Learning Breakdown Version 2

Dưới đây là phiên bản tổng hợp duy nhất, tích hợp tất cả thông tin từ các phiên bản bạn cung cấp về 14 môn học nền tảng trong Lý thuyết Khoa học Máy tính (Theoretical Computer Science - TCS). Nội dung được tổ chức rõ ràng, không bỏ sót bất kỳ thông tin nào, nhưng các phần lặp lại đã được gộp hoặc tối ưu để tránh trùng lặp. Mỗi môn học được trình bày với các hạng mục: **Khái niệm cốt lõi**, **Bài toán trọng tâm**, **Câu hỏi lớn**, **Ứng dụng trong TCS**, và **Bài toán kinh điển**, cùng các bảng tóm tắt và giải thích chi tiết.

Tổng hợp 14 môn học nền tảng trong Lý thuyết Khoa học Máy tính

1. Đại số trừu tượng (Abstract Algebra)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Nhóm (Groups)**: Cấu trúc với phép toán kết hợp, phần tử đơn vị, và nghịch đảo (VD: số nguyên với phép cộng, nhóm hoán vị S_n). - **Vành (Rings)**: Cấu trúc với hai phép toán (cộng, nhân) thỏa mãn tính giao hoán và phân phối (VD: số nguyên Z). - **Trường (Fields)**: Vành mà mọi phần tử khác 0 có nghịch đảo nhân (VD: số thực R, số phức C, trường hữu hạn F_p). - **Đồng cấu (Homomorphisms)**: Hàm bảo toàn cấu trúc giữa các đại số (nhóm, vành, trường). - **Lý thuyết Galois**: Liên kết trường mở rộng với nhóm đối xứng để nghiên cứu tính giải được của đa thức.
Bài toán trọng tâm	- Phân loại nhóm, vành, trường dựa trên tính chất (cyclic, Abel, không Abel). - Tìm đồng cấu và đẳng cấu giữa các cấu trúc. - Giải bài toán đối xứng (nhóm hoán vị, đối xứng đa giác). - Ứng dụng trong mật mã (RSA, mã dựa trên trường hữu hạn).
Câu hỏi lớn	- Làm sao mô tả và phân loại cấu trúc đại số tổng quát? - Cấu trúc đại số giải quyết bài toán tính toán (mã hóa, tối ưu) thế nào? - Ánh xạ giữa cấu trúc đại số giúp hiểu hệ thống phức tạp (TCS, vật lý lý thuyết) ra sao?
Ứng dụng trong TCS	- **Mật mã học**: Nhóm cyclic và trường hữu hạn trong RSA, Diffie-Hellman. - **Lý thuyết mã hóa**: Vành và trường cho mã sửa lỗi (Hamming, Reed-Solomon). - **Thiết kế thuật toán**: Nhóm trong thuật toán đồ thị, tối ưu hóa (Graph Isomorphism, phân tích đối xứng).
Bài toán kinh điển	- **Phân loại nhóm hữu hạn**: Liệt kê nhóm hữu hạn (cyclic, Abel, không Abel), dẫn đến bảng tuần hoàn nhóm đơn. - **Tìm đồng cấu nhóm**: Tìm đồng cấu từ nhóm G sang H. - **Bài toán từ đẳng cấu (Word Problem)**: Xác định liệu một từ trong nhóm có là phần tử đơn vị (không quyết định được trong một số trường hợp). - **Nhóm con Sylow**: Tìm nhóm con Sylow để phân tích cấu trúc nhóm hữu hạn. - **Phân loại nhóm Abel**: Phân loại nhóm Abel hữu hạn thành tổng trực tiếp nhóm cyclic. - **Bài toán Galois**: Nghiên cứu tính giải được của đa thức qua nhóm đối xứng.

Giải thích chi tiết bài toán kinh điển:

- **Phân loại nhóm hữu hạn**: Bài toán liệt kê và phân loại tất cả nhóm hữu hạn theo bậc (số phần tử), ví dụ nhóm cyclic Z_n hoặc nhóm dihedral D_n. Đây là thành tựu lớn thế kỷ 20, với bảng phân loại nhóm đơn là nền tảng. Ứng dụng trong mật mã học (nhóm cyclic trong RSA) và phân tích đối xứng đồ thị.
- **Tìm đồng cấu nhóm**: Yêu cầu tìm ánh xạ bảo toàn cấu trúc giữa hai nhóm, ví dụ từ Z_4 sang S_3. Liên quan đến biểu diễn dữ liệu và thuật toán (Graph Isomorphism).

- **Bài toán từ đẳng cấu**: Xác định liệu một từ (chuỗi các sinh trong nhóm) có biểu diễn phần tử đơn vị hay không. Không quyết định được trong một số nhóm tự do, liên quan đến kiểm chứng hình thức và mật mã học.
- **Nhóm con Sylow**: Tìm các nhóm con có bậc là lũy thừa của số nguyên tố trong nhóm hữu hạn, giúp phân tích cấu trúc nhóm. Ứng dụng trong phân tích thuật toán và mật mã.
- **Phân loại nhóm Abel**: Mọi nhóm Abel hữu hạn là tổng trực tiếp của các nhóm cyclic, ví dụ $Z_6 = Z_2 \times Z_3$. Liên quan đến biểu diễn dữ liệu và lý thuyết mã hóa.
- **Bài toán Galois**: Liên kết nhóm đối xứng của nghiệm đa thức với trường mở rộng, giúp xác định tính giải được bằng căn. Ứng dụng trong mật mã và lý thuyết tính toán.

---

2. Logic

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Logic mệnh đề**: Mệnh đề đúng/sai với phép nối (AND, OR, NOT). - **Logic vị từ**: Mở rộng với biến và lượng từ (∀, ∃). - **Lý thuyết mô hình**: Quan hệ giữa công thức logic và cấu trúc toán học. - **Lý thuyết chứng minh**: Phân tích tính đúng đắn của chứng minh. - **Tính toán logic**: Logic trong hệ thống tính toán (temporal logic, description logic).
Bài toán trọng tâm	- Xây dựng hệ thống logic để mô tả và kiểm chứng hệ thống tính toán. - Chứng minh/bác bỏ công thức logic. - Kiểm tra tính thỏa mãn (SAT). - Thiết kế công cụ tự động hóa suy luận (trình giải SAT, kiểm chứng mô hình).
Câu hỏi lớn	- Biểu diễn tri thức và lập luận chính xác trong máy tính thế nào? - Công thức logic có thể chứng minh/bác bỏ không? (Vấn đề quyết định) - Tự động hóa suy luận logic trong AI và kiểm chứng phần mềm ra sao?
Ứng dụng trong TCS	- **Kiểm chứng hình thức**: Kiểm tra tính đúng đắn phần cứng/phần mềm. - **AI**: Suy luận, biểu diễn tri thức. - **Ngữ nghĩa ngôn ngữ lập trình**: Định nghĩa ý nghĩa chương trình qua logic.
Bài toán kinh điển	- **Thỏa mãn (SAT)**: Tìm gán giá trị cho công thức CNF để đúng. - **Kiểm chứng mô hình**: Xác định hệ thống thỏa mãn công thức logic. - **Quyết định logic vị từ (Entscheidungsproblem)**: Kiểm tra công thức đúng mọi mô hình (không quyết định được). - **Định lý Gödel**: Hệ tiên đề đủ mạnh có mệnh đề không chứng minh được. - **Bài toán Thompson**: Liên quan đến tính quyết định của hệ thống logic.

Giải thích chi tiết bài toán kinh điển:

- **Thỏa mãn (SAT)**: Xác định liệu công thức logic mệnh đề (CNF) có gán giá trị đúng/sai để trở thành đúng. Là bài toán NP-complete đầu tiên (Cook-Levin), nền tảng cho trình giải SAT (Z3, MiniSAT). Ứng dụng trong kiểm chứng phần mềm, lập kế hoạch, và suy luận AI.
- **Kiểm chứng mô hình**: Kiểm tra liệu hệ thống (mô hình hóa bằng automata) thỏa mãn công thức logic (thường là temporal logic như LTL, CTL). Ứng dụng trong kiểm tra lỗi chip, giao thức mạng, hệ thống nhúng.
- **Quyết định logic vị từ**: Xác định công thức logic vị từ có đúng trong mọi mô hình (Entscheidungsproblem). Turing và Church chứng minh không quyết định được, minh họa giới hạn logic trong TCS.
- **Định lý Gödel**: Trong hệ tiên đề đủ mạnh (như số học Peano), tồn tại mệnh đề đúng nhưng không chứng minh được. Liên quan đến kiểm chứng hình thức và giới hạn tính toán.
- **Bài toán Thompson**: Liên quan đến tính quyết định trong logic và tính quy tắc của hệ thống, thường xuất hiện trong lý thuyết automata và kiểm chứng.

---

3. Toán rời rạc (Discrete Mathematics)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Quan hệ, hàm**: Liên hệ giữa tập hợp, ánh xạ duy nhất. - **Đồ thị, cây**: Mô hình hóa mạng, hệ thống phân cấp. - **Tổ hợp đếm**: Đếm cách sắp xếp/lựa chọn (hoán vị, tổ hợp). - **Lý thuyết tập hợp rời rạc**: Phép toán trên tập hữu hạn. - **Thuật toán cơ bản**: Phân tích, thiết kế thuật toán.
Bài toán trọng tâm	- Phân tích, thiết kế thuật toán trên đồ thị, cây (đường đi ngắn nhất, cây bao trùm). - Đếm số cấu hình/tổ hợp (số cách xếp n vật). - Tối ưu hóa trên cấu trúc rời rạc (luồng tối đa, tô màu). - Mô hình hóa hệ thống tính toán.
Câu hỏi lớn	- Mô hình hóa, giải bài toán thực tế bằng cấu trúc rời rạc thế nào? - Tối ưu hóa thuật toán trên cấu trúc rời rạc đến đâu? - Đếm/liệt kê tất cả cấu hình khả thi trong hệ thống phức tạp ra sao?
Ứng dụng trong TCS	- **Thiết kế thuật toán**: Phân tích độ phức tạp, hiệu suất (Dijkstra, Kruskal). - **Mạng máy tính**: Đồ thị mô hình hóa kết nối. - **Tối ưu hóa**: Tổ hợp trong lập lịch, phân bổ tài nguyên.
Bài toán kinh điển	- **Đường đi ngắn nhất**: Tìm đường đi ngắn nhất trong đồ thị có trọng số (Dijkstra, Bellman-Ford). - **Cây bao trùm tối thiểu**: Tìm cây với tổng trọng số nhỏ nhất (Kruskal, Prim). - **Luồng tối đa**: Tìm luồng lớn nhất trong mạng (Ford-Fulkerson). - **Tô màu bản đồ (Four Color Theorem)**: Tô bản đồ với tối đa 4 màu. - **Hanoi Tower**: Di chuyển đĩa theo quy tắc (đệ quy). - **Sắp xếp topo**: Sắp xếp đỉnh trong DAG theo thứ tự topological.

Giải thích chi tiết bài toán kinh điển:

- **Đường đi ngắn nhất**: Tìm đường đi ngắn nhất từ nguồn đến đích trong đồ thị có trọng số, sử dụng Dijkstra (O(V^2)), Bellman-Ford (O(VE)), hoặc A* (heuristic). Ứng dụng trong định tuyến mạng, GPS.
- **Cây bao trùm tối thiểu**: Tìm cây liên thông với tổng trọng số nhỏ nhất, sử dụng Kruskal hoặc Prim (O(E log V)). Ứng dụng trong thiết kế mạng, phân cụm dữ liệu.
- **Luồng tối đa**: Tìm luồng lớn nhất trong mạng luồng, sử dụng Ford-Fulkerson hoặc Edmonds-Karp (O(VE^2)). Ứng dụng trong tối ưu băng thông, phân phối hàng hóa.
- **Tô màu bản đồ**: Chứng minh mọi bản đồ phẳng có thể tô bằng 4 màu (Four Color Theorem). Liên quan đến tô màu đồ thị, ứng dụng trong lập lịch, phân bổ tần số.
- **Hanoi Tower**: Di chuyển n đĩa từ cột A sang cột C với quy tắc không đặt đĩa lớn lên nhỏ, sử dụng đệ quy. Ứng dụng trong phân tích thuật toán đệ quy.
- **Sắp xếp topo**: Sắp xếp các đỉnh trong đồ thị có hướng không chu trình (DAG) theo thứ tự topological, sử dụng DFS hoặc BFS (O(V+E)). Ứng dụng trong lập lịch, phân tích phụ thuộc.

---

4. Tổ hợp (Combinatorics)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Đếm**: Tính số cách chọn, sắp xếp (hoán vị, tổ hợp). - **Bao hàm-loại trừ**: Đếm tập hợp phức tạp. - **Hàm sinh**: Giải bài toán đếm, phân tích chuỗi.

Hạng mục	Nội dung
	- **Tổ hợp đồ thị**: Tính chất tổ hợp của đồ thị (số vòng, số cây). - **Thiết kế tổ hợp**: Khối, ma trận thỏa mãn ràng buộc.
Bài toán trọng tâm	- Đếm số cách sắp xếp/chọn với ràng buộc. - Phân tích cấu trúc tổ hợp trong đồ thị (số vòng, cây). - Thiết kế cấu trúc tổ hợp (mã hóa, lập lịch). - Giải bài toán tối ưu tổ hợp (ghép đôi, bao phủ).
Câu hỏi lớn	- Có bao nhiêu cách thực hiện nhiệm vụ/cấu hình hệ thống? - Tối ưu hóa cấu trúc tổ hợp trong bài toán thực tế thế nào? - Tổng quát hóa, phân loại cấu trúc tổ hợp ra sao?
Ứng dụng trong TCS	- **Phân tích thuật toán**: Tính số phép toán cần thiết. - **Mật mã học**: Thiết kế mã, giao thức. - **Tối ưu hóa**: Lập kế hoạch, phân bổ tài nguyên.
Bài toán kinh điển	- **Bao phủ tập hợp**: Tìm tập con nhỏ nhất bao phủ tập gốc. - **Đếm hoán vị**: Tính số cách sắp xếp với ràng buộc. - **Thiết kế khối**: Xây dựng khối thỏa mãn ràng buộc. - **Ramsey**: Tìm số lớn nhất để đồ thị chứa cấu trúc con. - **36 sĩ quan Euler**: Không tồn tại bảng Latin bậc 6. - **Xếp domino**: Phủ bảng bằng domino (Fibonacci). - **Các đường chéo đa giác**: Số cách nối đỉnh đa giác (Catalan). - **Chia bài**: Tính số cách chia bộ bài theo quy tắc.

Giải thích chi tiết bài toán kinh điển:

- **Bao phủ tập hợp**: Tìm tập con nhỏ nhất bao phủ tập vũ trụ U, là bài toán NP-complete, giải bằng thuật toán tham lam. Ứng dụng trong tối ưu trạm phát sóng, lập lịch.
- **Đếm hoán vị**: Tính số cách sắp xếp n đối tượng với ràng buộc (VD: A và B không cạnh nhau), dùng bao hàm-loại trừ hoặc hàm sinh. Ứng dụng trong lập lịch, thiết kế thí nghiệm.
- **Thiết kế khối**: Xây dựng khối hoặc ma trận thỏa mãn ràng buộc (VD: mỗi cặp xuất hiện k lần). Ứng dụng trong mã sửa lỗi, thiết kế thí nghiệm.
- **Ramsey**: Tìm số lớn nhất R(m,n) sao cho mọi đồ thị với số đỉnh ≥ R(m,n) chứa K_m hoặc đồ thị bổ không chứa K_n. Ứng dụng trong phân tích mạng, tổ hợp đồ thị.
- **36 sĩ quan Euler**: Euler chứng minh không tồn tại bảng Latin bậc 6 (36 ô, mỗi hàng/cột có 6 giá trị khác nhau). Liên quan đến thiết kế tổ hợp.
- **Xếp domino**: Tính số cách phủ bảng 2×n bằng domino 2×1, liên quan đến số Fibonacci. Ứng dụng trong đếm cấu hình.
- **Các đường chéo đa giác**: Đếm số cách nối đỉnh đa giác lồi bằng đường chéo không cắt nhau, liên quan đến số Catalan. Ứng dụng trong tổ hợp và thuật toán.
- **Chia bài**: Tính số cách chia bộ bài theo quy tắc (VD: mỗi người nhận k lá). Ứng dụng trong lập lịch, phân tích xác suất.

---

5. Lý thuyết tập hợp (Set Theory)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Tập hợp, phần tử**: Phép toán hợp, giao, hiệu, bổ sung. - **Quan hệ, hàm**: Quan hệ là tập con tích Descartes; hàm ánh xạ duy nhất. - **Số đếm, số thứ tự**: Độ lớn ($\aleph_0$, $2^{\aleph_0}$) và cấu trúc thứ tự. - **Tiên đề ZFC**: Tránh nghịch lý Russell. - **Lý thuyết mô hình, giả thuyết Continuum**: Nghiên cứu cấu trúc toán học.

Hạng mục	Nội dung
Bài toán trọng tâm	- Phân loại tập hợp dựa trên số đếm, thứ tự. - Chứng minh tính chất tập hợp, hàm, quan hệ. - Giải bài toán tính đếm được, ánh xạ tập hợp. - Nghiên cứu cấu trúc toán học qua lý thuyết mô hình.
Câu hỏi lớn	- Xây dựng nền tảng toán học không mâu thuẫn thế nào? - Có tập hợp nào giữa $\aleph_0$ và $2^{\aleph_0}$ không? (Giả thuyết Continuum) - Mô tả khái niệm trừu tượng trong tính toán (ngôn ngữ, máy Turing) ra sao?
Ứng dụng trong TCS	- **Ngữ nghĩa ngôn ngữ lập trình**: Định nghĩa kiểu dữ liệu. - **Logic, kiểm chứng hình thức**: Nền tảng logic vị từ. - **Lý thuyết tính toán**: Định nghĩa ngôn ngữ, máy Turing.
Bài toán kinh điển	- **Đếm tập hợp**: So sánh số đếm tập hợp vô hạn. - **Giả thuyết Continuum**: Có tập hợp giữa $\aleph_0$ và $2^{\aleph_0}$ không? - **Ánh xạ song ánh**: Chứng minh hai tập có cùng số đếm. - **Cantor**: So sánh lực lượng tập hợp ($\aleph_0$ vs. $2^{\aleph_0}$). - **Axiom of Choice**: Chọn phần tử từ hệ tập. - **Nghịch lý Russell**: Tập hợp các tập không chứa chính nó.

Giải thích chi tiết bài toán kinh điển:

- **Đếm tập hợp**: So sánh số đếm của tập hợp vô hạn (VD: số tự nhiên $\aleph_0$, số thực $2^{\aleph_0}$). Ứng dụng trong phân tích cấu trúc dữ liệu, ngữ nghĩa lập trình.
- **Giả thuyết Continuum**: Hỏi liệu có tập hợp nào có số đếm giữa $\aleph_0$ và $2^{\aleph_0}$. Gödel và Cohen chứng minh độc lập với ZFC. Liên quan đến logic và lý thuyết mô hình.
- **Ánh xạ song ánh**: Tìm ánh xạ song ánh để chứng minh hai tập có cùng số đếm (VD: số chẵn và số tự nhiên). Ứng dụng trong thiết kế kiểu dữ liệu.
- **Cantor**: Chứng minh tập lũy thừa của tập hợp có số đếm lớn hơn, dẫn đến phân cấp $\aleph_0, \aleph_1, \ldots$ Ứng dụng trong lý thuyết tính toán.
- **Axiom of Choice**: Tiên đề cho phép chọn một phần tử từ mỗi tập trong hệ tập, ảnh hưởng đến lý thuyết tập hợp và logic. Ứng dụng trong kiểm chứng hình thức.
- **Nghịch lý Russell**: Tập hợp các tập không chứa chính nó dẫn đến mâu thuẫn, thúc đẩy hệ tiên đề ZFC. Liên quan đến nền tảng logic trong TCS.

---

6. Xác suất rời rạc (Discrete Probability)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Biến ngẫu nhiên**: Ánh xạ từ không gian mẫu sang số thực. - **Kỳ vọng, phương sai**: Giá trị trung bình, độ phân tán. - **Phân phối rời rạc**: Bernoulli, Binomial, Poisson, Geometric. - **Bất đẳng thức**: Markov, Chebyshev, Chernoff. - **Chuỗi Markov**: Quá trình ngẫu nhiên không nhớ.
Bài toán trọng tâm	- Tính xác suất sự kiện trong không gian mẫu rời rạc. - Phân tích thuật toán ngẫu nhiên (QuickSort, Miller-Rabin). - Mô hình hóa hệ thống ngẫu nhiên (giao thức mạng, Monte Carlo). - Dự đoán hành vi chuỗi Markov.
Câu hỏi lớn	- Dự đoán, kiểm soát hệ thống ngẫu nhiên thế nào? - Thuật toán ngẫu nhiên hiệu quả hơn xác định khi nào?

Hạng mục	Nội dung
	- Phân tích độ tin cậy, hiệu suất hệ thống tính toán ra sao?
Ứng dụng trong TCS	- **Thuật toán ngẫu nhiên**: QuickSort, kiểm tra nguyên tố. - **Học máy lý thuyết**: Mô hình học thống kê. - **Phân tích mạng**: Mô hình lưu lượng, độ trễ bằng chuỗi Markov.
Bài toán kinh điển	- **Phân tích thuật toán ngẫu nhiên**: Phân tích QuickSort, Miller-Rabin. - **Chuỗi Markov**: Tính trạng thái ổn định. - **Chernoff**: Phân tích xác suất sự kiện ngẫu nhiên. - **Sinh nhật**: Xác suất trùng ngày sinh. - **Monty Hall**: Đổi cửa tăng cơ hội thắng. - **Xếp ngẫu nhiên mũ**: Xác suất không ai đội đúng mũ (derangement). - **Sinh con trước**: Xác suất chuỗi xuất hiện trước chuỗi khác.

Giải thích chi tiết bài toán kinh điển:

- **Phân tích thuật toán ngẫu nhiên**: Phân tích hiệu suất kỳ vọng của QuickSort (O(n log n)) hoặc Miller-Rabin (kiểm tra nguyên tố). Ứng dụng trong mật mã, học máy.
- **Chuỗi Markov**: Tính xác suất trạng thái ổn định hoặc thời gian trộn của chuỗi Markov. Ứng dụng trong phân tích mạng, mô phỏng hệ thống, Hidden Markov Models.
- **Chernoff**: Sử dụng bất đẳng thức Chernoff để phân tích sai lệch của biến ngẫu nhiên, giúp đánh giá thuật toán ngẫu nhiên. Ứng dụng trong phân tích hiệu suất mạng.
- **Sinh nhật**: Với 23 người, xác suất có ít nhất 2 người trùng ngày sinh > 50%. Ứng dụng trong phân tích xác suất và mật mã.
- **Monty Hall**: Trong trò chơi 3 cửa, đổi cửa tăng xác suất thắng từ 1/3 lên 2/3. Minh họa xác suất có điều kiện.
- **Xếp ngẫu nhiên mũ**: Xác suất không ai đội đúng mũ trong n người đội ngẫu nhiên (derangement), gần 1/e khi n lớn. Ứng dụng trong phân tích xáo trộn.
- **Sinh con trước**: Tính xác suất một chuỗi xuất hiện trước chuỗi khác trong dãy ngẫu nhiên. Ứng dụng trong phân tích chuỗi và xác suất.

---

7. Lý thuyết đồ thị (Graph Theory)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Đồ thị**: Đỉnh, cạnh (hướng/vô hướng). - **Cây**: Đồ thị không chu trình. - **Đường đi, chu trình**: Đường đi ngắn nhất, chu trình Euler, Hamilton. - **Luồng, ghép đôi**: Tối ưu luồng, ghép đôi tối đa. - **Lý thuyết Ramsey**: Tính chất tổ hợp đồ thị lớn.
Bài toán trọng tâm	- Tìm đường đi ngắn nhất, luồng tối đa. - Xác định liên thông, chu trình, số màu. - Giải bài toán ghép đôi, bao phủ, phân hoạch. - Phân tích tính chất tổ hợp đồ thị.
Câu hỏi lớn	- Mô hình hóa, tối ưu hóa hệ thống phức tạp (mạng, giao thông) bằng đồ thị thế nào? - Cải thiện độ phức tạp thuật toán đồ thị ra sao? - Cấu trúc tổ hợp đồ thị tiết lộ gì về hệ thống tính toán?
Ứng dụng trong TCS	- **Mạng máy tính**: Tối ưu định tuyến, băng thông. - **Thuật toán**: Dijkstra, Kruskal, Ford-Fulkerson. - **Tối ưu tổ hợp**: Lập lịch, phân bổ tài nguyên.

Hạng mục	Nội dung
Bài toán kinh điển	- **Ghép đôi tối đa**: Tìm tập cạnh lớn nhất không chung đỉnh. - **Tô màu đồ thị**: Tô đỉnh với số màu ít nhất. - **Đồng cấu đồ thị**: Kiểm tra hai đồ thị có cùng cấu trúc. - **Cầu Königsberg**: Không tồn tại đường đi Euler. - **Người đưa thư Trung Quốc**: Chu trình qua mọi cạnh với chi phí thấp nhất. - **4 màu**: Tô bản đồ với tối đa 4 màu. - **Hamiltonian vs Eulerian**: Đường đi qua mọi đỉnh hoặc cạnh.

Giải thích chi tiết bài toán kinh điển:

- **Ghép đôi tối đa**: Tìm tập cạnh lớn nhất không chung đỉnh, sử dụng thuật toán Hungarian hoặc Ford-Fulkerson. Ứng dụng trong phân bổ tài nguyên, lập lịch.
- **Tô màu đồ thị**: Tô các đỉnh sao cho không có đỉnh kề nhau cùng màu, với số màu ít nhất (NP-complete). Ứng dụng trong lập lịch, phân bổ tần số.
- **Đồng cấu đồ thị**: Xác định liệu hai đồ thị có cùng cấu trúc (Graph Isomorphism). Độ phức tạp chưa rõ (không biết là P hay NP). Ứng dụng trong phân tích mạng, nhận dạng mẫu.
- **Cầu Königsberg**: Euler chứng minh không tồn tại đường đi qua mọi cạnh đúng một lần, khai sinh lý thuyết đồ thị. Ứng dụng trong tối ưu hóa đường đi.
- **Người đưa thư Trung Quốc**: Tìm chu trình đi qua mọi cạnh ít nhất một lần với chi phí nhỏ nhất. Ứng dụng trong định tuyến, logistics.
- **4 màu**: Chứng minh bản đồ phẳng tô được bằng 4 màu (Four Color Theorem). Ứng dụng trong tô màu đồ thị, lập lịch.
- **Hamiltonian vs Eulerian**: Tìm đường đi qua mọi đỉnh (Hamilton, NP-complete) hoặc mọi cạnh (Euler, đa thức). Ứng dụng trong tối ưu hóa và phân tích mạng.

---

8. Lý thuyết số (Number Theory)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Số nguyên tố, chia hết**: Đồng dư, GCD, thuật toán Euclid. - **Hàm số học**: Euler, Mobius. - **Đồng dư**: Giải phương trình đồng dư, định lý Fermat, Euler. - **Mật mã số học**: Trường hữu hạn, nhóm cyclic. - **Phân tích số học**: Dãy số (Fibonacci, nguyên tố).
Bài toán trọng tâm	- Kiểm tra nguyên tố, phân tích thừa số nguyên tố. - Giải phương trình đồng dư. - Thiết kế giao thức mật mã (RSA, Diffie-Hellman). - Phân tích dãy số trong tổ hợp.
Câu hỏi lớn	- Bảo mật thông tin bằng tính chất số học thế nào? - Số nguyên tố phân bố ra sao? (Riemann Hypothesis) - Cấu trúc số học giải bài toán tính toán thế nào?
Ứng dụng trong TCS	- **Mật mã học**: RSA, mã hóa đường cong elliptic. - **Thuật toán**: Phân tích số, GCD (Euclid). - **Lý thuyết mã hóa**: Mã sửa lỗi trên trường hữu hạn.
Bài toán kinh điển	- **Kiểm tra nguyên tố**: Xác định số nguyên tố (Miller-Rabin, AKS). - **Phân tích số**: Phân tích số thành thừa số nguyên tố. - **Đồng dư tuyến tính**: Giải ax ≡ b (mod m). - **Fermat cuối cùng**: Không nghiệm nguyên cho x^n + y^n = z^n, n > 2.

Hạng mục	Nội dung
	- **Số hoàn hảo**: Tổng ước bằng chính nó. - **Goldbach**: Số chẵn > 2 là tổng hai số nguyên tố (chưa chứng minh).

Giải thích chi tiết bài toán kinh điển:

- **Kiểm tra nguyên tố**: Xác định số n là nguyên tố, dùng Miller-Rabin (ngẫu nhiên) hoặc AKS (xác định). Ứng dụng trong mật mã RSA, Diffie-Hellman.
- **Phân tích số**: Phân tích số hợp số thành thừa số nguyên tố, dùng Quadratic Sieve hoặc Number Field Sieve. Ứng dụng trong phá mã RSA.
- **Đồng dư tuyến tính**: Giải ax ≡ b (mod m), dùng thuật toán Euclid mở rộng. Ứng dụng trong mật mã, lập lịch.
- **Fermat cuối cùng**: Chứng minh x^n + y^n = z^n không có nghiệm nguyên không tầm thường với n > 2 (Wiles, 1995). Minh họa sức mạnh lý thuyết số.
- **Số hoàn hảo**: Tìm số mà tổng ước bằng chính nó (VD: 6, 28). Liên quan đến phân tích số học, ứng dụng trong tổ hợp.
- **Goldbach**: Giả thuyết mọi số chẵn > 2 là tổng hai số nguyên tố, chưa chứng minh. Ứng dụng trong phân tích số học và mật mã.

---

9. Hình học rời rạc (Discrete Geometry)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Điểm lưới**: Tọa độ nguyên. - **Bao lồi**: Tập lồi nhỏ nhất chứa các điểm. - **Đa giác, phân hoạch**: Phân chia không gian rời rạc. - **Định lý Pick**: Liên hệ diện tích, điểm lưới. - **Thuật toán hình học**: Giao điểm, khoảng cách, bao lồi.
Bài toán trọng tâm	- Tính diện tích, chu vi, số điểm lưới. - Tìm bao lồi, phân hoạch tối ưu. - Giải bài toán tối ưu hình học (TSP trên lưới). - Mô hình hóa cấu trúc không gian (vi mạch, đồ họa).
Câu hỏi lớn	- Mô tả, tối ưu hóa cấu trúc không gian rời rạc thế nào? - Thuật toán hình học đạt độ phức tạp nào? - Cấu trúc hình học tiết lộ gì về thiết kế hệ thống tính toán?
Ứng dụng trong TCS	- **Thiết kế vi mạch**: Sắp xếp thành phần. - **Đồ họa máy tính**: Tính bao lồi, phân hoạch. - **Hệ thống phân tán**: Mô hình vị trí, kết nối nút mạng.
Bài toán kinh điển	- **Bao lồi**: Tìm đa giác lồi nhỏ nhất chứa các điểm. - **Người bán hàng (TSP)**: Tìm hành trình ngắn nhất qua các điểm. - **Định lý Pick**: Tính diện tích đa giác trên lưới. - **Đóng gói hình tròn**: Sắp xếp hình tròn tối ưu. - **Erdős–Szekeres**: Tìm ngũ giác lồi trong tập điểm.

Giải thích chi tiết bài toán kinh điển:

- **Bao lồi**: Tìm đa giác lồi nhỏ nhất chứa các điểm, dùng Graham’s Scan hoặc Andrew’s Monotone Chain (O(n log n)). Ứng dụng trong đồ họa máy tính, thiết kế vi mạch.

- **Người bán hàng (TSP)**: Tìm hành trình ngắn nhất qua tất cả điểm và quay lại (NP-complete), giải bằng thuật toán xấp xỉ. Ứng dụng trong logistics, định tuyến.
- **Định lý Pick**: Tính diện tích đa giác trên lưới: Diện tích = I + B/2 - 1 (I: điểm bên trong, B: điểm trên biên). Ứng dụng trong đồ họa, phân tích hình học.
- **Đóng gói hình tròn**: Sắp xếp hình tròn tối ưu trong mặt phẳng, liên quan đến tối ưu không gian. Ứng dụng trong vật liệu học, thiết kế vi mạch.
- **Erdős–Szekeres**: Mọi tập n điểm trong mặt phẳng chứa tập con 5 điểm tạo ngũ giác lồi. Ứng dụng trong phân tích cấu trúc hình học.

---

10. Lý thuyết tính toán (Computability Theory)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Máy Turing**: Mô hình tính toán lý thuyết. - **Ngôn ngữ hình thức**: Chuỗi ký tự, ngữ pháp, automata. - **Tính quyết định**: Bài toán có thuật toán giải không. - **Bài toán dừng**: Không quyết định được. - **Độ phức tạp**: Phân loại bài toán theo tài nguyên.
Bài toán trọng tâm	- Xác định ngôn ngữ/bài toán có quyết định bởi máy Turing không. - Phân loại bài toán theo tính tính toán được. - Nghiên cứu giới hạn tính toán (bài toán dừng). - Phân tích độ phức tạp bài toán.
Câu hỏi lớn	- Bài toán nào máy tính giải được/không giải được? - Phân loại bài toán theo độ khó tính toán thế nào? - Giới hạn tính toán tiết lộ gì về trí tuệ, máy móc?
Ứng dụng trong TCS	- **Kiểm chứng hình thức**: Kiểm tra tính đúng đắn hệ thống. - **Thiết kế ngôn ngữ lập trình**: Ngôn ngữ hình thức cho trình biên dịch. - **AI lý thuyết**: Khả năng tính toán của mô hình học máy.
Bài toán kinh điển	- **Dừng (Halting)**: Kiểm tra chương trình có dừng không. - **Ngôn ngữ hình thức**: Kiểm tra ngôn ngữ được chấp nhận bởi automata. - **P vs NP**: Liệu P = NP? - **Post Correspondence**: Không quyết định được. - **Máy Turing**: Mô hình hóa tính toán.

Giải thích chi tiết bài toán kinh điển:

- **Dừng (Halting)**: Xác định liệu chương trình trên máy Turing có dừng lại hay không (Turing chứng minh không quyết định được). Ứng dụng trong phân tích chương trình, kiểm chứng phần mềm.
- **Ngôn ngữ hình thức**: Xác định liệu ngôn ngữ được chấp nhận bởi DFA, PDA, hoặc máy Turing. Ứng dụng trong thiết kế trình biên dịch, xử lý ngôn ngữ tự nhiên.
- **P vs NP**: Hỏi liệu mọi bài toán trong NP có thể được giải trong thời gian đa thức (P). Là câu hỏi mở lớn nhất trong TCS, ảnh hưởng đến thuật toán, mật mã.
- **Post Correspondence**: Tìm chuỗi ghép từ danh sách cặp chuỗi sao cho chuỗi trên và dưới giống nhau (không quyết định được). Minh họa giới hạn tính toán.
- **Máy Turing**: Mô hình tính toán trừu tượng, định nghĩa tính tính toán được. Ứng dụng trong lý thuyết tính toán, thiết kế ngôn ngữ lập trình.

---

11. Lý thuyết phạm trù (Category Theory)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Phạm trù**: Đối tượng, mũi tên thỏa mãn tính kết hợp, đơn vị. - **Hàm tử**: Ánh xạ bảo toàn cấu trúc. - **Biến đổi tự nhiên**: Ánh xạ giữa hàm tử. - **Tính đối ngẫu**: Quan hệ khái niệm đối lập. - **Cấu trúc đặc biệt**: Sản phẩm, coproduct, pullback, pushout.
Bài toán trọng tâm	- Mô tả cấu trúc toán học, tính toán bằng phạm trù. - Tìm hàm tử, biến đổi tự nhiên liên kết cấu trúc. - Nghiên cứu đối ngẫu để khám phá quan hệ. - Tổng quát hóa khái niệm trong logic, đại số, tính toán.
Câu hỏi lớn	- Thống nhất toán học, tính toán bằng ngôn ngữ chung thế nào? - Mô tả, phân tích cấu trúc tính toán qua phạm trù ra sao? - Phạm trù đơn giản hóa, khám phá khái niệm mới trong TCS thế nào?
Ứng dụng trong TCS	- **Ngữ nghĩa ngôn ngữ lập trình**: Mô tả kiểu dữ liệu, chương trình. - **Lý thuyết loại**: Nền tảng hệ thống loại trong lập trình hàm. - **Formal methods**: Kiểm chứng, thiết kế hệ thống.
Bài toán kinh điển	- **Tìm hàm tử**: Xây dựng hàm tử bảo toàn cấu trúc. - **Biến đổi tự nhiên**: Tìm biến đổi giữa hai hàm tử. - **Đối ngẫu phạm trù**: Khám phá quan hệ sản phẩm, coproduct. - **Yoneda Lemma**: Đối tượng xác định qua ánh xạ. - **Giới hạn/đối giới hạn**: Tìm cấu trúc phổ quát. - **Tích phủ (Pullback/Pushout)**: Tổng quát hóa giao, hợp có điều kiện.

Giải thích chi tiết bài toán kinh điển:

- **Tìm hàm tử**: Xây dựng hàm tử giữa hai phạm trù, ví dụ từ tập hợp sang nhóm cyclic. Ứng dụng trong ngữ nghĩa lập trình, lý thuyết loại.
- **Biến đổi tự nhiên**: Tìm ánh xạ giữa hai hàm tử bảo toàn cấu trúc, ví dụ trong không gian tô pô. Ứng dụng trong lập trình hàm (Haskell), kiểm chứng.
- **Đối ngẫu phạm trù**: Khám phá quan hệ giữa sản phẩm và coproduct, hỗ trợ thiết kế kiểu dữ liệu và kiểm chứng hình thức.
- **Yoneda Lemma**: Chứng minh mọi đối tượng trong phạm trù được xác định bởi các ánh xạ từ nó, nền tảng cho lý thuyết functor. Ứng dụng trong đại số trừu tượng, lập trình hàm.
- **Giới hạn/đối giới hạn**: Tìm cấu trúc phổ quát (giới hạn: giao, đối giới hạn: hợp), tổng quát hóa các phép toán. Ứng dụng trong logic, tô pô.
- **Tích phủ (Pullback/Pushout)**: Tổng quát hóa giao và hợp có điều kiện, ứng dụng trong kiểm chứng hệ thống, thiết kế kiểu dữ liệu.

12. Lý thuyết thông tin (Information Theory)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Entropy**: Đo lường mức độ không chắc chắn của nguồn thông tin (Shannon entropy: H(X) = -Σ p(x) log p(x)). - **Thông tin lẫn (Mutual Information)**: Đo lường thông tin chung giữa hai biến ngẫu nhiên. - **Mã hóa nguồn**: Nén dữ liệu để giảm kích thước (Huffman, Lempel-Ziv). - **Mã hóa kênh**: Đảm bảo truyền dữ liệu đáng tin cậy qua kênh nhiễu (Hamming, Reed-Solomon). - **Giới hạn Shannon**: Dung lượng kênh tối đa cho truyền thông không lỗi.
Bài toán trọng tâm	- Tính entropy, thông tin lẫn để phân tích hệ thống thông tin. - Thiết kế mã nén hiệu quả (mã hóa nguồn).

Hạng mục	Nội dung
	- Thiết kế mã sửa lỗi mạnh mẽ (mã hóa kênh). - Đánh giá dung lượng kênh truyền thông.
Câu hỏi lớn	- Làm thế nào để nén dữ liệu tối ưu mà không mất thông tin? - Truyền dữ liệu qua kênh nhiễu với độ tin cậy cao nhất ra sao? - Đo lường và tối ưu hóa thông tin trong hệ thống tính toán thế nào?
Ứng dụng trong TCS	- **Nén dữ liệu**: Giảm kích thước tệp (ZIP, JPEG, MP3). - **Mã sửa lỗi**: Đảm bảo truyền dữ liệu đáng tin cậy (Internet, lưu trữ). - **Học máy**: Phân tích thông tin lẫn trong mô hình học. - **Mật mã học**: Thiết kế giao thức an toàn dựa trên entropy.
Bài toán kinh điển	- **Mã hóa Huffman**: Xây dựng mã nén tối ưu dựa trên tần suất ký tự. - **Mã sửa lỗi Hamming**: Thiết kế mã phát hiện và sửa lỗi. - **Tính entropy**: Tính entropy của nguồn thông tin. - **Giới hạn Shannon**: Tính dung lượng kênh truyền thông. - **Kolmogorov Complexity**: Đo độ phức tạp thông tin của chuỗi.

Giải thích chi tiết bài toán kinh điển:

- **Mã hóa Huffman**: Xây dựng mã tiền tố tối ưu dựa trên tần suất xuất hiện của ký tự, sử dụng cây nhị phân (O(n log n)). Ứng dụng trong nén dữ liệu (ZIP, JPEG).
- **Mã sửa lỗi Hamming**: Thiết kế mã phát hiện và sửa lỗi bit dựa trên khoảng cách Hamming, sử dụng ma trận kiểm tra chẵn lẻ. Ứng dụng trong truyền thông, lưu trữ dữ liệu (CD, mạng).
- **Tính entropy**: Tính entropy H(X) của nguồn thông tin để đánh giá độ không chắc chắn hoặc lượng thông tin trung bình. Ứng dụng trong nén dữ liệu, phân tích học máy.
- **Giới hạn Shannon**: Tính dung lượng kênh C = B log(1 + S/N) (B: băng thông, S/N: tỷ số tín hiệu/nhiễu), xác định tốc độ truyền tối đa không lỗi. Ứng dụng trong thiết kế mạng, viễn thông.
- **Kolmogorov Complexity**: Đo độ phức tạp của chuỗi bằng độ dài chương trình ngắn nhất sinh ra nó (không tính được). Ứng dụng trong phân tích dữ liệu, lý thuyết thông tin.

13. Cấu trúc dữ liệu & giải thuật cơ bản & nâng cao (Data Structures & Algorithms)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Cấu trúc dữ liệu**: Cách tổ chức và lưu trữ dữ liệu (mảng, danh sách liên kết, cây, bảng băm, đồ thị). - **Giải thuật**: Quy trình từng bước để giải bài toán (tìm kiếm, sắp xếp, đệ quy, tham lam, quy hoạch động). - **Độ phức tạp**: Phân tích thời gian (O(n), O(log n)) và không gian. - **Cấu trúc cây**: Cây nhị phân, cây tìm kiếm, heap, trie. - **Thuật toán đồ thị**: Tìm đường đi, luồng, ghép đôi.
Bài toán trọng tâm	- Thiết kế cấu trúc dữ liệu hiệu quả cho bài toán cụ thể. - Phát triển thuật toán tối ưu về thời gian và không gian. - Phân tích độ phức tạp của thuật toán. - Giải bài toán tìm kiếm, sắp xếp, tối ưu trên cấu trúc dữ liệu.
Câu hỏi lớn	- Làm thế nào để tổ chức dữ liệu hiệu quả cho truy xuất nhanh? - Thiết kế thuật toán với độ phức tạp tối ưu nhất ra sao? - Cân bằng giữa thời gian và không gian trong thiết kế thuật toán thế nào?
Ứng dụng trong TCS	- **Hệ thống cơ sở dữ liệu**: Tối ưu truy vấn, lưu trữ (B-tree, bảng băm). - **Mạng máy tính**: Định tuyến, luồng dữ liệu (Dijkstra, Ford-Fulkerson). - **Học máy**: Tối ưu xử lý dữ liệu lớn (k-d tree, hash). - **Phần mềm**: Xây dựng hệ thống hiệu suất cao.

Hạng mục	Nội dung
Bài toán kinh điển	- **Tìm kiếm nhị phân**: Tìm phần tử trong mảng đã sắp xếp. - **Sắp xếp nhanh (QuickSort)**: Sắp xếp mảng bằng phân hoạch. - **Cây tìm kiếm nhị phân (BST)**: Quản lý dữ liệu động. - **Bảng băm**: Truy xuất nhanh với hàm băm. - **Quy hoạch động (Knapsack)**: Tối ưu bài toán chọn vật phẩm.

Giải thích chi tiết bài toán kinh điển:

- **Tìm kiếm nhị phân**: Tìm phần tử trong mảng đã sắp xếp với độ phức tạp O(log n), chia đôi không gian tìm kiếm mỗi bước. Ứng dụng trong cơ sở dữ liệu, tìm kiếm hiệu quả.
- **Sắp xếp nhanh (QuickSort)**: Sắp xếp mảng bằng cách chọn chốt và phân hoạch, độ phức tạp trung bình O(n log n). Ứng dụng trong xử lý dữ liệu, tối ưu hiệu suất.
- **Cây tìm kiếm nhị phân (BST)**: Cấu trúc cây cho phép chèn, xóa, tìm kiếm với độ phức tạp trung bình O(log n) (nếu cân bằng). Ứng dụng trong cơ sở dữ liệu, quản lý dữ liệu động.
- **Bảng băm**: Sử dụng hàm băm để ánh xạ khóa vào vị trí lưu trữ, truy xuất O(1) trung bình. Ứng dụng trong cơ sở dữ liệu, bộ nhớ đệm, mật mã.
- **Quy hoạch động (Knapsack)**: Tối ưu chọn vật phẩm với trọng lượng và giá trị giới hạn, sử dụng bảng lưu kết quả phụ (O(nW)). Ứng dụng trong lập lịch, phân bổ tài nguyên.

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Cấu trúc dữ liệu nâng cao**: Cây Fenwick (Binary Indexed Tree), cây Segment, cây Suffix, cấu trúc Union-Find có tối ưu hóa, van Emde Boas Tree, Splay Tree, Treap, Bloom Filter, Skip List, Persistent Data Structures. - **Thuật toán nâng cao**: Quy hoạch động tối ưu (Divide-and-Conquer Optimization, Knuth Optimization), thuật toán đồ thị phức tạp (Hopcroft-Karp, Dinic), thuật toán xấp xỉ cho bài toán NP-khó, thuật toán ngẫu nhiên nâng cao (Las Vegas, Monte Carlo), thuật toán trực tuyến. - **Độ phức tạp phân tích nâng cao**: Amortized analysis, competitive analysis, smoothed analysis. - **Cấu trúc dữ liệu động**: Hỗ trợ cập nhật và truy vấn trên dữ liệu thay đổi (Dynamic Connectivity, Range Queries). - **Cấu trúc dữ liệu xác suất**: Sử dụng xác suất để tối ưu hiệu suất (Bloom Filter, Cuckoo Hashing).
Bài toán trọng tâm	- Thiết kế cấu trúc dữ liệu hỗ trợ truy vấn và cập nhật động với độ phức tạp tối ưu (O(log n) hoặc O(1)). - Phát triển thuật toán cho bài toán NP-khó với xấp xỉ hoặc heuristic hiệu quả. - Phân tích hiệu suất thuật toán trong các kịch bản trực tuyến hoặc ngẫu nhiên. - Giải bài toán trên dữ liệu lớn với ràng buộc thời gian và không gian nghiêm ngặt. - Tối ưu hóa thuật toán đồ thị và tổ hợp (luồng tối đa, ghép đôi, phân hoạch).
Câu hỏi lớn	- Làm thế nào để thiết kế cấu trúc dữ liệu tối ưu cho các truy vấn động trên dữ liệu lớn? - Có thể đạt được độ phức tạp cận tuyến tính hoặc hằng số cho bài toán phức tạp không? - Thuật toán xấp xỉ hoặc ngẫu nhiên cải thiện hiệu suất thế nào so với thuật toán xác định? - Cân bằng giữa tính chính xác và hiệu suất trong thuật toán trực tuyến ra sao?
Ứng dụng trong TCS	- **Hệ thống cơ sở dữ liệu phân tán**: Tối ưu truy vấn phạm vi, kết nối động (Segment Tree, Union-Find). - **Mạng máy tính**: Tối ưu luồng mạng, định tuyến động (Dinic, Dynamic Shortest Path). - **Học máy**: Xử lý dữ liệu lớn, tìm kiếm gần đúng (Bloom Filter, Locality-Sensitive Hashing). - **Mật mã học**: Tối ưu cấu trúc dữ liệu cho bài toán tìm kiếm và xác minh (van Emde Boas, Suffix Tree). - **Xử lý văn bản và sinh học**: Tìm kiếm chuỗi, phân tích DNA (Suffix Array, Suffix Tree).
Bài toán kinh điển	- **Range Minimum Query (RMQ)**: Tìm giá trị nhỏ nhất trong đoạn bằng Segment Tree hoặc Sparse Table. - **Dynamic Connectivity**: Kiểm tra kết nối trong đồ thị động bằng Union-Find hoặc Link-Cut Tree. - **Luồng tối đa (Dinic)**: Tìm luồng tối đa trong mạng với thuật toán Dinic. - **Longest Common Substring**: Tìm chuỗi con chung dài nhất bằng Suffix Tree hoặc Suffix Array. - **Approximate String Matching**: Tìm chuỗi gần giống bằng Dynamic Programming hoặc Locality-Sensitive Hashing.

Giải thích chi tiết bài toán kinh điển:

- **Range Minimum Query (RMQ)**: Tìm giá trị nhỏ nhất trong đoạn [i, j] của mảng, sử dụng Segment Tree (O(log n) truy vấn, O(n) xây dựng) hoặc Sparse Table (O(1) truy vấn, O(n log n) xây dựng). Ứng dụng trong xử lý truy vấn phạm vi (cơ sở dữ liệu, học máy), phân tích dữ liệu lớn.
- **Dynamic Connectivity**: Xác định liệu hai đỉnh trong đồ thị động (thêm/xóa cạnh) có kết nối, sử dụng Union-Find với tối ưu hóa rank và path compression (O(α(n)) amortized) hoặc Link-Cut Tree (O(log n)). Ứng dụng trong mạng máy tính, hệ thống phân tán, phân tích đồ thị.
- **Luồng tối đa (Dinic)**: Tìm luồng tối đa trong mạng luồng bằng thuật toán Dinic, sử dụng BFS để tìm tầng và DFS để tìm luồng chặn (O(V²E) hoặc O(VE log V) với tối ưu). Ứng dụng trong tối ưu băng thông mạng, lập lịch, phân bổ tài nguyên.
- **Longest Common Substring**: Tìm chuỗi con chung dài nhất của hai chuỗi, sử dụng Suffix Tree (O(n)) hoặc Suffix Array (O(n log n)). Ứng dụng trong xử lý văn bản, phân tích DNA, nén dữ liệu.
- **Approximate String Matching**: Tìm chuỗi gần giống với chuỗi mục tiêu, sử dụng Dynamic Programming (Levenshtein Distance, O(nm)) hoặc Locality-Sensitive Hashing cho dữ liệu lớn. Ứng dụng trong tìm kiếm văn bản, nhận dạng mẫu, sinh học.

14. Phân tích thuật toán (Analysis of Algorithms)

Hạng mục	Nội dung
Khái niệm cốt lõi	- **Độ phức tạp thời gian**: Số phép toán theo kích thước đầu vào (O, Ω, Θ). - **Độ phức tạp không gian**: Bộ nhớ cần thiết cho thuật toán. - **Phân tích tiệm cận**: Đánh giá hiệu suất khi n → ∞. - **Phân tích ngẫu nhiên**: Xác suất hiệu suất của thuật toán ngẫu nhiên. - **NP-completeness**: Phân loại bài toán khó (P, NP, NP-complete).
Bài toán trọng tâm	- Tính toán độ phức tạp thời gian và không gian của thuật toán. - So sánh hiệu suất thuật toán (QuickSort vs MergeSort). - Xác định bài toán thuộc lớp P, NP, hay NP-complete. - Tối ưu thuật toán cho bài toán thực tế.
Câu hỏi lớn	- Làm thế nào để đánh giá chính xác hiệu suất thuật toán? - Có thể cải thiện thuật toán để đạt độ phức tạp thấp hơn không? - P = NP hay không, và điều này ảnh hưởng thế nào đến TCS?
Ứng dụng trong TCS	- **Thiết kế thuật toán**: Đảm bảo hiệu suất tối ưu (Dijkstra, A*). - **Hệ thống phân tán**: Tối ưu tài nguyên, độ trễ. - **Học máy**: Phân tích thuật toán học (gradient descent). - **Mật mã học**: Đánh giá độ phức tạp phá mã.
Bài toán kinh điển	- **Phân tích QuickSort**: Tính độ phức tạp trung bình O(n log n). - **NP-completeness (SAT)**: Chứng minh bài toán là NP-complete. - **Master Theorem**: Phân tích thuật toán đệ quy. - **Amortized Analysis**: Đánh giá chi phí trung bình của dãy thao tác. - **Phân tích thuật toán xấp xỉ**: Đánh giá hiệu suất thuật toán gần tối ưu.

Giải thích chi tiết bài toán kinh điển:

- **Phân tích QuickSort**: Tính độ phức tạp trung bình O(n log n) và trường hợp xấu nhất O(n²) của QuickSort, dựa trên phân hoạch ngẫu nhiên. Ứng dụng trong tối ưu thuật toán sắp xếp.
- **NP-completeness (SAT)**: Chứng minh bài toán thỏa mãn (SAT) là NP-complete (Cook-Levin), nền tảng để phân loại bài toán khó. Ứng dụng trong kiểm chứng, tối ưu.
- **Master Theorem**: Phân tích độ phức tạp của thuật toán đệ quy dạng T(n) = aT(n/b) + f(n), cung cấp công thức tổng quát. Ứng dụng trong phân tích thuật toán đệ quy (MergeSort, Strassen).
- **Amortized Analysis**: Đánh giá chi phí trung bình của dãy thao tác, ví dụ: chèn vào danh sách động (O(1) amortized). Ứng dụng trong thiết kế cấu trúc dữ liệu (vector, queue).

- **Phân tích thuật toán xấp xỉ**: Đánh giá tỷ lệ xấp xỉ của thuật toán cho bài toán NP-khó (VD: Vertex Cover với tỷ lệ 2). Ứng dụng trong tối ưu tổ hợp, lập lịch.

Bảng danh sách các bài toán kinh điển theo môn toán

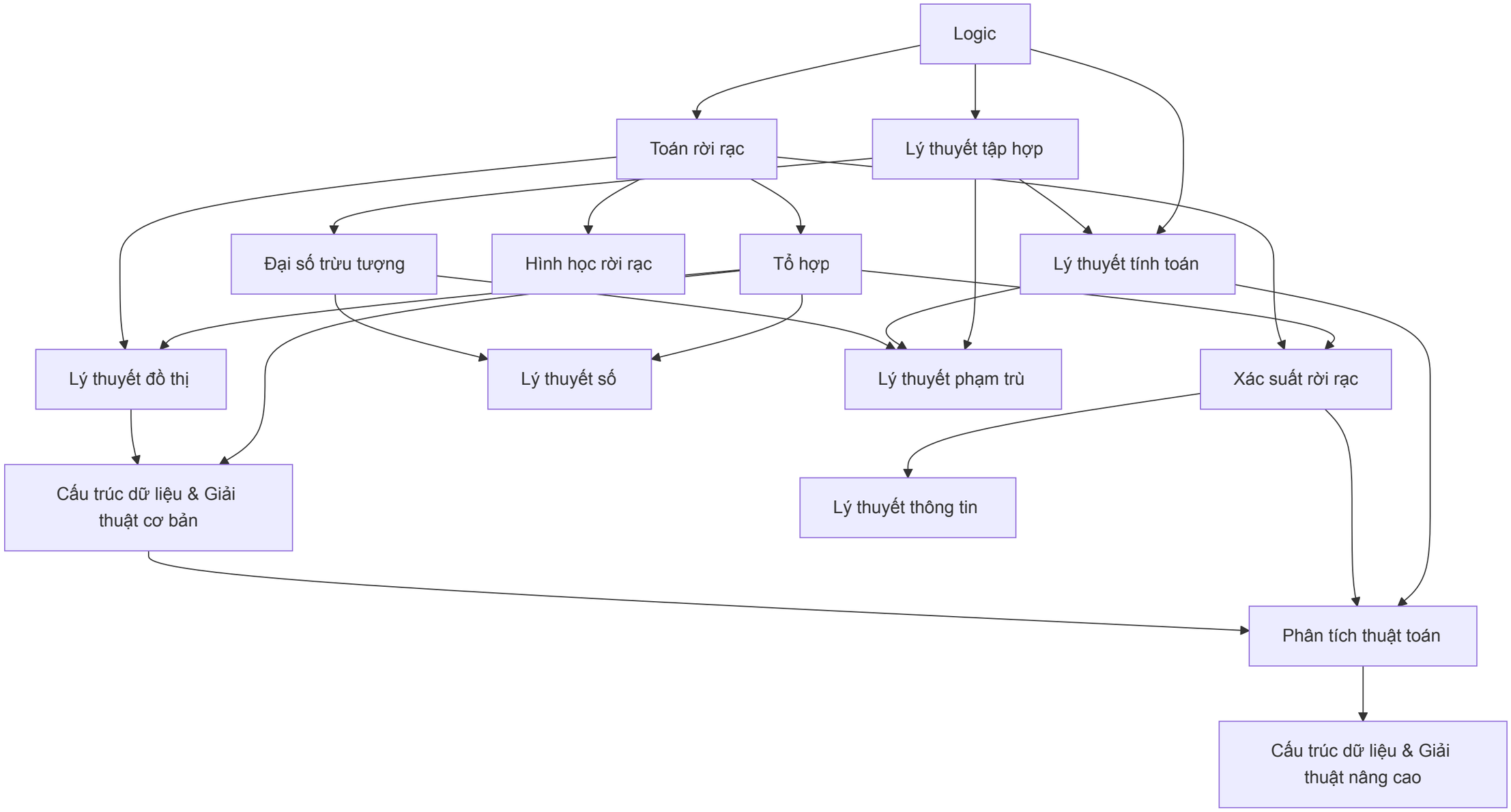
Môn toán	Tên bài toán	Mô tả ngắn gọn	Liên hệ với TCS	Ứng dụng thực tế
Đại số trừu tượng	**Phân loại nhóm hữu hạn**	Liệt kê nhóm hữu hạn theo bậc (cyclic, Abel, không Abel).	Biểu diễn dữ liệu, thuật toán đồ thị, mật mã học.	Mật mã (RSA), mã sửa lỗi, phân tích đối xứng.
	Tìm đồng cấu nhóm	Tìm ánh xạ bảo toàn cấu trúc giữa hai nhóm.	Biểu diễn dữ liệu, thuật toán (Graph Isomorphism).	Mật mã, phân tích dữ liệu, giao thức an toàn.
	Bài toán từ đẳng cấu	Xác định từ có là phần tử đơn vị trong nhóm.	Kiểm chứng hình thức, lý thuyết tính toán.	Mật mã, phân tích chương trình.
	Nhóm con Sylow	Tìm nhóm con Sylow của nhóm hữu hạn.	Phân tích cấu trúc nhóm, thuật toán.	Mật mã, phân tích đối xứng.
	Phân loại nhóm Abel	Phân loại nhóm Abel thành tổng nhóm cyclic.	Biểu diễn dữ liệu, lý thuyết mã hóa.	Mã sửa lỗi, phân tích dữ liệu.
	Bài toán Galois	Nghiên cứu tính giải được của đa thức qua nhóm đối xứng.	Mật mã, lý thuyết tính toán.	Mật mã, phân tích đa thức.
Logic	**Thỏa mãn (SAT)**	Tìm gán giá trị cho công thức CNF để đúng.	NP-complete, kiểm chứng hình thức.	Kiểm chứng phần mềm, AI, lập kế hoạch.
	Kiểm chứng mô hình	Xác định hệ thống thỏa mãn công thức logic.	Logic, biểu diễn dữ liệu.	Kiểm tra chip, giao thức mạng.
	Quyết định logic vị từ	Kiểm tra công thức đúng mọi mô hình (không quyết định).	Logic, lý thuyết tính toán.	Kiểm chứng, thiết kế ngôn ngữ lập trình.
	Định lý Gödel	Hệ tiên đề mạnh có mệnh đề không chứng minh được.	Logic, giới hạn tính toán.	Kiểm chứng, nền tảng toán học.
	Bài toán Thompson	Liên quan đến tính quyết định của hệ thống logic.	Logic, lý thuyết automata.	Kiểm chứng, phân tích hệ thống.
Toán rời rạc	**Đường đi ngắn nhất**	Tìm đường đi ngắn nhất trong đồ thị có trọng số.	Thiết kế thuật toán tối ưu.	Định tuyến mạng, GPS.
	Cây bao trùm tối thiểu	Tìm cây với tổng trọng số nhỏ nhất.	Tối ưu tổ hợp, thuật toán.	Thiết kế mạng, phân cụm dữ liệu.
	Luồng tối đa	Tìm luồng lớn nhất trong mạng.	Tối ưu tổ hợp, thuật toán.	Băng thông mạng, phân phối hàng.
	Tô màu bản đồ	Tô bản đồ với tối đa 4 màu.	Tối ưu tổ hợp, tô màu đồ thị.	Lập lịch, phân bổ tần số.
	Hanoi Tower	Di chuyển đĩa theo quy tắc (đệ quy).	Phân tích thuật toán đệ quy.	Phân tích thuật toán, giáo dục.
	Sắp xếp topo	Sắp xếp đỉnh trong DAG theo thứ tự topological.	Thuật toán, biểu diễn dữ liệu.	Lập lịch, phân tích phụ thuộc.
Tổ hợp	**Bao phủ tập hợp**	Tìm tập con nhỏ nhất bao phủ tập gốc.	Tối ưu tổ hợp, lý thuyết thông tin.	Tối ưu trạm phát sóng, lập lịch.
	Đếm hoán vị	Tính số cách sắp xếp với ràng buộc.	Tối ưu tổ hợp, lý thuyết thông tin.	Lập lịch, thiết kế thí nghiệm.
	Thiết kế khối	Xây dựng khối thỏa mãn ràng buộc.	Lý thuyết mã hóa, tối ưu tổ hợp.	Mã sửa lỗi, thiết kế thí nghiệm.
	Ramsey	Tìm số lớn nhất để đồ thị chứa cấu trúc con.	Tổ hợp đồ thị, lý thuyết thông tin.	Phân tích mạng, tổ hợp.
	36 sĩ quan Euler	Không tồn tại bảng Latin bậc 6.	Thiết kế tổ hợp, lý thuyết mã hóa.	Thiết kế thí nghiệm, mã hóa.
	Xếp domino	Phủ bảng bằng domino (Fibonacci).	Tối ưu tổ hợp, đếm cấu hình.	Phân tích cấu hình, tổ hợp.

Môn toán	Tên bài toán	Mô tả ngắn gọn	Liên hệ với TCS	Ứng dụng thực tế
	Các đường chéo đa giác	Số cách nối đỉnh đa giác (Catalan).	Tối ưu tổ hợp, thuật toán.	Phân tích cấu trúc, tổ hợp.
	Chia bài	Tính số cách chia bài theo quy tắc.	Tối ưu tổ hợp, xác suất.	Lập lịch, phân tích xác suất.
Lý thuyết tập hợp	**Đếm tập hợp**	So sánh số đếm tập hợp vô hạn.	Biểu diễn dữ liệu, logic.	Ngữ nghĩa lập trình, lý thuyết mô hình.
	Giả thuyết Continuum	Có tập hợp giữa $\aleph_0$ và $2^{\aleph_0}$ không?	Logic, lý thuyết mô hình.	Nền tảng toán học, logic vị từ.
	Ánh xạ song ánh	Chứng minh hai tập có cùng số đếm.	Biểu diễn dữ liệu, logic.	Thiết kế kiểu dữ liệu, kiểm chứng.
	Cantor	So sánh lực lượng tập hợp vô hạn.	Lý thuyết tính toán, logic.	Phân tích tính toán, ngữ nghĩa.
	Axiom of Choice	Chọn phần tử từ hệ tập.	Logic, kiểm chứng hình thức.	Kiểm chứng, nền tảng toán học.
	Nghịch lý Russell	Tập hợp không chứa chính nó (mâu thuẫn).	Logic, lý thuyết tập hợp.	Nền tảng logic, kiểm chứng.
Xác suất rời rạc	**Phân tích thuật toán ngẫu nhiên**	Phân tích QuickSort, Miller-Rabin.	Thuật toán tối ưu, lý thuyết thông tin.	Mật mã, học máy, phân tích dữ liệu.
	Chuỗi Markov	Tính trạng thái ổn định, thời gian trộn.	Biểu diễn dữ liệu, lý thuyết thông tin.	Phân tích mạng, mô phỏng hệ thống.
	Chernoff	Phân tích xác suất sự kiện ngẫu nhiên.	Lý thuyết thông tin, thuật toán.	Phân tích mạng, học máy.
	Sinh nhật	Xác suất trùng ngày sinh.	Xác suất, lý thuyết thông tin.	Mật mã, phân tích xác suất.
	Monty Hall	Đổi cửa tăng cơ hội thắng.	Xác suất có điều kiện.	Giáo dục, phân tích xác suất.
	Xếp ngẫu nhiên mũ	Xác suất không ai đội đúng mũ.	Xác suất, xáo trộn.	Phân tích xáo trộn, xác suất.
	Sinh con trước	Xác suất chuỗi xuất hiện trước.	Xác suất, phân tích chuỗi.	Phân tích chuỗi, xác suất.
Lý thuyết đồ thị	**Ghép đôi tối đa**	Tìm tập cạnh lớn nhất không chung đỉnh.	Tối ưu tổ hợp, thuật toán.	Phân bổ tài nguyên, lập lịch.
	Tô màu đồ thị	Tô đỉnh với số màu ít nhất.	Tối ưu tổ hợp, thuật toán xấp xỉ.	Lập lịch, phân bổ tần số.
	Đồng cấu đồ thị	Kiểm tra hai đồ thị có cùng cấu trúc.	Đại số, thuật toán tối ưu.	Phân tích mạng, nhận dạng mẫu.
	Cầu Königsberg	Không tồn tại đường đi Euler.	Lý thuyết đồ thị, tối ưu hóa.	Tối ưu đường đi, logistics.
	Người đưa thư Trung Quốc	Chu trình qua mọi cạnh chi phí thấp nhất.	Tối ưu tổ hợp, thuật toán.	Định tuyến, logistics.
	4 màu	Tô bản đồ với tối đa 4 màu.	Tối ưu tổ hợp, tô màu đồ thị.	Lập lịch, phân bổ tần số.
	Hamiltonian vs Eulerian	Đường đi qua mọi đỉnh/cạnh.	Tối ưu tổ hợp, thuật toán.	Tối ưu hóa mạng, phân tích.
Lý thuyết số	**Kiểm tra nguyên tố**	Xác định số nguyên tố (Miller-Rabin, AKS).	Thuật toán tối ưu, lý thuyết số.	Mật mã (RSA), bảo mật hệ thống.
	Phân tích số	Phân tích số thành thừa số nguyên tố.	Thuật toán, đại số trừu tượng.	Mật mã (phá RSA), bảo mật.
	Đồng dư tuyến tính	Giải $ax \equiv b \pmod m$.	Lý thuyết số, thuật toán.	Mật mã, lập lịch, mã hóa.
	Fermat cuối cùng	Không nghiệm $x^n + y^n = z^n$, $n > 2$.	Lý thuyết số, phân tích.	Nền tảng toán học, mật mã.
	Số hoàn hảo	Tổng ước bằng chính nó.	Phân tích số học, tổ hợp.	Phân tích số, tổ hợp.
	Goldbach	Số chẵn > 2 là tổng hai số nguyên tố.	Phân tích số học, mật mã.	Mật mã, phân tích số.
Hình học rời rạc	**Bao lồi**	Tìm đa giác lồi nhỏ nhất chứa các điểm.	Thuật toán tối ưu, biểu diễn dữ liệu.	Đồ họa, thiết kế vi mạch.
	Người bán hàng (TSP)	Tìm hành trình ngắn nhất qua các điểm.	Tối ưu tổ hợp, thuật toán xấp xỉ.	Logistics, định tuyến mạng.
	Định lý Pick	Tính diện tích đa giác trên lưới.	Biểu diễn dữ liệu, tổ hợp.	Đồ họa, phân tích hình học.
	Đóng gói hình tròn	Sắp xếp hình tròn tối ưu.	Tối ưu không gian, thuật toán.	Vật liệu học, thiết kế vi mạch.

Môn toán	Tên bài toán	Mô tả ngắn gọn	Liên hệ với TCS	Ứng dụng thực tế
	Erdős–Szekeres	Tìm ngũ giác lồi trong tập điểm.	Phân tích cấu trúc hình học.	Phân tích hình học, tổ hợp.
Lý thuyết tính toán	**Dừng (Halting)**	Kiểm tra chương trình có dừng không.	Logic, giới hạn tính toán.	Phân tích chương trình, kiểm chứng.
	Ngôn ngữ hình thức	Kiểm tra ngôn ngữ được chấp nhận bởi automata.	Biểu diễn dữ liệu, lý thuyết tính toán.	Trình biên dịch, xử lý ngôn ngữ.
	P vs NP	Liệu P = NP?	Độ phức tạp, thuật toán tối ưu.	Tối ưu thuật toán, mật mã.
	Post Correspondence	Không quyết định được.	Lý thuyết tính toán, logic.	Giới hạn tính toán, kiểm chứng.
	Máy Turing	Mô hình hóa tính toán.	Lý thuyết tính toán, biểu diễn dữ liệu.	Thiết kế ngôn ngữ, lý thuyết AI.
Lý thuyết phạm trù	**Tìm hàm tử**	Xây dựng hàm tử bảo toàn cấu trúc.	Biểu diễn dữ liệu, suy luận cấu trúc.	Ngữ nghĩa lập trình, lý thuyết loại.
	Biến đổi tự nhiên	Tìm biến đổi giữa hai hàm tử.	Logic, biểu diễn dữ liệu.	Lập trình hàm, kiểm chứng.
	Đối ngẫu phạm trù	Khám phá quan hệ sản phẩm, coproduct.	Biểu diễn dữ liệu, suy luận.	Thiết kế kiểu dữ liệu, kiểm chứng.
	Yoneda Lemma	Đối tượng xác định qua ánh xạ.	Lý thuyết functor, đại số.	Lập trình hàm, kiểm chứng.
	Giới hạn/đối giới hạn	Tìm cấu trúc phổ quát.	Logic, tổng quát hóa.	Logic, kiểm chứng hệ thống.
	Tích phủ (Pullback/Pushout)	Tổng quát hóa giao, hợp có điều kiện.	Logic, biểu diễn dữ liệu.	Kiểm chứng, thiết kế kiểu dữ liệu.
Lý thuyết thông tin	**Mã hóa Huffman**	Xây dựng mã tiền tố tối ưu dựa trên tần suất ký tự.	Nén dữ liệu, lý thuyết thông tin.	Nén tệp (ZIP, JPEG), truyền thông.
	Mã sửa lỗi Hamming	Thiết kế mã phát hiện và sửa lỗi bit.	Mã hóa kênh, lý thuyết thông tin.	Truyền thông, lưu trữ (CD, mạng).
	Tính entropy	Tính entropy của nguồn thông tin.	Phân tích thông tin, nén dữ liệu.	Nén dữ liệu, học máy, mật mã.
	Giới hạn Shannon	Tính dung lượng kênh truyền thông.	Lý thuyết thông tin, truyền thông.	Thiết kế mạng, viễn thông.
	Kolmogorov Complexity	Đo độ phức tạp thông tin của chuỗi.	Lý thuyết thông tin, phân tích dữ liệu.	Phân tích dữ liệu, nén dữ liệu.
Cấu trúc dữ liệu & Giải thuật (cơ bản & nâng cao)	**Tìm kiếm nhị phân**	Tìm phần tử trong mảng đã sắp xếp.	Thuật toán tối ưu, biểu diễn dữ liệu.	Cơ sở dữ liệu, tìm kiếm hiệu quả.
	Sắp xếp nhanh (QuickSort)	Sắp xếp mảng bằng phân hoạch.	Thuật toán tối ưu, xử lý dữ liệu.	Xử lý dữ liệu, tối ưu hiệu suất.
	Cây tìm kiếm nhị phân (BST)	Quản lý dữ liệu động bằng cây.	Biểu diễn dữ liệu, thuật toán.	Cơ sở dữ liệu, quản lý dữ liệu.
	Bảng băm	Truy xuất nhanh với hàm băm.	Biểu diễn dữ liệu, thuật toán.	Cơ sở dữ liệu, bộ nhớ đệm, mật mã.
	Quy hoạch động (Knapsack)	Tối ưu chọn vật phẩm với giới hạn.	Tối ưu tổ hợp, thuật toán.	Lập lịch, phân bổ tài nguyên.
	Range Minimum Query (RMQ)	Tìm giá trị nhỏ nhất trong đoạn bằng Segment Tree hoặc Sparse Table.	Biểu diễn dữ liệu, thuật toán tối ưu.	Cơ sở dữ liệu, học máy, phân tích dữ liệu.
	Dynamic Connectivity	Kiểm tra kết nối trong đồ thị động bằng Union-Find hoặc Link-Cut Tree.	Biểu diễn dữ liệu, thuật toán đồ thị.	Mạng máy tính, hệ thống phân tán.
	Luồng tối đa (Dinic)	Tìm luồng tối đa trong mạng với thuật toán Dinic.	Tối ưu tổ hợp, thuật toán đồ thị.	Băng thông mạng, lập lịch, phân bổ tài nguyên.
	Longest Common Substring	Tìm chuỗi con chung dài nhất bằng Suffix Tree hoặc Suffix Array.	Xử lý chuỗi, thuật toán văn bản.	Phân tích DNA, xử lý văn bản, nén dữ liệu.

Môn toán	Tên bài toán	Mô tả ngắn gọn	Liên hệ với TCS	Ứng dụng thực tế
	Approximate String Matching	Tìm chuỗi gần giống bằng Dynamic Programming hoặc LSH.	Xử lý chuỗi, thuật toán xấp xỉ.	Tìm kiếm văn bản, sinh học, nhận dạng mẫu.
Phân tích thuật toán	**Phân tích QuickSort**	Tính độ phức tạp trung bình O(n log n).	Phân tích độ phức tạp, thuật toán.	Tối ưu thuật toán sắp xếp.
	NP-completeness (SAT)	Chứng minh bài toán là NP-complete.	Độ phức tạp, lý thuyết tính toán.	Kiểm chứng, tối ưu, mật mã.
	Master Theorem	Phân tích thuật toán đệ quy.	Phân tích độ phức tạp, thuật toán.	Phân tích MergeSort, Strassen.
	Amortized Analysis	Đánh giá chi phí trung bình của dãy thao tác.	Phân tích độ phức tạp, cấu trúc dữ liệu.	Thiết kế cấu trúc dữ liệu, queue.
	Phân tích thuật toán xấp xỉ	Đánh giá hiệu suất thuật toán gần tối ưu.	Tối ưu tổ hợp, thuật toán.	Lập lịch, tối ưu tổ hợp.

Thứ tự học đề xuất



Các môn chuyên sâu hơn sau đó

Complexity Theory, Model Theory, Formal Methods, Type Theory