国際情報オリンピックシラバス
The International Olympiad in Informatics Syllabus
日本語版 (ベータ版) *

2014 年 11 月 1 日　日本語版 (ベータ版) 作成

1　バージョン・状態 (Version and status information)

これは，オーストラリアのブリスベンで行われた **IOI 2013** の公式シラバスである．[*1]

This is the official Syllabus version for **IOI 2013** in Brisbane, Australia.

このシラバスは IOI に関する公式文書である．毎年，その年の IOI のために，シラバスの更新版が ISC により作成される（IOI Regulations, Statue 3.13 を参照）．

The Syllabus is an official document related to the IOI. For each IOI, an up-to-date version of the Syllabus is produced by the ISC, as described in the IOI Regulations, Statue 3.13.

2　著者・連絡先 (Authors and Contact Information)

IOI シラバスの原案の共著者は Tom Verhoeff, Gyula Horváth, Krzysztof Diks, Gordon Cormack である．2007 年より，このシラバスは Michal Forišek により管理されている．

The original proposal of the IOI Syllabus was co-authored by Tom Verhoeff[*2], Gyula Horváth[*3], Krzysztof Diks[*4], and Gordon Cormack[*5]. Since 2007, the maintainer of the Syllabus is Michal Forišek[*6].

シラバスに関する意見・評価を歓迎する．送付先は現バージョンの管理者のメールアドレス (`forisek@dcs.fmph.uniba.sk`) まで．

You are welcome to send any feedback on the Syllabus to the current maintainer's e-mail address (`forisek@dcs.fmph.uniba.sk`).

シラバスの改良に協力することに興味のある人は，`http://ksp.sk/~misof/ioi-syllabus/` を参照．このページからは，シラバスの背景に関する追加の情報やその他の情報が入手できる．

For people interested in contributing to the quality of the Syllabus, some additional background on the Syllabus and other miscellaneous information can be found at `http://ksp.sk/~misof/ioi-syllabus/`.

---

* (訳者注) IOI シラバスの公式版は英語で書かれている．この日本語版 (ベータ版) は情報オリンピック日本委員会が独自に作成したものである．

[*1] (訳者注) 2014 年 11 月 1 日時点において，IOI 2013 のシラバスが最新版である．IOI 2014 では IOI 2013 のシラバスがそのまま用いられた．

[*2] TU Eindhoven, The Netherlands, `t.verhoeff@tue.nl`

[*3] University of Szeged, Hungary, `horvath@inf.u-szeged.hu`

[*4] Warsaw University, Poland, `diks@mimuw.edu.pl`

[*5] University of Waterloo, Canada, `gvcormac@uwaterloo.ca`

[*6] Comenius University, Slovakia, `forisek@dcs.fmph.uniba.sk`

3 はじめに (Introduction)

このシラバスの主な目的は，次の 2 つである．
This Syllabus has two main purposes.

1 つ目は，競技課題が国際情報オリンピック (International Olympiad in Informatics, IOI) に適切なものであるかを決定する際の指針を提供することである．競技課題を選ぶ際には，国際科学委員会 (International Scientific Committee, ISC) は，この文書に基づき提案された課題を評価する．
The first purpose is to provide a set of guidelines that help decide whether a task is suitable for the International Olympiad in Informatics (IOI). Based on this document, the International Scientific Committee (ISC) evaluates the task proposals when selecting the competition tasks.

2 つ目は，1 つ目と関連したことだが，各地域のオリンピックの主催者がその地域の生徒を IOI に向けて準備させる際の手助けとなることである．
The second and closely related purpose is to help the organizers of national olympiads prepare their students for the IOI.

これらの目的のために，このシラバスでは数学や計算機科学の項目・概念を分類する．易しい話題のうちのいくつかは「**含まれる** (*included*)」に分類される．一方，高度な話題や関係の無い話題のうちのいくつかは「**明確に除外される** (*explicitly excluded*)」に分類される．より正確に言うと，このシラバスでは項目を次の 5 つの種類のいずれかに分類する:
The Syllabus aims to achieve these goals by providing a classification of topics and concepts from mathematics and computer science. In particular, some of the easy topics are classified as *included*; and on the other end, some hard topics and some unrelated topics are *explicitly excluded*. More precisely, this Syllabus classifies each topic into one of five categories:

♡ **含まれる (制限無し)** (Included, unlimited)
: これらの項目は予備知識とみなされる．競技参加者には，これらを知っていることが期待される．これらの項目は補足説明を伴わずに課題の記述の中に現れることがある．
Topics in this category are considered to be prerequisite knowledge. Contestants are expected to know them. These topics can appear in task descriptions without further clarification.
例 (Example): **整数** *(Integer)* in §4.1

△ **含まれる (課題の記述中で明確にすべき)** (Included, to be clarified)
: 競技参加者はこれらの項目を知っているべきである．しかし，これらが課題の記述の中に現れるときは，著者は十分に明確にしなければならない．
Contestants should know this topic, but when it appears in a task description, the author must always clarify it sufficiently.
例 (Example): **有向グラフ** *(Directed graph)* in §4.2 DS2

⊖ **含まれる (課題の記述中には現れない)** (Included, not for task description)
: これらの項目は課題の記述の中に現れるべきではない．しかし，解答プログラムを作成するときや，モデル解法を理解する際に，これらの項目の知識が必要なこともある．
Topics that belong to this category should not appear in tasks descriptions. However, developing solutions and understanding model solutions may require the knowledge of these topics.
例 (Example): **計算量上界の漸近解析** *(Asymptotic analysis of upper complexity bounds)* in §5.2 AL1

焦点の外である (Outside of focus)
: これは**デフォルトの分類**である（これはシラバスの以前のバージョンとは異なる）．このシラバスの中

に明確に述べられていない項目は，全てこの種類に分類される．
As opposed to previous versions of the Syllabus, this is now the **default category**. Any topic that is not explicitly addressed by the Syllabus should be considered to belong to this category.

競技参加者がこの項目に関する知識を持っていることは期待されていない．ほとんどの競技課題は，この分類に属すいかなる項目とも関係無い．
Contestants are not expected to have knowledge of these topics. Most competition tasks will not be related to any topics from this category.

しかし，これは，この分類に属す項目と関係のある競技課題を避けるべきであるという意味では無い．IOI の活動範囲を広げるために，そのような競技課題が ISC により望まれることもあるかもしれない．
However, note that this is not intended to prevent the inclusion of a competition task that is related to a particular topic from this category. The ISC may wish to include such a competition task in order to broaden the scope of the IOI.

もし，そのような競技課題が IOI で検討されるときは，ISC は，課題が特定の項目に関する予備知識が無くてもある程度は解けること，そして，課題が ♡ ・△ の概念を用いて正確・簡潔・明快に述べられることを確かめる．
If such a task is considered for the IOI, the ISC will make sure that the task can reasonably be solved without prior knowledge of the particular topic, and that the task can be stated in terms of ♡ and △ concepts in a precise, concise, and clear way.

最近の IOI に出題された例としては，IOI 2010 カナダ大会の Languages (別名: Wikipedia) や，IOI 2012 イタリア大会の Odometer (別名: robot with pebbles) がある．
Examples of such tasks being used at recent IOIs include Languages (a.k.a. Wikipedia) from IOI 2010 in Canada, and Odometer (a.k.a. robot with pebbles) from IOI 2012 in Italy.

明確に除外される (Explicitly excluded)

難しいアルゴリズムの項目のうちのいくつかは明確に除外される．競技参加者には，これらの分野の**知識が要求される**課題は出題されないことが保証される．すなわち，全ての競技課題には，これらの項目の知識が無くても作成可能な満点解法が存在する．ここには，主に，難しい教科書のアルゴリズムが含まれる．
Some of the harder algorithmic topics are explicitly marked as excluded. It is guaranteed that there will not be a competition task that *requires* the contestants to know these areas. In other words, each competition task will have a perfect solution that can be produced without the knowledge of these topics. This category mainly contains hard textbook algorithms.
ただし，このシラバスが，競技参加者が課題に取り組む際に適用可能な技術を，いかなる意味においても制限していると解釈すべきではない．
Still, note that the Syllabus must not be interpreted to restrict in any way the techniques that contestants are allowed to apply in solving the competition tasks.
ここには，IOI の範囲から明らかに逸脱した項目も含まれる．
Additionally, this category is used for topics that clearly fall outside the scope of the IOI.
例 (Examples): **最大流アルゴリズム** *(Maximum flow algorithms)* in §5.2 AL3 / **微積分** *(Calculus)* in §4.3

これらのいずれの種類についても，全てをもれなく列挙することは明らかに不可能である．このシラバスに挙げるリストは，境界線上の例を示したものである．「**含まれる** *(Included)*」に分類された項目よりも易しいか同程度の項目は，「含まれる」と解釈せよ．また，「**明確に除外される** *(Explicitly excluded)*」に分類された項目よりも難しいか同程度の項目は，「明確に除外される」と解釈せよ．

Obviously, none of the categories can ever be exhaustively enumerated. Instead, the list given in the following Sections should serve as examples that map out the boundary: anything easier or similar to *Included* topics is to be considered Included as well, and anything similar or harder than the *Explicitly excluded* topics is Excluded, too.

もし，分類が不明確な項目がある場合は，明確にすべきであるという依頼を，現在のシラバスの管理者に送ることを勧める．

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.

競技課題に用いられる用語・表記法の適切性に関する話題は，本シラバスの範囲外である（T. Verhoeff: *Concepts, Terminology, and Notations for IOI Competition Tasks*, `http://scienceolympiads.org/ioi/sc/documents/terminology.pdf` を参照）．

Note that issues related to the usage of suitable terminology and notations in competition tasks are beyond the scope of this document.[*7]

以下の節に項目の分類を与える．IEEE-CS カリキュラムから引用した項目はサンセリフ体 (sans serif font) で表す（ACM/IEEE-CS Joint Curriculum Task Force: *Computing Curricula 2001: Computer Science Volume*,`http://www.acm.org/sigcse/cc2001/` を参照）．

The rest of this document contains the classification of topics. Topics literally copied from the IEEE-CS Curriculum[*8] are typeset in sans serif font.

4 数学 (Mathematics)

4.1 算術および幾何 (Arithmetics and Geometry)

- ♡ 整数 (integers), 演算 (operations) (指数演算を含む (incl. exponentiation)), 比較 (comparison)
- ♡ 整数の基本性質 (basic properties of integers) (符号 (sign), 偶奇 (parity), 可除性 (divisibility))
- ♡ 素数 (prime numbers), 剰余に関する基本演算 (basic modular arithmetic) — 加法 (addition), 減法 (subtraction), 乗法 (multiplication)
- ♡ 分数 (fractions), 百分率 (percentages)
- ♡ 直線 (line), 線分 (line segment), 角度 (angle), 三角形 (triangle), 四角形 (rectangle), 正方形 (square), 円 (circle)
- ♡ 点 (point), ベクトル (vector), 平面座標 (coordinates in the plane)
- ♡ 多角形 (polygon) (頂点 (vertex), 辺 (side/edge), 単純 (simple), 凸 (convex), 内部 (inside), 領域 (area))
- △ ユークリッド距離 (Euclidean distances)
- ⊖ ピタゴラスの定理 (Pythagorean theorem)

明確に除外される *(Explicitly excluded)*: 剰余に関する除算・逆元 (modular division and inverse elements), 複素数 (complex numbers), 一般の二次曲線 (general conics) (放物線 (parabolas), 双曲線 (hyperbolas), 楕円 (ellipses)), 三角関数 (trigonometric functions)

4.2 離散構造 (Discrete Structures) (DS)

DS1. 関数・関係・集合 (Functions, relations, and sets)

[*7] See T. Verhoeff: *Concepts, Terminology, and Notations for IOI Competition Tasks*, `http://scienceolympiads.org/ioi/sc/documents/terminology.pdf`

[*8] ACM/IEEE-CS Joint Curriculum Task Force: *Computing Curricula 2001: Computer Science Volume*, `http://www.acm.org/sigcse/cc2001/`

△ 関数 (functions) (全射 (surjections), 単射 (injections), 逆写像 (inverses), 合成 (composition))
△ 関係 (relations) (反射律 (reflexivity), 対称律 (symmetry), 推移律 (transitivity), 同値関係 (equivalence relations), 全順序関係・線形順序関係 (total/linear order relations), 辞書式順序 (lexicographic order))
♡ 集合 (sets) (ベン図 (Venn diagrams), 補集合 (complements), 直積 (Cartesian products), ベキ集合 (power sets))

明確に除外される *(Explicitly excluded)*: 濃度・可算性 (cardinality and countability) (無限集合に関するもの (of infinite sets))

DS2. 基礎的な論理 (Basic logic)

♡ 命題論理 (propositional logic)
♡ 論理結合 (logical connectives) (および，それらの基本性質 (incl. their basic properties))
♡ 真偽表 (truth tables)
♡ 一階述語論理 (predicate logic)
♡ 全称記号・存在記号 (universal and existential quantification) (注意: 全称記号・存在記号が入れ子になった定義はできる限り避けるべきである．(Note: statements should avoid definitions with nested quantifiers whenever possible.))
⊖ モーダスポネンス (単純な論証) (modus ponens), モーダストレンス (対偶による証明) (modus tollens)

注意: これらは表記法に関するものではない．過去の競技課題では，∧, ∨, ∀, ∃ などの数学記号を使わずに自然言語で論理が説明されていた．

N.B. This article is not concerned with notation. In past task descriptions, logic has been expressed in natural language rather than mathematical symbols, such as ∧, ∨, ∀, ∃.

焦点の外である *(Out of focus)*: 標準形 (normal forms)

明確に除外される *(Explicitly excluded)*: 妥当性 (validity), 一階述語論理の限界 (limitations of predicate logic)

DS3. 証明の技法 (Proof techniques)

△ 含意・逆・対偶・否定・矛盾などの概念 (notions of implication, converse, inverse, contrapositive, negation, and contradiction)
⊖ 直接の証明 (direct proofs), 反例・対偶・矛盾による証明 (proofs by: counterexample, contraposition, contradiction)
⊖ 数学的帰納法 (mathematical induction)
⊖ 強帰納法 (strong induction) (完全帰納法ともいう (also known as complete induction))
♡ 再帰的定義 (recursive mathematical definitions) (相互の再帰的定義を含む (incl. mutually recursive definitions))

DS4. 数え上げの基礎 (Basics of counting)

♡ 数え上げの議論 (counting arguments) (和・積の規則 (sum and product rule), 算術級数・幾何級数 (arithmetic and geometric progressions), フィボナッチ数 (Fibonacci numbers))
△ 置換・組み合わせ (permutations and combinations) (基本的定義 (basic definitions))
△ 階乗関数 (factorial function), 二項係数 (binomial coefficients)
⊖ 包除原理 (inclusion-exclusion principle)
⊖ 鳩ノ巣原理 (pigeonhole principle)
⊖ パスカルの恒等式 (Pascal's identity), 二項定理 (binomial theorem)

明確に除外される *(Explicitly excluded)*: 帰納的関係を解くこと (Solving of recurrence relations)

DS5. グラフ・木 (Graphs and trees)

△ 木 (Trees) とその基本性質 (and their basic properties), 根付き木 (rooted trees)
△ 無向グラフ (undirected graphs) (次数 (degree), 道 (path), 閉路 (cycle), 連結性 (connectedness), オイラー路・オイラー閉路，ハミルトン路・ハミルトン閉路 (Euler/Hamilton path/cycle), 握手補題 (handshaking lemma))
△ 有向グラフ (directed graphs) (入次数 (in-degree), 出次数 (out-degree), 有向路・有向閉路 (directed path/cycle), オイラー路・オイラー閉路，ハミルトン路・ハミルトン閉路 (Euler/Hamil-

ton path/cycle))
△ 全域木 (spanning trees)
△ 走査戦略 (traversal strategies)
△ 辺や頂点がラベル・重さ・色などで「装飾」されたグラフ ('decorated' graphs with edge/node labels, weights, colors)
△ 多重グラフ (multigraphs), 自己ループを含むグラフ (graphs with self-loops)
△ 二部グラフ (bipartite graphs)

焦点の外である *(Out of focus)*: 平面グラフ (planar graphs), 超グラフ (hypergraphs).

DS6. 離散確率 (Discrete probability)

有限通りの状態における確率（この場合の確率は組合せ論でなる）は**焦点の外である** *(Out of focus)*. より複雑なものは全て**明確に除外される** *(Explicitly excluded)*.

Applications where everything is finite (and thus arguments about probability can be easily turned into combinatorial arguments) are *Out of focus*, everything more complicated is *Explicitly excluded*.

4.3 数学のその他の分野 (Other Areas in Mathematics)

焦点の外である *(Out of focus)*: 3 次元空間の幾何 (geometry in 3D space), 線形方程式系 (systems of linear equations)

明確に除外される *(Explicitly excluded)*: 線形代数 (Linear algebra) (多項式・行列に関する非自明な演算を全て含む (incl. all non-trivial operations on polynomials and matrices)), 微積分 (Calculus), 統計学 (Statistics)

5 計算機科学 (Computing Science)

5.1 プログラミングの原理 (Programming Fundamentals) (PF)

PF1. プログラミング構造の基礎 (Fundamental programming constructs) (抽象計算機に関するもの (for abstract machines))
♡ 高水準プログラミグ言語の基礎的構文と意味論 (Basic syntax and semantics of a higher-level language) (IOI の競技規則で定められた IOI で使用可能なプログラミグ言語の少なくとも 1 つについて (at least one of the specific languages available at an IOI, as announced in the *Competition Rules* for that IOI))
♡ 変数 (variables), 型 (types), 式・代入 (expressions, and assignment)
♡ 単純な入出力 (Simple I/O)
♡ 条件付き反復制御構造 (conditional and iterative control structures)
♡ 関数・パラメータ (functions and parameter passing)
⊖ 構造化された分割 (structured decomposition)

PF2. アルゴリズム・問題解決 (Algorithms and problem-solving)
⊖ 問題解決戦略 (Problem-solving strategies (理解して–計画して–実行して–確認せよ (understand–plan–do–check), 関連事項の分離 (separation of concerns), 一般化 (generalization), 特殊化 (specialization), 場合分け (case distinction), 逆向きに考える (working backwards), etc.)[*9]
⊖ 問題解決過程におけるアルゴリズムの役割 (the role of algorithms in the problem-solving process)
⊖ アルゴリズム実装の戦略 (implementation strategies for algorithms) (§6 SE1 を参照 (also see §6 SE1))
⊖ デバッグ戦略 (debugging strategies) (§6 SE1 を参照 (also see §6 SE3))

[*9] See G. Polya: *How to Solve It: A New Aspect of Mathematical Method*, Princeton Univ. Press, 1948

△ アルゴリズムの概念・特性 (the concept and properties of algorithms) (正しさ (correctness), 効率 (efficiency))

PF3. データ構造の基礎 (Fundamental data structures)

♡ 基本データ型 (primitive types) (ブール型 (boolean), 符号付き・符号無し整数型 (signed/unsigned integer), 文字型 (character))
♡ 配列 (arrays) (多重配列を含む (incl. multidimensional arrays))
♡ 構造体 (records)
♡ 文字列・文字列処理 (strings and string processing)
△ 静的割り当て・スタックによる割り当て (static and stack allocation) (自動メモリ管理の初歩 (elementary automatic memory management))
△ 連結構造 (linked structures) (線形・分岐 (linear and branching))
△ 連結構造に関する静的メモリ実装戦略 (Static memory implementation strategies for linked structures)
△ スタック・待ち行列の実装戦略 (implementation strategies for stacks and queues)
△ グラフ・木の実装戦略 (implementation strategies for graphs and trees)
△ 正しいデータ構造を選択する戦略 (strategies for choosing the right data structure)

焦点の外である *(Out of focus)*: メモリ内のデータの表現 (data representation in memory), ヒープ割り当て (heap allocation), 実行時の記憶領域管理 (runtime storage management), ポインタ・参照 (pointers and references)[*10], 数値的に安定な課題における実数の初等的使用 (elementary use of real numbers in numerically stable tasks).

明確に除外される *(Explicitly excluded)*: 実数の浮動小数点型表示 (the floating-point representation of real numbers), 浮動小数点型計算の精度誤差 (precision errors when computing with floating-point numbers)

IOI において，一般に，浮動小数点型を避けるべきである理由がよく知られている（G. Horváth and T. Verhoeff: *Numerical Difficulties in Pre-University Education and Competitions*, Informatics in Education 2:21–38, 2003 を参照）．しかし，現在使用されているインターフェースにより，これらの問題点のいくつかは解決される．今日では，ある種の競技課題（ユークリッド距離を計算し最小値を答える課題など）については，浮動小数点型を安全に用いることができるはずである．

Regarding floating point numbers, there are well-known reasons why they should be, in general, avoided at the IOI.[*11] However, the currently used interface removes some of those issues. In particular, it should now be safe to use floating point numbers in some types of tasks – e.g., to compute some Euclidean distances and return the smallest one.

PF4. 再帰 (Recursion)

♡ 再帰の概念 (the concept of recursion)
♡ 再帰的な数学的関数 (recursive mathematical functions)
♡ 単純な再帰プロシージャ (simple recursive procedures) (相互の再帰を含む (incl. mutual recursion))
⊖ 分割統治法 (divide-and-conquer strategies)
⊖ 再帰の実装 (implementation of recursion)
⊖ 再帰的バックトラック (recursive backtracking)

PF5. イベント駆動型プログラミング (Event-driven programming)

応答型環境でのやりとりに関する競技課題が出題されることもある．提供された環境とのやりとりを実装する競技課題は，△ に分類される．

Some competition tasks may involve a dialog with a reactive environment. Implementing such

[*10] 推論の複雑さが拡張性メモリ効率の本質的でない利点を上回る．IOI の競技課題は静的メモリ実装で解けるべきである．(The inessential advantage of scalable memory efficiency is outweighed by the increased complexity in reasoning. Static memory implementations should suffice to solve IOI tasks.)

[*11] See G. Horváth and T. Verhoeff: *Numerical Difficulties in Pre-University Education and Competitions*, Informatics in Education 2:21–38, 2003

an interaction with the provided environment is △.

応答型課題の実装と直接関係の無いものは，全て**焦点の外である** *(Out of focus)*.
Everything not directly related to the implementation of reactive tasks is *Out of focus*

5.2 アルゴリズム・計算量 (Algorithms and Complexity) (AL)

IEEE-CS カリキュラムより引用する:
We quote from the IEEE-CS Curriculum:

> アルゴリズムは情報科学 (computer science) やソフトウェア工学 (software engineering) の基礎である．実世界におけるソフトウェアシステムの性能は，次の 2 点のみに依存する: (1) アルゴリズムの選択，(2) 実装の様々なレイヤーにおける適合性・効率．したがって，アルゴリズムの良い設計は全てのソフトウェアシステムの性能において極めて重要である．また，アルゴリズムの学習により，プログラミング言語・プログラミングの枠組み・計算機のハードウェアやその他の実装の側面に依存しない問題解決技術が得られるだけでなく，問題の本質を見抜く洞察力も得られる．
> Algorithms are fundamental to computer science and software engineering. The real-world performance of any software system depends only on two things: (1) the algorithms chosen and (2) the suitability and efficiency of the various layers of implementation. Good algorithm design is therefore crucial for the performance of all software systems. Moreover, the study of algorithms provides insight into the intrinsic nature of the problem as well as possible solution techniques independent of programming language, programming paradigm, computer hardware, or any other implementation aspect.

AL1. アルゴリズム解析の基礎 (Basic algorithmic analysis)

- △ アルゴリズムの仕様 (algorithm specification), 事前条件 (precondition), 事後条件 (postcondition), 正しさ (correctness), 不変性 (invariants)
- ⊖ 計算量上界の漸近解析 (Asymptotic analysis of upper complexity bounds) (できれば非公式に (informally if possible))
- ⊖ 大きい O 記法 (Big O notation)
- ⊖ 標準的な計算量クラス (Standard complexity classes) (定数 (constant), 対数 (logarithmic), 線形 (linear), $\mathcal{O}(n \log n)$, 二乗 (quadratic), 三乗 (cubic), 指数 (exponential))
- ⊖ アルゴリズムにおける時間計算量・空間計算量のトレードオフ (time and space tradeoffs in algorithms)

焦点の外である *(Out of focus)*: 最良・平均・最悪の場合の振る舞いの違いの特定 (identifying differences among best, average, and worst case behaviors), 小さい o 記法 (little o), Ω 記法・Θ 記法 (Omega, and Theta notation), 性能の経験的測定 (empirical measurements of performance)
明確に除外される *(Explicitly excluded)*: 平均計算量上界の漸近解析 (asymptotic analysis of average complexity bounds), 再帰的関係による再帰的アルゴリズム解析 (using recurrence relations to analyze recursive algorithms)

AL2. アルゴリズム的戦略 (Algorithmic strategies)

- ⊖ 単純なループ設計戦略 (simple loop design strategies)
- ⊖ 力ずくのアルゴリズム (brute-force algorithms) (全数探索 (exhaustive search))
- ⊖ 貪欲法 (greedy algorithms)
- ⊖ 分割統治法 (divide-and-conquer)
- ⊖ バックトラック法 (backtracking) (再帰的・非再帰的なもの (recursive and non-recursive)), 分枝限定法 (branch-and-bound)
- ⊖ パターンマッチング，文字列・テキストアルゴリズム (pattern matching and string/text algorithms)[*12]

[*12] 基本的なアルゴリズムのみが ⊖ に分類される．例えば，Knuth-Morris-Pratt アルゴリズムは**焦点の外である** *(Out of focus)*.
(Only basic algorithms are ⊖: for instance, the Knuth-Morris-Pratt algorithm should already be considered *Out of*

- ⊖ 動的計画法 (dynamic programming)[*13]

焦点の外である *(Out of focus)*: 発見的解法 (heuristics), 離散近似アルゴリズム (discrete approximation algorithms), 乱択アルゴリズム (randomized algorithms)

明確に除外される *(Explicitly excluded)*: 数値近似アルゴリズム (numerical approximation algorithms)

AL3a. アルゴリズム (Algorithms)

- ⊖ 整数に対する単純な数論アルゴリズム (simple number theory algorithms involving integers): 基数変換 (radix conversion), ユークリッドのアルゴリズム (Euclid's algorithm), $\mathcal{O}(\sqrt{n})$ 回の除法の試行による素数判定 (primality test by $\mathcal{O}(\sqrt{n})$ trial division), エラトステネスの篩 (Sieve of Eratosthenes), 素因数分解 (除法の試行または篩) (factorization (by trial division or a sieve)), 効率の良いベキ乗 (efficient exponentiation)
- ⊖ 任意精度の単純な整数演算 (simple operations on arbitrary precision integers) (加法・減法および単純な乗法 (addition, subtraction, simple multiplication))[*14]
- ⊖ 単純な配列操作 (simple array manipulation) (充填 (filling), 移動 (shifting), 回転 (rotating), 反転 (reversal), サイズ変更 (resizing), 最小値・最大値 (minimum/maximum), 接頭部和 (prefix sums), ヒストグラム (histogram), バケットソート (bucket sort))
- ⊖ 逐次 (sequential) 処理・探索 (processing/search), 二分探索 (and binary search)
- ⊖ 二次ソートアルゴリズム (Quadratic sorting algorithms) (選択 (selection), 挿入 (insertion))
- ⊖ ピボットによる配列分割 (Partitioning an array according to a pivot), クイックソート (Quicksort), クイックセレクト (Quickselect) (k 番目に小さい元の探索 (to find the k-th smallest element))
- ⊖ 最悪計算量が $\mathcal{O}(n \log n)$ となる ($\mathcal{O}(n \log n)$ worst-case) ソートアルゴリズム (sorting algorithms) (ヒープソート (heap sort), マージソート (merge sort))
- ⊖ 有向木の走査 (traversals of ordered trees) (前順・間順・後順 (pre-, in-, and post-order))
- ⊖ グラフの深さ優先探索・幅優先探索 (Depth- and breadth-first traversals of graphs), 無向グラフの連結成分の決定 (determining connected components of an undirected graph)
- ⊖ 最短路アルゴリズム (shortest-path algorithms) (ダイクストラ法 (Dijkstra), ベルマン-フォード法 (Bellman-Ford), フロイド-ワーシャル法 (Floyd-Warshall))
- ⊖ 推移閉包 (transitive closure) (フロイドのアルゴリズム (Floyd's algorithm))
- ⊖ 最小全域木 (minimum spanning tree) (ヤルニーク-プリム法, クラスカル法 (Jarník-Prim and Kruskal algorithms))
- ⊖ トポロジカルソート (topological sort)
- ⊖ オイラー路・オイラー閉路（の存在）を判定するアルゴリズム (algorithms to determine the (existence of an) Euler path/cycle)

明確に除外される *(Explicitly excluded)*: 最大流アルゴリズム (maximum flow algorithms), 二部グラフマッチングアルゴリズム (bipartite matching algorithms), 有向グラフの強連結成分分解 (strongly connected components in directed graphs)

AL3b. データ構造 (Data structures)

- ⊖ 二分ヒープデータ構造 (binary heap data structure)
- ⊖ 二分探索木 (Binary search trees) (バランスの取れていないもの (unbalanced))
- ⊖ 抽象データ型との関係 (interaction with abstract data types): 優先度付きキュー (priority queues); 順序付き集合・順序無し集合・写像 (ordered and unordered sets and maps)

---

focus.)

[*13] IEEE-CS カリキュラムではこの項目を AL8 に分類しているが, ここに分類されるべきであろう．(The IEEE-CS Curriculum puts this under AL8, but we believe it belongs here.)

[*14] これらの演算の実装が必要なことは, 課題の記述から明らかにするべきである．(The necessity to implement these operations should be obvious from the problem statement.)

- ⊖ 区間木 (interval trees)[*15,*16], フェンウィック木 (and Fenwick trees)[*17]
- ⊖ グラフの表現 (representations of graphs) (隣接リスト (adjacency lists), 隣接行列 (adjacency matrix))
- ⊖ 素集合の表現 (representation of disjoint sets): Union-Find データ構造 (the Union-Find data structure)

明確に除外される *(Explicitly excluded)*: 複雑なヒープ構造 (二項ヒープ・フィボナッチヒープなど) (Complex heap variants such as binomial and Fibonacci heaps)，一般の平衡木の実装 (implementation of any general balanced tree) (AVL 木，ツリープ，スプレー木など (e.g., AVL, treaps, splay trees)), ハッシュテーブルによる実装 (Using and implementing hash tables) (衝突の解消戦略を含む (incl. strategies to resolve collisions))

AL4. 分散アルゴリズム (Distributed algorithms)

焦点の外である *(Out of focus)*

AL5. 計算可能性の基礎 (Basic computability)

計算可能性に関連した項目は全て明確に除外される *(Explicitly excluded)*.
All topics related to computability are *Explicitly excluded*

例えば次のものが挙げられる (This includes the following): 易しい問題・難しい問題 (tractable and intractable problems), 計算不能問題 (uncomputable functions), 停止問題 (the halting problem), 計算不能性の帰結 (implications of uncomputability)

計算モデルの基礎については，AL7 を参照.
However, see AL7 for basic computational models.

AL6. 計算複雑性クラス P & NP (The complexity classes P and NP)

非決定性，NP 困難性の証明 (帰着) やこれらに関連した全ての項目は明確に除外される *(Explicitly excluded)*.
Topics related to non-determinism, proofs of NP-hardness (reductions), and everything related is *Explicitly excluded*.

ここでの記述は，学部・大学院レベルの形式言語・計算複雑性の講義で扱われる内容に関するものである．これらの項目が「明確に除外される *(Explicitly excluded)*」からといって，NP 困難な問題が IOI に出題できないというわけではない.
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 *Explicitly excluded* does not mean that an NP-hard problem cannot appear at an IOI.

AL7. オートマトン・文法 (Automata and grammars)

- △ バッカス-ナウア記法による単純な文法の理解 (Understanding a simple grammar in Backus-Naur form)

[*15] 配列から構成される完全二分木を含む (クエリをサポートする．各要素と区間の更新を対数時間で行う)．このデータ構造はセグメント木 (segment tree)，領域木 (range tree)，トーナメント木 (tournament tree) とも呼ばれる．(This is the data structure that contains a complete binary tree built on top of an array; supporting queries and updates on individual elements and intervals in logarithmic time. Sometimes this data structure is called a segment tree, a range tree, or a tournament tree.)

[*16] 計算幾何の分野では，似た名称の別のデータ構造も用いられているので注意せよ (それらはこの項目の対象外である)．(Note that in computational geometry there are different data structures with similar names. Those are not covered by this item.)

[*17] フェンウィック氏の論文 *A New Data Structure for Cumulative Frequency Tables* で初めて導入された．BIT (binary indexed tree) ともいう．フェンウィック木の 2 次元版は IOI 2001 の課題 Mobiles で用いられた．メモリ使用効率の良い区間木の 1 つのバージョンとみなすこともできる．(First introduced in P. Fenwick: *A New Data Structure for Cumulative Frequency Tables*, Software – Practice And Experience 24(3):327–336, 1994. Also known as binary indexed trees. A 2D version of a Fenwick tree was used in the IOI 2001 task Mobiles. This can be seen as a more space-efficient version of an interval tree.)

焦点の外である *(Out of focus)*: 有限状態機械 (finite-state machines), 文脈自由文法 (context-free grammars), 関連した文字列書き換え系 (and related rewriting systems), 正規表現 (regular expressions)

明確に除外される *(Explicitly excluded)*: 教科書に書かれているような証明・構成法．例えば，言語の交わりに関するオートマトンの積構成，NFA から DFA への変換，DFA 最小化，文法の標準形など．All textbook proofs and constructions – e.g., the product construction of an automaton for the intersection of languages, the conversion of a NFA to a DFA, DFA minimization, grammar normal forms.

AL8. 高度なアルゴリズム解析 (Advanced algorithmic analysis)

- ⊖ 組み合わせゲーム理論の基礎 (basics of combinatorial game theory), 勝利・敗北ポジション (winning and losing positions), 最適ゲームのミニマックスアルゴリズム (minimax algorithm for optimal game playing)

焦点の外である *(Out of focus)*: オンラインアルゴリズム (online algorithms), 組み合わせ最適化 (combinatorial optimization), 乱択アルゴリズム (randomized algorithms)

明確に除外される *(Explicitly excluded)*: ならし解析 (amortized analysis), アルファ-ベータ法 (alpha-beta pruning), スプレイグ-グランディ理論 (Sprague-Grundy theory)

AL9. 暗号アルゴリズム (Cryptographic algorithms)

焦点の外である *(Out of focus)*

AL10. 幾何アルゴリズム (Geometric algorithms) (2 次元の場合．整数座標が望ましい (in two dimensions, preferably with integer coordinates))

- ⊖ 点の表現 (representing points), ベクトル (vectors), 直線 (lines), 線分 (line segments).
- ⊖ 同一直線上の点の判定 (checking for colinear points), 平行・直交ベクトル (parallel/orthogonal vectors), 時計回りの回転 (clockwise turns).
- ⊖ 2 直線の交点 (intersection of two lines).
- ⊖ 多角形 (polygons): 面積および重心の計算 (computing area and center of mass).
- ⊖ （一般の・凸）多角形がある点を含むことの判定 (checking whether a (general/convex) polygon contains a point).
- ⊖ 座標圧縮 (coordinates compression).
- ⊖ 凸包計算アルゴリズム (convex hull finding algorithms)
- ⊖ 平面走査法 (sweeping line method)

AL11. 並列アルゴリズム (Parallel algorithms)

焦点の外である *(Out of focus)*

5.3 計算機科学のその他の分野 (Other Areas in Computing Science)

(以下に述べる) GV を除き，これらの項目は全て明確に除外される *(Explicitly excluded)*.
Except for GV (specified below), all areas are *Explicitly excluded*.

AR. 基本設計概念・組織 (Architecture and Organization)

OS. 基本ソフト (Operating Systems)

NC. ネットワークを中心とした計算 (Net-Centric Computing) (別名: クラウドコンピューティング (a.k.a. cloud computing))

PL. プログラミング言語 (Programming Languages)

HC. 人間と計算機の相互作用 (Human-Computer Interaction)

GV. グラフィック・視覚計算 (Graphics and Visual Computing)

画像データ処理の基本的側面は焦点の外である *(Out of focus)*．それ以外のものは全て (OpenGL のようなグラフィックライブラリの使用も含めて) 明確に除外される *(Explicitly excluded)*.
Basic aspects of processing graphical data are *Out of focus*, everything else (including the use of

graphics libraries such as OpenGL) is *Explicitly excluded*.

IS. 知的システム (Intelligent Systems)

IM. 情報の管理 (Information Management)

SP. 社会的・職業的な事柄 (Social and Professional Issues)

CN. 計算科学 (Computational Science)

注意: AR はデジタル系統・アセンブリ言語・命令パイプライン・キャッシュメモリなどに関するものである．OS は基本ソフトの設計 *(design)* に関するものであり，その使用に関するものではない．PL はプログラミング言語の解析・設計 *(analysis and design)* に関するものであり，その使用に関するものではない．HC はユーザインタフェースの設計 *(design)* に関するものである．

Notes: AR is about digital systems, assembly language, instruction pipelining, cache memories, etc. OS is about the *design* of operating systems, not their usage. PL is about the *analysis and design* of programming languages, not their usage. HC is about the *design* of user interfaces.

OS の使用法 *(Usage)*・GUI・プログラミング言語は，§7 および §5.1 を参照．

Usage of the operating system, GUIs and programming languages is covered in §7 and §5.1.

6 ソフトウェア工学 (Software Engineering) (SE)

IEEE-CS カリキュラムから引用する:

We quote from the IEEE-CS Curriculum:

> ソフトウェア工学は，ユーザや顧客の要求をみたすソフトウェアシステムを効果的・効率的に組み立てる理論・知識・練習の応用に関する訓練である．
>
> Software engineering is the discipline concerned with the application of theory, knowledge, and practice for effectively and efficiently building software systems that satisfy the requirements of users and customers.

IOI の競技におけるソフトウェア工学の応用は，時間的な制約の中で，単独の開発者による小さい，単発のプロジェクトのために，軽い技術を用いることに関するものである．これに含まれる項目は全て $\ominus$ に分類される．

In the IOI competition, the application of software engineering concerns the use of light-weight techniques for small, one-off, single-developer projects under time pressure. All included topics are $\ominus$.

SE1. ソフトウェア設計 (Software design)

- $\ominus$ 基礎設計の概念・原理 (fundamental design concepts and principles)
- $\ominus$ 設計様式 (design patterns)
- $\ominus$ 組織化された設計 (structured design)

特に，競技参加者には次のことが期待される．

In particular, contestants may be expected to

- 使用が許されたプログラミング言語の 1 つを用いて，抽象的なアルゴリズムを具体的で効率の良いプログラムに変換すること（標準ライブラリや競技用の特定のライブラリを用いることもある）．

  Transform an abstract algorithm into a concrete, efficient program expressed in one of the allowed programming languages, possibly using standard or competition-specific libraries.
- 所定のフォーマットに従いテキストファイルから入力データを読み込み出力データを書き出すプログラムを作成すること．

  Make their programs read data from and write data to text files according to a prescribed simple format

(すぐ後の SE2 も参照 (See also SE2 immediately below))

明確に除外される *(Explicitly excluded)*: ソフトウェアアーキテクチャ (software architecture), 再使用のための設計 (design for reuse), オブジェクト指向解析・設計 (object-oriented analysis and design), コンポーネントレベルの設計 (component-level design)

SE2. API の使用 (Using APIs)

⊖ API (アプリケーションプログラミングインターフェース) を用いたプログラミング (API (Application Programming Interface) programming)

特に，競技参加者には次のことが期待される．

In particular, contestants may be expected to

- 仕様が与えられた競技用ライブラリを用いること．
  Use competition-specific libraries according to the provided specification.

明確に除外される *(Explicitly excluded)*: 手本によるプログラミング (programming by example), API 環境におけるデバッグ (debugging in the API environment), クラスブラウザや関連したツール (class browsers and related tools), コンポーネント計算の導入 (introduction to component-based computing)

SE3. ソフトウェア開発ツール・環境 (Software tools and environments)

⊖ プログラミング環境 (programming environments), (IDE (統合開発環境) を含む (incl. IDE (Integrated Development Environment)))

特に，競技参加者には次のことが期待される．

In particular, contestants may be expected to

- 提供されたプログラミングエディタの 1 つを用いて，プログラムを書き，編集すること．
  Write and edit program texts using one of the provided program editors.
- 自分自身のプログラムをコンパイルし，実行すること．
  Compile and execute their own programs.
- 自分自身のプログラムのデバッグを行うこと．
  Debug their own programs.

明確に除外される *(Explicitly excluded)*: 検査ツール (testing tools), 設定管理ルール (configuration management tools), 要求分析・設計のモデル化ツール (requirements analysis and design modeling tools), ツール統合機構 (tool integration mechanisms)

SE4. ソフトウェア生産過程 (Software processes)

⊖ ソフトウェアのライフサイクル・生産過程のモデル (software life-cycle and process models)

特に，競技参加者には次のことが期待される．

In particular, contestants may be expected to

- 解答プログラムの開発過程の様々な段階を理解して，適切な方法を選択すること．
  Understand the various phases in the solution development process and select appropriate approaches.

明確に除外される *(Explicitly excluded)*: 開発過程の評価モデル (process assessment models), ソフトウェア生産過程の基準 (software process metrics)

SE5. ソフトウェアの要求・仕様 (Software requirements and specification)

⊖ 実用的要求・実用的でない要求 (functional and nonfunctional requirements)

⊖ 形式仕様技術に関する基本的概念 (basic concepts of formal specification techniques)

特に，競技参加者には次のことが期待される．

In particular, contestants may be expected to

- 要求される効率を理解し，自然言語による正確な記述（数学的に定式化されている場合もそうでない場合もある）を計算モデルによる問題に変換すること．

Transform a precise natural-language description (with or without mathematical formalism) into a problem in terms of a computational model, including an understanding of the efficiency requirements.

明確に除外される *(Explicitly excluded)*: プロトタイプの作成 (prototyping), 要求誘出 (requirements elicitation), 要求分析モデル (requirements analysis modeling techniques)

SE6. ソフトウェア検証 (Software validation)

- ⊖ 基礎の検査 (testing fundamentals), 検証計画の作成やテストケースの生成を含む (including test plan creation and test case generation)
- ⊖ ブラックボックステスト・ホワイトボックステスト (black-box and white-box testing techniques)
- ⊖ ユニット (unit), 統合 (integration), 検証 (validation), システム検査 (and system testing)
- ⊖ 検査 (inspections)

特に，競技参加者には次のことが期待される．

In particular, contestants may be expected to

- よくある誤りを検出する機会を最大化する技術（構造化されたコードを通す，コードの点検，組み込みテスト，テスト実行）を適用すること．
  Apply techniques that maximize the the opportunity to detect common errors (e.g. through well-structured code, code review, built-in tests, test execution).
- 自分自身のプログラム（の一部）を検査すること．
  Test (parts of) their own programs.

明確に除外される *(Explicitly excluded)*: 検証計画 (validation planning), オブジェクト指向検査 (object-oriented testing)

SE7. ソフトウェア進化 (Software evolution)

明確に除外される *(Explicitly excluded)*: ソフトウェア維持管理 (software maintenance), 維持管理可能ソフトウェアの特徴 (characteristics of maintainable software), リエンジニアリング (re-engineering), レガシーシステム (legacy systems), ソフトウェアの再利用 (software reuse)

SE8. ソフトウェアプロジェクト管理 (Software project management)

- ⊖ プロジェクトのスケジュール管理 (project scheduling) (特に時間の管理 (especially time management))
- ⊖ リスク分析 (risk analysis)
- ⊖ ソフトウェア構造の管理 (software configuration management)

特に，競技参加者には次のことが期待される．

In particular, contestants may be expected to

- 様々な活動に費やされる時間を管理すること．
  Manage time spent on various activities.
- 複数の方法がある場合に，リスクを比較検討すること．
  Weigh risks when choosing between alternative approaches.
- 解答プログラム作成中に，様々なバージョンとその状態を把握すること．
  Keep track of various versions and their status while developing solutions.

明確に除外される *(Explicitly excluded)*: ソフトウェア品質保証 (software quality assurance), チーム管理 (team management), ソフトウェア測定・推定技術 (software measurement and estimation techniques), プロジェクト管理ツール (project management tools)

SE9. コンポーネント計算 (Component-based computing)

明確に除外される *(Explicitly excluded)*

SE10. 形式手法 (Formal methods)

形式手法の概念 (Formal methods concepts) (正しさの証明の概念 (notion of correctness proof), 不変性 (invariant))

前アサーション・後アサーション (Pre and post assertions)

特に，競技参加者には次のことが期待される．
In particular, contestants may be expected to

- アルゴリズムやプログラムの正しさ・効率性について論理的に考えること．
  Reason about the correctness and efficiency of algorithms and programs.

明確に除外される *(Explicitly excluded)*: 形式的検証 (formal verification), 形式仕様記述言語 (formal specification languages), 実行可能あるいは実行可能でない仕様記述 (executable and non-executable specifications)

SE11. ソフトウェア信頼性 (Software reliability)
明確に除外される *(Explicitly excluded)*

SE12. 特殊なシステムの開発 (Specialized systems development)
明確に除外される *(Explicitly excluded)*

7 計算機リテラシー (Computer Literacy)

この節の内容は $\ominus$ である．
The text of this section is $\ominus$.

競技参加者は，コンピュータの基礎的な構造や操作についての知識を持ち，理解していなければならない (CPU, メモリ, I/O など)．競技参加者は，グラフィカルなユーザインターフェースを持つ標準的なコンピュータや，補助アプリケーション付き基本ソフト (OS)，競技用に提供された開発環境を使用できることが期待される．特に，ファイル操作の技能は役に立つ (フォルダの作成，ファイルのコピー，移動など)．

Contestants should know and understand the basic structure and operation of a computer (CPU, memory, I/O). They are expected to be able to use a standard computer with graphical user interface, its operating system with supporting applications, and the provided program development tools for the purpose of solving the competition tasks. In particular, some skill in file management is helpful (creating folders, copying and moving files).

設備の詳細は IOI の**競技規則** *(Competition Rules)* に記述される．通常，いくつかのサービスは標準的なウェブブラウザ上で利用可能である．競技用ツール (個別の説明書の付属したもの) が提供されることがあるかもしれない．

Details of these facilities will be stated in the *Competition Rules* of the particular IOI. Typically, some services are available through a standard web browser. Possibly, some competition-specific tools are made available, with separate documentation.

同程度の機能を持つ複数のツールが利用可能なことがよくある．競技参加者には，全てのツールの特徴を知っていることは期待されていない．競技参加者は，自分自身が最適だと思うものを選び使用すればよい．

It is often the case that a number of equivalent tools are made available. The contestants are not expected to know all the features of all these tools. They can make their own choice based on what they find most appropriate.

焦点の外である *(Out of focus)*: 電卓 (calculator), ワードプロセッサ (word-processors), 表計算ソフトウェア (spreadsheet applications), データベース管理システム (database management systems), 電子メールクライアント (e-mail clients), グラフィックツール (graphics tools) (描画 (drawing), 塗装 (painting))