1. Qui hoạch động : Quy hoạch động giải các bài toán bằng cách kết hợp các lời giải của các bài toán con của bài toán đang xét.
Phương pháp này khả dụng khi các bài toán con không độc lập đối với nhau, tức là khi các bài toán con có dùng chung những bài toán "cháu". Qui hoạch động giải các bài toán "cháu" dùng chung này một lần và lưu lời giải của chúng trong một bảng và sau đó khỏi phải tính lại khi gặp lại bài toán cháu đó. Qui hoạch động được áp dụng cho những bài toán tối ưu hóa (optimization problem).
Bốn bước của qui hoạch động : Đặc trưng hóa cấu trúc của lời giải tối ưu. + Định nghĩa giá trị của lời giải tối ưu một cách đệ quy. + Tính trị của lời giải tối ưu theo kiểu **từ dưới lên**. + Cấu tạo lời giải tối ưu từ những thông tin đã được tính toán.
Các thành phần của quy hoạch động : + Tiểu cấu trúc tối ưu - Một bài toán có tính chất tiểu cấu trúc tối ưu nếu lời giải tối ưu chứa trong nó những lời giải tối ưu của những bài toán con. + Những bài toán con trùng lắp - Khi một giải thuật đệ quy gặp lại cùng một bài toán con nhiều lần, ta bảo rằng bài toán tối ưu hóa có những bài toán con trùng lắp.
Chuỗi con chung dài nhất LCS : O(m+n)
Bài toán cái túi (Knapsack) : Bài toán cái túi có thể dễ dàng giải được nếu M không lớn, nhưng khi M lớn thì thời gian chạy trở nên không thể chấp nhận được. Phương pháp này không thể làm việc được khi M và trọng lượng/kích thước là những số thực thay vì số nguyên. Giải thuật qui hoạch động để giải bài toán cái túi có thời gian chạy **tỉ lệ với NM**.
Giải thuật Warshall [$O(V^3)$- Giải thuật Floyd [$O(V^3)$] : thể hiện sự áp dụng chiến lược quy hoạch động vì sự tính toán căn cứ vào một hệ thức truy hồi nhưng lại không xây dựng thành giải thuật đệ quy. Thay vào đó là một giải thuật lặp với sự hỗ trợ của một ma trận để lưu trữ các kết quả trung gian.
2. Giải thuật tham lam
Các giải thuật tối ưu hóa thường đi qua một số bước với một tập các khả năng lựa chọn tại mỗi bước. Một giải thuật tham lam thường chọn một khả năng mà xem như **tốt nhất tại lúc đó**. Tức là, giải thuật chọn một khả năng tối ưu cục bộ với hy vọng sẽ dẫn đến một lời giải tối ưu toàn cục. VD : +Bài toán xếp lịch cho các hoạt động + Bài toán cái túi dạng phân số + Bài toán mã Huffman+ Giải thuật Prim để tính cây bao trùm tối thiểu.
Hai thành phần chính của giải thuật tham lam :
+ Tính chất lựa chọn tham lam : Lựa chọn được thực hiện bởi giải thuật tham lam tùy thuộc vào những lựa chọn đã làm cho đến bây giờ, nhưng nó không tùy thuộc

vào bất kỳ lựa chọn trong tương lai hay những lời giải của những bài toán con. Như vậy, một giải thuật tham lam tiến hành theo kiểu **từ trên xuống**, thực hiện mỗi lúc một lựa chọn tham lam.
+ Tiểu cấu trúc tối ưu: Một bài tóan có tính chất tiểu cấu trúc tối ưu nếu một lời giải tối ưu chứa trong nó những lời giải tối ưu cho những bài toán con.
Dùng giải thuật tham lam cho bài toán cái túi dạng phân số và qui hoạch động cho bài toán cái túi dạng 0-1.
Giải thuật tham lam cho bài toán xếp lịch các hoạt động:
Hoạt động được chọn bởi thủ tục GREEDY-ACTIVITY-SELECTER thường là hoạt động với *thời điểm kết thúc sớm nhất* mà có thể được xếp lịch một cách hợp lệ. Hoạt động được chọn theo cách "tham lam" theo nghĩa nó sẽ để lại cơ hội để xếp lịch cho được nhiều hoạt độngkhác. Giải thuật tham lam không nhất thiết đem lại lời giải tối ưu. Tuy nhiên thủ tục GREEDY-ACTIVITY-SELECTOR thường tìm được một lời giải tối ưu cho một thể hiện của bài toán xếp lịch các hoạt động.
Bài toán cái túi dạng phân số (knapsack) : **O(n) + Giải thuật HUFFMAN** (dùng phổ biến và rất hữu hiệu cho việc nén dữ liệu) trên tập *n* ký tự sẽ là **:**
O(nlgn) + Bài toán tô màu đồ thị : Đầu tiên ta cố tô cho được nhiều đỉnh với màu đầu tiên, và rồi dùng một màu mới tô các đỉnh chưa tô sao cho tô được càng nhiều đỉnh càng tốt. Và quá trình này được lặp lại với những màu khác cho đến khi mọi đỉnh đều được tô màu**. Độ phức tạp** của thủ tục SAME_COLOR: **O(n).** Nếu m là số màu được dùng để tô đồ thị thì thủ tục SAME_COLOR được gọi tất cả m lần. Do đó, độ phức tạp của toàn giải thuật: **m* O(n).** Vì m thường là một số nhỏ⇒ độ phức tạp tuyến tính . Ứng dụng : Xếp lịch thi học kỳ , gán tần số trong lĩnh vực vô tuyến ,điện thoại di động.
3. Giải thuật quay lui : "bước hướng về lời giải đầy đủ và ghi lại thông tin về bước này mà sau đó nó có thể bị tháo gỡ và xóa đi khi phát hiện rằng bước này đã không dẫn đến lời giải đầy đủ, tức là một bước đi dẫn đến "tình thế bế tắc"(dead-end). (Hành vi này được gọi là quay lui - backtracking.) VD : bài toán tám con hậu ,bài toán con mã đi tuần
Một phương pháp tổng quát để giải quyết vấn đề: thiết kế giải thuật tìm lời giải cho bài tóan không phải là bám theo một tập qui luật tính tóan được xác định mà là bằng cách *thử và sửa sai* .Khuôn mẫu thông thường là phân rã quá trình thử và sửa sai thành những công tác bộ phận. Thường thì những công tác bộ phận này được diễn tả theo lối đệ quy một cách thuận tiện và bao gồm việc *thăm dò một số hữu*

hạn những công tác con.Ta có thể coi toàn bộ quá trình này như là một *quá trình tìm kiếm* mà dần dần cấu tạo và duyệt qua một cây các công tác con.

Tìm tất cả các lời giải : Một khi một lời giải được tìm thấy và ghi lại, ta tiếp tục xét ứng viên kế trong quá trình chọn ứng viên một cách có hệ thống.

Thời gian tính toán của các giải thuật quay lui thường là hàm mũ (*exponential*). Nếu mỗi nút trên cây không gian trạng thái có trung bình α nút con, và chiều dài của lối đi lời giải là N, thì số nút trên cây sẽ **tỉ lệ với α^N.**

4. Giải thuật nhánh và cận (branch-and-bound)

Ý tưởng nhánh và cận: Trong quá trình tìm kiếm một lối đi tốt nhất (tổng trọng số nhỏ nhất) cho bài toán TSP, có một kỹ thuật tỉa nhánh quan trọng là kết thúc sự tìm kiếm ngay khi thấy rằng nó không thể nào thành công được. Giả sử một lối đi đơn có chi phí x đã được tìm thấy. Thì thật vô ích để duyệt tiếp trên lối đi chưa-đầy-đủ nào mà chi phí cho đến hiện giờ đã lớn hơn x. Điều này có thể được thực hiện bằng cách không gọi đệ quy thủ tục visit nếu lối đi chưa-đầy-đủ hiện hành đã lớn hơn chi phí của lối đi đầy đủ tốt nhất cho đến bây giờ. Rõ ràng ta sẽ không bỏ sót lối đi chi phí nhỏ nhất nào nếu ta bám sát một chiến lược như vậy. Kỹ thuật tính cận (bound) của các lời giải chưa-đầy-đủ để hạn chế số lời giải phải dò tìm được gọi là *giải thuật nhánh và cận*.

Giải thuật này có thể áp dụng khi có chi phí được gắn vào các lối đi.

Bài toán người thương gia du hành (TSP) + Bài toán Chu trình Hamilton (HCP) : Để giải bài toán (HCP), ta có thể cải biên giải thuật tìm kiếm theo chiều sâu trước (DFS) để giải thuật này có thể sinh ra mọi lối đi đơn mà đi qua mọi đỉnh trong đồ thị.

5. NP-Complete

P : Tập hợp tất cả những bài toán có thể giải được bằng những giải thuật tất định trong thời gian đa thức.

NP: tập hợp tất cả những bài toán mà có thể được giải bằng giải thuật không tất định trong thời gian đa thức.

VD : Bài toán có tồn tại lối đi dài nhất từ đỉnh x đến đỉnh y ; Bài toán thỏa mãn mạch logic CSP là một bài toán thuộc lớp NP

Tất định : khi giải thuật đang làm gì, cũng chỉ có một việc duy nhất có thể được thực hiện kế tiếp.

VD **:** Xếp thứ tự bằng phương pháp chèn thuộc lớp P vì có độ phức tạp đa thức $O(N^2)$

Không tất định: khi một giải thuật gặp một sự lựa chọn giữa nhiều khả năng, nó có quyền năng "tiên đóan" để biết chọn một khả năng thích đáng. VD : Cho A là một mảng số nguyên. Một giải thuật không tất định NSORT(A, n) sắp thứ tự các số theo thứ tự tăng và xuất chúng ra theo thứ tự này.
Sự phân giải một giải thuật không tất định có thể được thực hiện bằng một sự song song hóa không hạn chế .Mỗi lần có bước lựa chọn phải thực hiện, giải thuật tạo ra nhiều bản sao của chính nó .Mỗi bản sao được thực hiện cho khả năng lựa chọn. Như vậy nhiều khả năng được thực hiện cùng một lúc : +Bản sao đầu tiên kết thúc thành công thì làm kết thúc tất cả các quá trình tính tóan khác + Nếu một bản sao kết thúc thất bại thì chỉ bản sao ấy kết thúc mà thôi.
NP-complete : Có một danh sách những bài toán mà đã biết là thuộc về lớp NP nhưng không rõ có thể thuộc về lớp P hay không. Tức là, ta giải chúng dễ dàng trên một máy không tất định nhưng chưa ai có thể tìm thấy một giải thuật hữu hiệu chạy trên máy tính thông thường để giải bất kỳ một bài toán nào của chúng.Những bài toán NP này lại có thêm một tính chất:"Nếu bất kỳ một trong những bài toán này có thể giải được trong thời gian đa thức thì tất cả những bài toán thuộc lớp NP cũng sẽ được giải trong thời gian đa thức trên một máy tất định." Đây là bài toán **NP-complete .** Để chứng minh một bài toán thuộc loại NP là NP-đầy đủ, ta chỉ cần chứng tỏ rằng một bài toán NP-đầy đủ đã biết nào đó thì khả thu giảm đa thức về bài toán mới ấy.
Một số bài toán NP-đầy đủ : - Bài toán thỏa mãn mạch logic CSP : Nếu tồn tại một giải thuật thời gian đa thức để giải bài toán thỏa mãn mạch logic thì tất cả mọi bài toán trong lớp NP có thể được giải trong thời gian đa thức - **Bài toán phân hoạch số**: Cho một tập những số nguyên, có thể phân hoạch chúng thành hai tập con mà có tổng trị số bằng nhau ? - **Bài toán qui hoạch nguyên**: Cho một bài toán qui hoạch tuyến tính, liệu có tồn tại một lời giải toàn số nguyên - **Xếp lịch công việc trên đa bộ xử lý** : Cho một kỳ hạn và một tập các công tác có chiều dài thời gian khác nhau phải được thực thi trên hai bộ xử lý. Vấn đề là có thể sắp xếp để thực thi tất cả những công tác đó sao cho thỏa mãn kỳ hạn không - **Bài toán phủ đỉnh** (VERTEX COVER): Cho một đồ thị và một số nguyên N, có thể kiếm được một tập nhỏ hơn N đỉnh mà chạm hết mọi cạnh trong đồ thị - **Bài toán xếp thùng** (BIN PACKING): cho n món đồ mà phải đặt vào trong các thùng có sức chứa bằng nhau L. Món đồ i đòi hỏi l_i đơn vị sức chứa của thùng. Mục đích là xác định số thùng ít nhất cần để chứa tất cả n món đồ đó.? **Bài toán người thương gia du hành** (TSP): cho một tập các thành phố và khoảng cách giữa mỗi cặp thành phố,

tìm một lộ trình đi qua tất cả mọi thành phố sao cho tổng khoảng cách của lộ trình nhỏ hơn M+? **Bài toán chu trình Hamilton** (HCP): Cho một đồ thị, tìm một chu trình đơn mà đi qua tất cả mọi đỉnh.
Bài toán **NP-đầy đủ** trong các lãnh vực : giải tích số, sắp thứ tự và tìm kiếm, xử lý dòng ký tự, Mô hình hóa hình học, xử lý đồ thị. Sự đóng góp quan trọng nhất của lý thuyết về NP-đầy đủ là: *nó cung cấp một cơ chế để xác định một bài toán mới trong các lãnh vực trên là "dễ" hay "khó".***Một số kỹ thuật để đối phó với những bài toán NP-đầy đủ :** + Dùng "giải thuật xấp xỉ để tìm lời giải xấp xỉ tối ưu (near-optimal) + Dựa vào hiệu năng của trường hợp trung bình để phát triển một giải thuật mà tìm ra lời giải trong một số trường hợp nào đó, mặc dù không làm việc được trong mọi trường hợp+ Sử dụng những giải thuật có độ phức tạp hàm mũ nhưng hữu hiệu, ví dụ như giải thuật quay lui+ Đưa heuristic vào giải thuật để tăng thêm hiệu quả của giải thuật+ Sử dụng metaheuristic.
Heuristic là tri thức về bài toán cụ thể được sử dụng để dẫn dắt quá trình tìm ra lời giải của giải thuật. Nhờ sự thêm vào các heuristic mà giải thuật trở nên hữu hiệu hơn.
Meta heuristic là loại heuristic tổng quát có thể áp dụng cho nhiều lớp bài tóan.Gần đây meta heuristic là một lãnh vực nghiên cứu phát triển mạnh mẽ, với sự ra đời của nhiều meta heuristic như:- giải thuật di truyền - giải thuật mô phỏng luyện kim - tìm kiếm tabu (Tabu search) …
Bốn lớp bài toán phân theo độ khó:
Những bài toán bất khả quyết : Đây là những bài toán chưa hề có giải thuật để giải._VD: Bài toán quyết định xem một chương trình có dừng trên một máy Turing + **Những bài toán khó giải** : đây là những bài toán mà không tồn tại giải thuật thời gian đa thức để giải chúng. Chỉ tồn tại giải thuật thời gian hàm mũ để giải chúng + **Những bài toán NP-đầy đủ** : Những bài toán NP-đầy đủ là một lớp con đặc biệt của lớp bài toán NP + **Những bài toán P**.
Approximation algorithm : An algorithm that returns near-optimal solutions is called an *approximation algorithm.*
The Vertex-Cover Problem : O(E)
Bruce-force : Sequential search, selection sort + Divide-and-conquer : Quicksort, mergesort, binary search
Decrease-and-conquer : Insertion sort, DFS, BFS , Sắp xếp Tôpô , Sinh hoán vị PERM + Transform-and-conquer : Heapsort, Gauss elimination, Horner , Rabin-Karp + Greedy : Prim's, Dijkstra's , Cái túi dạng phân số, Mã HUFFMAN, Tô màu

đồ thị, Xếp lịch các hoạt động + Dynamic Programming : Floyd’s, Warshall, Cái túi Knapsack, Chuỗi con chung dài nhất LCS, Nhân xâu ma trận