ✗ Halfspace intersection, Voronoi diagrams, Delaunay triangulations.
- ✗ Computing coordinates of circle intersections against lines and circles.
- ✗ Linear programming in 3 or more dimensions and its geometric interpretations.
- ✗ Center of mass of a 2D object.
- ✗ Computing and representing the composition of geometric transformations if the knowledge of linear algebra gives an advantage.

AL11. Parallel algorithms

This entire section is **?**.

6 Matemaika

6.1 Aritmetika és geometria

NT1	NT2	OKTV	Leírás

- ✓ Integers, operations (incl. exponentiation), comparison
- ✓ Basic properties of integers (sign, parity, divisibility)
- ✓ Basic modular arithmetic: addition, subtraction, multiplication
- ✓📄 Prime numbers
- ✓ Fractions, percentages
- ✓ Line, line segment, angle, triangle, rectangle, square, circle
- ✓ Point, vector, coordinates in the plane

- ✓ Polygon (vertex, side/edge, simple, convex, inside, area)
- ✓ Euclidean distances
- ✓ Pythagorean theorem

- ✗ Additional topics from number theory.

- ✗ geometry in 3D or higher dimensional spaces
- ✗ analyzing and increasing precision of floating-point computations
- ✗ modular division and inverse elements
- ✗ complex numbers,
- ✗ general conics (parabolas, hyperbolas, ellipses)
- ✗ trigonometric functions

6.2 Discrete Structures (DS)

DS1. Functions, relations, and sets

- ✓ Functions (surjections, injections, inverses, composition)
- ✓ Relations (reflexivity, symmetry, transitivity, equivalence relations, total/linear order relations, lexicographic order)
- ✓ Sets (inclusion/exclusion, complements, Cartesian products, power sets)

- ✗ Cardinality and countability (of infinite sets)

DS2. Basic logic

- ✓ First-order logic
- ✓ Logical connectives (incl. their basic properties)
- ✓ Truth tables
- ✓ Universal and existential quantification (Note: statements should avoid definitions with nested quantifiers whenever possible.)
- ✓ Modus ponens and modus tollens

- ? Normal forms

- ✗ Validity
- ✗ Limitations of predicate logic

DS3. Proof techniques

- ✓ Notions of implication, converse, inverse, contrapositive, negation, and contradiction
- ✓ Direct proofs, proofs by: counterexample, contraposition, contradiction
- ✓ Mathematical induction
- ✓ Strong induction (also known as complete induction)
- ✓ Recursive mathematical definitions (incl. mutually recursive definitions)

DS4. Basics of counting

- ✓ Counting arguments (sum and product rule, arithmetic and geometric progressions, Fibonacci numbers)
- ✓ Permutations and combinations (basic definitions)
- ✓ Factorial function, binomial coefficients
- ✓ Inclusion-exclusion principle
- ✓ Pigeonhole principle
- ✓ Pascal's identity, Binomial theorem

- ✗ Solving of recurrence relations
- ✗ Burnside lemma

DS5. Graphs and trees

- ✓ Trees and their basic properties, rooted trees
- ✓ Undirected graphs (degree, path, cycle, connectedness, Euler/Hamilton path/cycle, handshaking lemma)
- ✓ Directed graphs (in-degree, out-degree, directed path/cycle, Euler/Hamilton path/cycle)
- ✓ Spanning trees
- ✓ Traversal strategies
- ✓ 'Decorated' graphs with edge/node labels, weights, colors
- ✓ Multigraphs, graphs with self-loops
- ✓ Bipartite graphs
- ✓ Planar graphs

- ✗ Hypergraphs

- ✗ Specific graph classes such as perfect graphs
- ✗ Structural parameters such as treewidth and expansion
- ✗ Planarity testing
- ✗ Finding separators for planar graphs

DS6. Discrete probability

Applications where everything is finite (and thus arguments about probability can be easily turned into combinatorial arguments) are **?**, everything more complicated is ✗.

6.3 Other Areas in Mathematics

- ✗ Geometry in three or more dimensions.
- ✗ Linear algebra, including (but not limited to):
  - Matrix multiplication, exponentiation, inversion, and Gaussian elimination
  - Fast Fourier transform
- ✗ Calculus
- ✗ Theory of combinatorial games, e.g., NIM game, Sprague-Grundy theory
- ✗ Statistics