**Chương 4: Chiến lược biến thể để trị**

* **Thường làm việc với hai bước:** 
  + B1: là bước biến thể, thể hiện qua việc bài toán được biến đổi để chuyển sang dạng dễ dẫn đến lời giải.
  + B2: là bước tìm ra lời giải cho bài toán
  + Có 3 biến dạng chính của bước 1: Biến thể đưa đến 1 thể hiện đơn giản hơn (đơn giản hóa thể hiện – instance simplification); Biến thể đưa đến 1 biểu diễn khác (biến đổi biểu diễn – representation change); Biến thể đưa đến 1 thể hiện của bài toán khác đã có tồn tại giải thuật để giải (thu giảm bài toán – problem reduction)
* **Các giải thuật liên quan:**
  + Giải thuật PresortUniqueElement: ở giải thuật này bước biển thể xếp thứ tự của mảng trước đã giúp đưa mảng về dạng thức dễ kiểm tra hơn. Độ phức tạp của giải thuật này là tổng độ phức tạp của việc sắp thứ tự mảng và kiểm tra các phần tử kế cận nhau, vì việc sắp thứ tự độ phức tạp nlogn trong trường hợp xấu nhất và công tác kiểm tra tốn n-1 (O(n)) tác vụ so sánh nên độ phức tạp giải thuật = O(nlogn), (nếu sử dụng phương pháp trực tiếp thì là O(n2))
  + Giải thuật Guass để giải hệ pt tuyến tính: biến đổi hệ thống n phương trình tuyến tính với n thành 1 hệ thống tương đương, thể hiện tinh thần “đơn giản hóa thể hiện” (độ phức tạp là O(n3))
  + Cấu trúc dữ liệu heap và heapsort
    - Cấu trúc heap: là cây nhị phân gần đầy đủ (nearly complete binary tree). Biểu diễn cây nhị phân gần đầy đủ của cấu trúc heap với 1 mảng bằng cách đặt nút gốc tại vị trí 1, con của nó ở vị trí 2 và 3, các nút kế tiếp ở vị trí 4,5,6,… (Ứng với một nút tại vt j trong mảng thì ta có cha của nó sẽ ở vt = j div 2, hai con của nó sẽ ở vt 2j và 2j+1). Các tác vụ thêm, xóa, uphead đòi hỏi lgN so sánh còn thao tác downheap đòi hỏi 2lgN so sánh trên cấu trúc heap N phần tử;
    - Giải thuật heapsort (cấu trúc heap top-down): gồm 2 bước: (b1) tạo 1 cấu trúc heap chứa những phần tử cần sắp thứ tự. (b2) lần lượt xóa bỏ chúng ra khỏi heap theo một thứ tự. (Độ phức tạp ít hơn 3MlgM lần so sánh để sắp thứ tự M phần tử; thể hiện tinh thần biến đổi biểu diễn – representation change)
    - Cấu trúc heap bottom-up: phương pháp này coi các vị trí trong mảng như là nút gốc của các cấu trúc heap có kích thước nhỏ hơn để đảm bảo chúng thỏa mãn cấu trúc heap ( độ phức tạp là tuyến tính O(M) với cấu trúc heap M phần tử)
    - Kết luận rằng: cấu tạo cấu trúc heap theo kiểu dưới lên thì hữu hiệu hơn cấu tạo từ trên xuống (với độ phức tạp là MlgM)
  + Giải thuật Horner để định trị đa thức: thể hiện kỹ thuật “thay đổi biểu diễn”. Giải thuật horner tính lũy thừa nhị phân từ trái sang phải có độ phức tạp là O(log(n))
  + Giải thuật so trùng dòng ký tự RABIN-KARP: với văn bản là 1 mảng ký tự T[1..n] và khuôn mẫu là 1 mảng P[1..m], độ phức tạp là O((n-m+1)\*m), tuy nhiên thường chỉ có 1 vài bước dịch chuyển hợp lệ và do đó thời gian chạy bình thường là O(n+m)

**Chương 5: Quy hoạch động và giải thuật tham lam**

* **Quy hoạch động**
  + Quy hoạch động giải các bài toán tối ưu hóa bằng cách kết hợp các lời giải của các bài toán con của bài toán đang xét, phương pháp này khả dụng khi khi các bài toán con không độc lập với nhau, quy hoạch động giải các bài toán “cháu” dùng chung một lần rồi lưu lời giải trong 1 bảng và sau đó không phải tính lại. Quy hoạch động thường dùng giải các bài toán tối ưu hóa
  + 4 bước quy hoạch động: (b1) đặc trưng hóa cấu trúc của lời giải tối ưu. (b2) định nghĩa giá trị của lời giải tối ưu một cách đệ quy. (b3) tính trị của lời giải tối ưu theo kiểu từ dưới lên. (b4) cấu tạo lời giải tối ưu từ những thông tin đã được tính toán và lưu trữ
  + 2 thành phần của quy hoạch động: tính chất tiểu cấu trúc tối ưu (optimal substructure) và các bài toán con trùng lắp (overlapping subproblems)
  + Các bài toán liên quan: (1) Nhân chuỗi ma trận (tính chi phí tối ưu MATRIX-CHAIN-ORDER có độ phức tạp là O(n3), tạo lời giải tối ưu MATRIX-CHAIN-MULTIPLY); (2) Chuỗi con chung dài nhất, với đầu vào là 2 chuỗi có độ dài lần lượt là m,n (tìm lời giải LCS-LENGTH thì độ phức tạp là O(mn), in ra kết quả PRINT-LCS có độ phức tạp là O(m+n)); (3) Bài toán cái túi dạng 0-1, với cái túi có sức chứa tối đa là M và N mặt hàng thì độ phức tạp của giải thuật là O(MN); (4) Giải thuật Warshall tính bao đóng truyền của 1 đồ thị được biểu diễn bằng ma trận kế cận (độ phức tạp O(V3), với đồ thị có V đỉnh); (5) Giải thuật Floyd tìm các lối đi ngắn nhất dựa trên đồ thì có trọng số (có hướng hoặc không), độ phức tạp là O(V3)
* **Giải thuật tham lam**
  + Các giải thuật tối ưu hóa thường đi qua một số bước với tập các khả năng lựa chọn tại mỗi bước, một giải thuật tham lam thường chọn 1 khả năng được xem là tốt nhất tại lúc đó - tức là chọn 1 khả năng tối ưu cục bộ với hy vọng dẫn đến 1 lời giải tối ưu toàn cục
  + 2 thành phần chính của giải thuật tham lam: tính chất lựa chọn tham lam (greedy choice property) và tiểu cấu trúc tối ưu
  + Các bài toán liên quan: (1) Bài toán xếp lịch các hoạt động GREEDY-ACTIVITY-SELECTOR; (2) Bài toán cái túi dạng phân số GREEDY-KNAPSACK có độ phức tạp là O(n); (3) Bài toán nén file dùng mã Huffman (độ phức tạp là O(nlgn) trên tập n ký tự); (4) Cây bao trùm tối thiểu – giải thuật Prim, độ phức tạp khi biểu diễn độ thị bằng tập ds kế cận (hàng đợi có thứ tự ưu tiên Q bằng cấu trúc Heap – phải là min-heap) là O(ElgV), khi biểu diễn bằng ma trận là O(V2); (5) Giải thuật Dijkstra tìm lối đi ngắn nhất từ 1 đỉnh đến các đỉnh còn lại, độ phức tạp khi biểu diễn đồ thị bằng tập ds kế cận là O(V+E)lgV, khi biểu diễn bằng ma trận là O(V2+E); (6) Bài toán tô màu đồ thị SAME-COLOR, độ phức tạp là O(n2) với n là số đỉnh của đồ thị
  + Phân tích độ phức tạp của giải thuật Huffman: giả sử hàng đợi có thứ tự ưu tiên Q được hiện thực bằng cấu trúc Heap. Cho 1 tập C gồm n ký tự, việc khởi tạo Q được thực thi với thời gian O(n) nếu ta áp dụng cách tạo cấu trúc Heap từ dưới lên. Vòng lặp for được thực thi chính xác gồm n-1 lần và vì mỗi tác vụ làm việc trên Heap đòi hỏi O(lgn), vòng lặp này đóng góp chi phí O(nlgn) vào thời gian tính toán. Như vậy, độ phức tạp của giải thuật HUFFMAN trên tập n ký tự sẽ là O(nlgn)
* **So sánh giải thuật tham lam với quy hoạch động:** giải thuật quy hoạch động đem lại lời giải tối ưu, còn giải thuật tham lam không nhất thiết đem lại lời giải tối ưu

**Chương 6: Giải thuật quay lui**

* **Giải thuật quay lui**
  + Thiết kế giải thuật tìm lời giải cho 1 bài toán không phải là bám theo 1 tập quy luật tính toán đã xác định mà là bằng cách *thử và sửa sai (trial and error)*
  + Các thuật ngữ nhận biết: Thử và sửa sai; Chọn bước đi tới; Bế tắc “dead-end”; Quay lui; Đệ quy; Cây không gian trạng thái
  + Các bài toán liên quan: (1) Bài toán đường đi của quân mã; (2) Bài toán 8 quân hậu
  + Sự mở rộng của giải thuật quay lui đệ quy là tìm không chỉ 1 lời giải mà tất cả những lời giải của bài toán đã cho (phương pháp: một khi lời giải được tìm thấy và ghi lại, ta tiếp tục xét các ứng viên kế tiếp trong quá trình chọn ứng viên 1 cách có hệ thống), khuôn mẫu tổng quát cho sự mở rộng này còn gọi là giải thuật vét cạn
  + Cây không gian trạng thái được dùng để diễn tả quá trình làm việc của giải thuật quay lui; Thời gian tính toán của giải thuật quay lui là O(an) với a là số nút con mà 1 nút trên cây tìm kiếm có được và n là chiều dài của lối đi trên cây
* Giải thuật nhánh và cận
  + Các thuật ngữ nhận biết: Chi phí của lời giải tốt nhất cho đến bây giờ; Cận của lời giải chưa đầy đủ; Tỉa nhánh; Giải thuật quay lui vét cạn
  + Các bài toán liên quan: (1) Bài toán thương gia du hành; (2) Bài toán sinh ra các hoán vị; (3) Bài toán cái túi dạng 0-1

**Chương 7: Vấn đề NP – đầy đủ**

* + Một số bài toán NP đầy đủ: (1) Bài toán tìm đồ thị con đầy đủ lớn nhất trong 1 đồ thị; (2) Bài toán phân hoạch số; (3) Bài toán cái túi; (4) Bài toán quy hoạch nguyên; (5) Xếp lịch công việc trên đa bộ xử lý; (6) Bài toán phủ định; (7) Bài toán xếp thùng
  + Một số kỹ thuật đối phó với những bài toán NP-đầy đủ:
    - Dùng giải thuật xấp xỉ để tìm lời giải xấp xỉ tối ưu
    - Dựa vào hiệu quả của trường hợp trung bình để phát triển 1 giải thuật mà ta có thể dựa vào đó để tìm ra lời giải trong 1 số trường hợp
    - Nếu kích thước đầu vào nhỏ ta có thể dùng các giải thuật có độ phức tạp hàm mũ thì vẫn hữu hiệu, vd: giải thuật quay lui, nhánh và cận
    - Đưa ra các Heuristic vào giải thuật để tăng tính hữu hiệu
    - Sử dụng các metaheuristic chú giải thêm cho thuật ngữ này
  + Đóng góp của vấn đề NP-đầy đủ
    - Có nhiều bài toán NP-đầy đủ trong nhiều lĩnh vực: giải tích số, sắp thứ tự và tìm kiếm, xử lý dòng ký tự, xử lý đồ thị, xử lý hình ảnh,..
    - Nó cung cấp 1 cơ chế để xác định 1 bài toán mới trong các lĩnh vực trên là “dễ” hay “khó”
    - Có 4 lớp bài toán phân theo độ khó (giảm dần): Bài toán bất khả quyết (chưa hề có lời giải) -> Bài toán khó giải (chỉ tồn tại giải thuật thời gian hàm mũ để giải chúng) -> Bài toán NP-đầy đủ (là 1 lớp con đặc biệt của lớp bài toán NP) -> Bài toán P (tồn tại giải thuật có độ phức tạp đa thức để giải)