***1.Khái niệm thuật toán:***

* Một dãy hữu hạn các thao tác và trình tự thực hiện các thao tác đó sao cho sau khi thực hiện dãy thao tác này theo trình tự đã chỉ ra với đầu vào xác định ta thu được kết quả đầu ra mong muốn
* Một chương trình là một cách biểu diễn hình thức thuật toán bởi một ngôn ngữ lập trình cụ thể

**Các đặc trưng của thuật toán**

* Đầu vào và Đầu ra: Mỗi thuật toán được gắn với một bài toán cụ thể cần xác định đầu vào và đầu ra và điều kiện của chúng
* Tính đúng đắn: đảm bảo kết quả tính toán hay các thao tác mà máy tính thực hiện được là chính xác
* Tính xác định (rõ ràng): các bước thực hiện có trình tự xác định
* Tính dừng: phải cho ra kết quả sau một số hữu hạn bước
* Tính phổ dụng: áp dụng cho các bài toán cùng dạng
* Tính khách quan: cho kết quả như nhau khi chạy trên các máy tính khác nhau

**Biểu diễn thuật toán**

* Cách 1: Ngôn ngữ tự nhiên
* Cách 2: Ngôn ngữ lưu đồ (sơ đồ khối)
* Cách 3: Mã giả
* Cách 4: Các ngôn ngữ lập trình như Pascal, C/C++,Java, ... nhưng không nhất thiết phải sử dụng đúng ký pháp của ngôn ngữ đó

**Biểu diễn thuật toán bằng ngôn ngữ tự nhiên**

* Sử dụng ngôn ngữ thường ngày để liệt kê các bước của thuật toán
* Đặc điểm:

+ Không yêu cầu người viết thuật toán cũng như người đọc thuật toán phải nắm các quy tắc

+ Cách biểu diễn này thường dài dòng, không thể hiện rõ cấu trúc của thuật toán, đôi lúc gây hiểu lầm hoặc khó hiểu cho người đọc

+ Không có một quy tắc cố định nào trong việc thể hiện thuật toán bằng ngôn ngữ tự nhiên. Nên viết các bước con lùi vào bên phải và đánh số bước theo quy tắc phân cấp như 1., 1.1., 1.1.1.,...

**Sử dụng lưu đồ thuật toán**

– Thuật toán được mô tả dưới dạng sơ đồ khối

– Sử dụng các ký hiệu khác nhau để biểu thị chuỗi hoạt động, cần thiết để giải quyết một vấn đề

– Đóng vai trò như một bản thiết kế hoặc sơ đồ logic của giải pháp cho một vấn đề

**Sử dụng mã giả (pseudo code)**

* Khi thể hiện thuật toán bằng mã giả, ta sẽ vay mượn các cú pháp của một ngôn ngữ lập trình nào đó để thể hiện thuật toán
* Dùng mã giả vừa tận dụng được các khái niệm trong ngôn ngữ lập trình, vừa giúp người cài đặt dễ dàng nắm bắt nội dung thuật toán
* Trong mã giả vẫn có thể dùng một phần ngôn ngữ tự nhiên

**Phân tích thuật toán**

* **Độ phức tạp của thuật toán**

+ Độ phức tạp tính toán của thuật toán được xác định như là lượng tài nguyên các loại mà thuật toán đòi hỏi sử dụng

+ Có hai loại tài nguyên quan trọng đó là thời gian và bộ nhớ (không gian lưu trữ)

+ Việc tính chính xác được các loại tài nguyên mà thuật toán đòi hỏi là rất khó. Vì thế ta quan tâm đến việc đưa ra các đánh giá sát thực cho các đại lượng này

+Trong môn học này ta đặc biệt quan tâm đến đánh giá thời gian cần thiết để thực hiện thuật toán mà ta sẽ gọi là thời gian tính của thuật toán

* **Phép toán cơ bản (phép tính cơ bản)**

◦ Đo thời gian tính bằng đơn vị đo nào?

◦ Định nghĩa. Ta gọi phép toán cơ bản là phép toán có thể thực hiện với thời gian bị chặn bởi một hằng số không phụ thuộc vào kích thước dữ liệu đầu vào

◦ Để tính toán thời gian tính của thuật toán ta sẽ đếm số phép toán cơ bản mà nó phải thực hiện

◦ Ví dụ: trong phân tích thuật toán, ta thường coi các phép gán, cộng, trừ, nhân, chia, lũy thừa, ... là các phép toán cơ bản

* **Đếm số phép toán cơ bản**

◦ Nhận diện các phép toán cơ bản trong thuật toán

◦ Nếu có nhiều phép toán cơ bản thì xác định phép toán cơ bản T chiếm nhiều thời gian chạy nhất so với các phép toán cơ bản còn lại (ví dụ: i = i + 1 coi phép cộng chạy lâu hơn phép gán)

◦ Phép toán T thường xuất hiện trong các vòng lặp

◦ Đếm số lần thực hiện phép toán T, sẽ thu được hàm thời gian chạy f(n)

* **Các loại thời gian tính**

◦ Thời gian tối thiểu cần thiết để thực hiện thuật toán với mọi bộ dữ liệu đầu vào kích thước n. Thời gian như vậy sẽ được gọi là thời gian tính tốt nhất của thuật toán với đầu vào kích thước n

◦ Thời gian nhiều nhất cần thiết để thực hiện thuật toán với mọi bộ dữ liệu đầu vào kích thước n. Thời gian như vậy sẽ được gọi là thời gian tính tồi nhất của thuật toán với đầu vào kích thước n

◦ Thời gian trung bình cần thiết để thực hiện thuật toán trên tập hữu hạn các đầu vào kích thước n.

Thời gian như vậy sẽ được gọi là thời gian tính trung bình của thuật toán

***Ký hiệu tiệm cận***

◦ Các ký hiệu Q, O, W

◦ Được xác định đối với các hàm nhận giá trị nguyên không âm

◦ Dùng để so sánh tốc độ tăng của các hàm

◦ Được sử dụng để mô tả thời gian tính của thuật toán. Thay vì nói chính xác, ta có thể nói thời gian

tính là, chẳng hạn, Q(n^2))

***Ký hiệu Q***

Phân tích thuật toán

Q(g(n)) = {f(n) | tồn tại các hằng số c1,c2 và n0 sao cho 0<=c1g(n) <= f(n) <= c2g(n), với mọi n>=n0 }

Đối với hàm g(n) cho trước, ta ký hiệu Q(g(n)) là tập các hàm

Ta nói rằng g(n) là đánh giá tiệm cận đúng cho f(n)

**Ví dụ**

◦ 10n^2- 3n = Q(n^2)

◦ Với giá trị nào của các hằng số n0, c1, và c2 thì bất đẳng thức sau đây là đúng với n ≥ n0:

c1n^2 ≤ 10n2- 3n ≤ c2n^2

◦ Ta có thể lấy c1 bé hơn hệ số của số hạng với số mũ cao nhất, còn c2 lấy lớn hơn hệ số này, chẳng hạn: c1=9 < 10 < c2 = 11, n0 = 10.

◦ Tổng quát, để so sánh tốc độ tăng của các đa thức, cần nhìn vào số hạng với số mũ cao nhất

***Kí hiệu O lớn***

Đối với hàm g(n) cho trước, ta ký hiệu O(g(n)) là tập các hàm

O(g(n)) = {f(n) | tồn tại các hằng số dương c và n0 sao cho:

f(n) <= c\*g(n) với mọi n>=n0 }

***Ví dụ kí hiệu O lớn***

◦ Chứng minh: f(n) = n^2 + 2n + 1 là O(n^2)

 Cần chỉ ra: n^2 + 2n + 1 ≤ c\*n^2

với c là hằng số nào đó và khi n > n0 nào đó

◦ Ta có:2n^2 ≥ 2n khi n ≥ 1 và n^2 ≥ 1 khi n ≥ 1

◦ Vì vậy n^2 + 2n + 1 ≤ 4\*n^2 với mọi n ≥ 1

◦ Như vậy hằng số c = 4, và n0 = 1

***Ví dụ kí hiệu O lớn***

◦ Ta thấy rằng: Nếu f(n) là O(n^2) thì nó cũng là O(n^k) với k > 2.

◦ Chứng minh: f(n) = n^2 + 2n + 1 thông thuộc O(n).

◦ Phản chứng. Giả sử trái lại, khi đó phải tìm được hằng số c và số n0 để cho:

N^2 + 2n + 1 ≤ c\*n khi n ≥ n0

◦ Suy ra n^2 < n2 + 2n + 1 ≤ c\*n với mọi n ≥ n0

◦ Từ đó ta thu được: n < c với mọi n ≥ n0 ?!

***Kí hiệu W***

Phân tích thuật toán

Đối với hàm g(n) cho trước, ta ký hiệu W(g(n)) là tập các hàm:

W(g(n)) = {f(n)| tồn tại các hằng số dương c và n0 sao cho cg(n) <= f(n) với mọi n >= n0 }

Ta nói g(n) là cận dưới tiệm cận của f(n)

Ví dụ kí hiệu W

◦ Chứng minh: f(n) = n^2- 2000n là W(n^2)

 Cần chỉ ra: n^2^- 2000n >= c\*n^2 với c là hằng số nào đó và khi n > n0 nào đó

◦ Ta có: n^2- 2000n >= 0.5\*n^2 với mọi n ≥ 10000

(vì n^2- 2000n - 0.5\*n^2 = 0.5\*n^2- 2000n = n(0.5\*n -2000) >= 0 khi n ≥ 10000)

◦ Như vậy hằng số c = 1, và n0 = 10000

***Cách nói về thời gian tính***

◦ Nói “Thời gian tính là O(f(n))” hiểu là: Đánh giá trong tình huống tồi nhất (worst case) là O(f(n)). Thường nói: “Đánh giá thời gian tính trong tình huống tồi nhất là O(f(n))”

 Nghĩa là thời gian tính trong tình huống tồi nhất được xác định bởi một hàm nào đó g(n) thuộc O(f(n))

◦ “Thời gian tính là W(f(n))” hiểu là: Đánh giá trong tình huống tốt nhất (best case) là W(f(n)). Thường nói: “Đánh giá thời gian tính trong tình huống tốt nhất là W(f(n))”

 Nghĩa là thời gian tính trong tình huống tốt nhất được xác định bởi một hàm nào đó g(n) thuộc W(f(n))

**Đánh giá độ phức tạp thuật toán**

* Phân tích các chương trình đệ quy

◦ Với các chương trình đệ quy có gọi các chương trình đệ quy con, ta không thể áp dụng cách tính như trên được vì một chương trình đệ quy sẽ gọi chính bản thân nó

◦ Cần lập thành các phương trình đệ quy, sau đó giải phương trình đệ quy, nghiệm của phương trình đệ quy chính là thời gian thực hiện của chương trình đệ quy đó

* Thành lập phương trình đệ quy

◦ Phương trình đệ quy là một phương trình biểu diễn mối liên hệ giữa T(n) và T(k) T(n) là thời gian thực hiện chương trình với kích thước dữ liệu đầu vào là n

T(k) là thời gian thực hiện chương trình với kích thước dữ liệu đầu vào là k(k <n)

◦ Để thành lập phương trình đệ quy ta phải căn cứ vào chương trình đệ quy

* Giải phương trình đệ quy

◦ Có ba phương pháp giải phương trình đệ quy

+Phương pháp truy hồi

Dùng đệ quy để thay thế bất kỳ T(m) với m < n vào phía phải của phương trình cho đến khi tất cả T(m) với m>1 được thay thế bởi biểu thức của các T(1)

Vì T(1) luôn là hằng nên chúng ta có công thức của T(n) chứa các số hạng chỉ liên quan đến n và các hằng số

+Phương pháp đoán nghiệm

Đoán một nghiệm f(n) và dùng chứng minh quy nạp để chứng tỏ rằng T(n) ≤ f(n) với mọi n f(n) là một trong các hàm quen thuộc như logn, n, nlogn, n^2, n^3, 2n, n!, n^nn

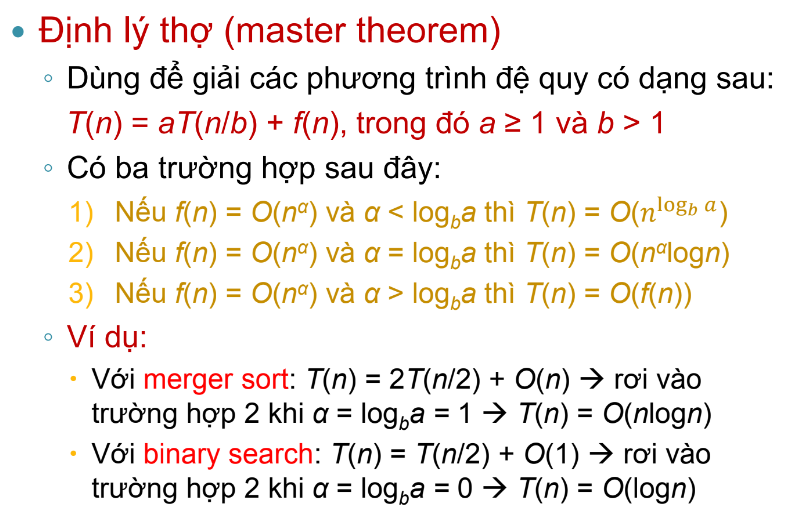
Đôi khi ta chỉ đoán dạng của f(n) trong đó có một vài tham số chưa xác định (ví dụ, f(n) = an^2 với a chưa xác định) và trong quá trình chứng minh quy nạp ta suy diễn ra giá trị thích hợp của các tham số

+Lời giải tổng quát của một lớp phương trình đệ quy

Để giải bài toán kích thước n, ta chia thành a bài toán con và mỗi bài toán con có kích thước n/b. Giải các bài toán con này và tổng hợp kết quả để được kết quả của

bài toán ban đầu Đối với các bài toán con, ta cũng làm tương tự một chương trình đệ quy

**Định lý thợ (master theorem)**



***2.Phân tích thiết kế thuật toán lặp và đệ quy***

*Các yếu tố phân tích, thiết kế thuật toán*

Muốn tìm thuật toán để giải quyết bài toán thì

◦ Phân tích thuật toán

◦ Chọn cấu trúc dữ liệu phù hợp

◦ Thiết kế thuật toán

◦ Chứng minh tính đúng đắn của thuật toán

◦ Chọn ngôn ngữ lập trình

◦ Cài đặt thuật toán để tạo ra chương trình

◦ Kiểm thử và gỡ lỗi logic để hoàn thiện chương trình

**Phân tích thuật toán**

◦ Khi phân tích, thiết kế thuật toán ta phân tích, thiết kế các yếu tố:

 Đặc tả bài toán

 Cách thức tiến hành (ý tưởng giải quyết bài toán)

 Tính đúng đắn của thuật toán

 Truy hồi và đệ quy

 Độ phức tạp của thuật toán

**Đặc tả bài toán**

◦ Phải hiểu được bài toán yêu cầu làm gì

◦ Xác định các điều kiện và miền ràng buộc của dữ liệu vào (tiền điều kiện) và dữ liệu ra (hậu điều kiện)

◦ Hiểu được các ví dụ mẫu nếu có

**Cách thức tiến hành**

◦ Đặc tả bài toán  những ý tưởng khác nhau để giải quyết bài toán  lựa chọn ý tưởng phù hợp nhất

cho bài toán đặt ra  mô hình hóa bởi sơ đồ thuật giải hoặc giả mã, thậm chí mã thật cho thuật toán

**Tính đúng đắn của thuật toán**

◦ Tính đúng đắn của thuật toán lặp được chứng minh thông qua việc xác định bất biến vòng lặp, đó là một mệnh đề logic đúng trước và sau mỗi bước lặp

 Thiết lập bất biến trước khi vào vòng lặp;

 Duy trì bất biến trong từng bước trong vòng lặp;

 Đảm bảo bất biến đúng khi ra khỏi vòng lặp.

* ***Chú ý:*** trong khi thực hiện vòng lặp đôi khi có thể có những tình huống thoát bất thường khỏi vòng lặp do tìm thấy kết quả hoặc có những bất thường không tìm được kết quả đối với những dữ liệu đầu vào đặc biệt nào đó.

***Truy hồi và đệ quy***

◦ Khi xác định được bất biến vòng lặp ta xây dựng biểu thức toán học truy hồi và chương trình đệ quy

tương ứng để cài đặt thuật toán

***Độ phức tạp của thuật toán***

◦ Độ phức tạp thuật toán lặp và đệ quy sẽ cho chúng ta thấy được tốc độ tính toán và so sánh tốt xấu so với các thuật toán khác

Ví dụ: Nhập vào số nguyên không âm n và tính n!

◦ ***Cách thức tiến hành (ý tưởng thuật toán)***

 Bước 1. Nhập vào giá trị nguyên không âm n.

 Bước 2: Khởi gán S[0] = 1.

 Bước 3. Thực hiện vòng lặp i lần lượt từ 1 đến n với

mỗi bước lặp ta tính S[i] = S[i-1] \* i.

 Bước 4. Xuất ra giá trị S[n] là kết quả cần tính.

Để tối ưu độ phức tạp không gian của thuật toán, thay vì dùng mảng S[n+1] thì ta dùng một biến S

Độ phức tạp của thuật toán trên

◦ Độ phức tạp của thuật toán lặp là O(n) do chỉ có một

vòng lặp n bước

◦ Độ phức tạp của thuật toán đệ quy cũng là O(n)

***3.Phương pháp phân tích thiết kế chia để trị***

Áp dụng cho các bài toán có thể giải quyết bằng cách:

◦ Chia nhỏ ra thành các bài toán con

◦ Giải quyết các bài toán con

◦ Lời giải của các bài toán con được tổng hợp lại thành lời giải cho bài toán ban đầu

* Phương pháp chia để trị:

◦ Bước 1: Chia/Tách nhỏ

Chia bài toán ban đầu thành các bài toán con đồng dạng với bài toán ban đầu

◦ Bước 2: Trị/Giải quyết bài toán con

Giải các bài toán con các lời giải con

◦ Bước 3: Kết hợp lời giải

Tổng hợp các lời giải con lời giải của bài toán ban đầu

***Lưu ý:***

- Các bài toán con lại được chia thành các bài toán nhỏ hơn nữa

- Dừng lại khi kích thước bài toán đủ nhỏ mà ta có thể giải dễ dàng bài toán cơ sở

* Giảm để trị : Trường hợp đặc biệt của chia để trị

Quy về đúng một bài toán nhỏ hơn

Có ba chiến lược

◦ Giảm bởi một hằng số

◦ Giảm bởi một hệ số

◦ Giảm kích thước của biến/Chia cắt

Tìm điểm chia cắt

Xử lý tùy theo điều kiện (vd: >, <, =)

Lưu ý:

- Dừng chia cắt khi không thể chia cắt được nữa

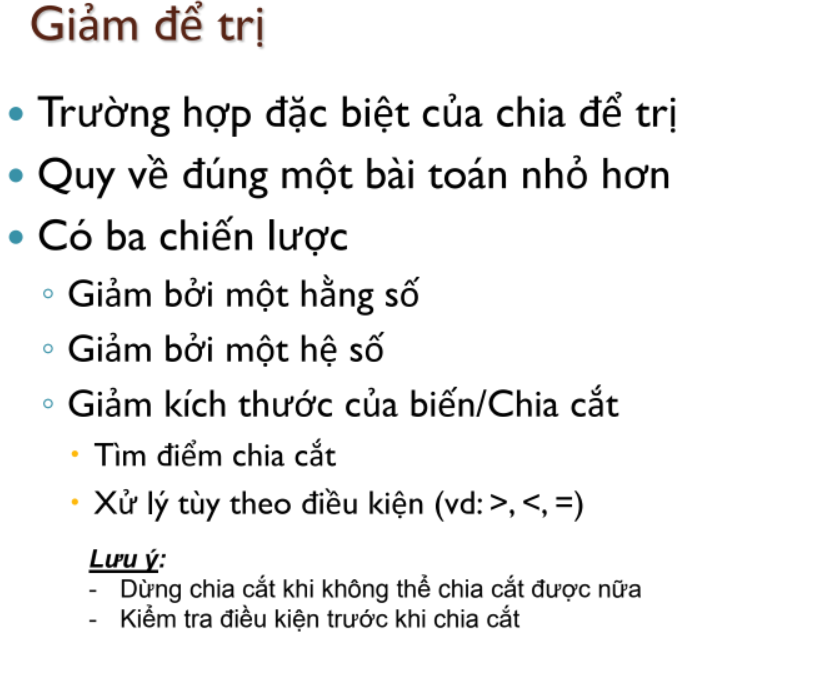
- Kiểm tra điều kiện trước khi chia cắt

***Ưu điểm:***

* Giảm độ phức tạp về thời gian đối với nhiều bài toán
* Chia để trị là một trong số các kĩ thuật mạnh mẽ để thiết kế thuật toán

***Nhược điểm:***

* Sử dụng đệ quy nên tốn bộ nhớ và cần kỹ thuật xử lý
* Không phải là giải pháp để giải quyết mọi bài toán



***4. Phương pháp quay lui***

**Ý tưởng**

◦ Kỹ thuật thiết kế thuật toán dựa trên đệ quy

◦ Tìm lời giải từng bước, tại mỗi bước

+ Nếu có một lựa chọn chấp nhận được , ghi nhận và tiến hành thử bước tiếp theo

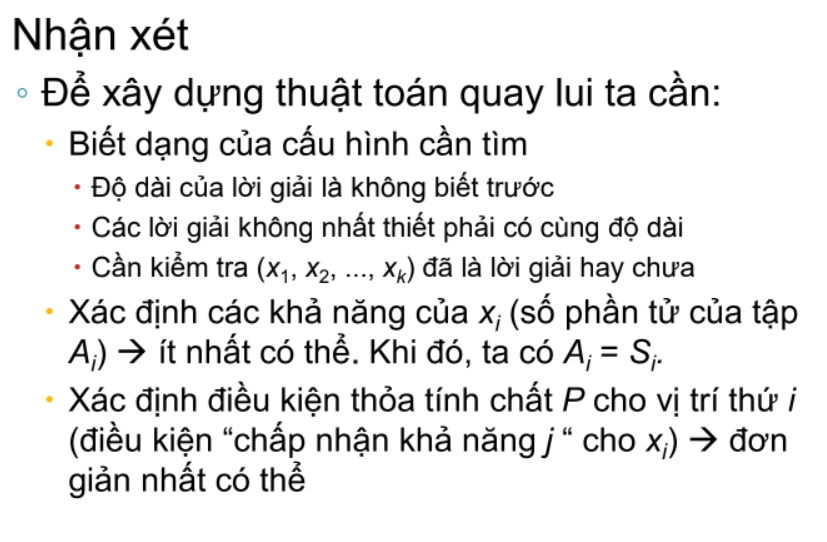
◦ Nếu không có lựa chọn nào khả thi

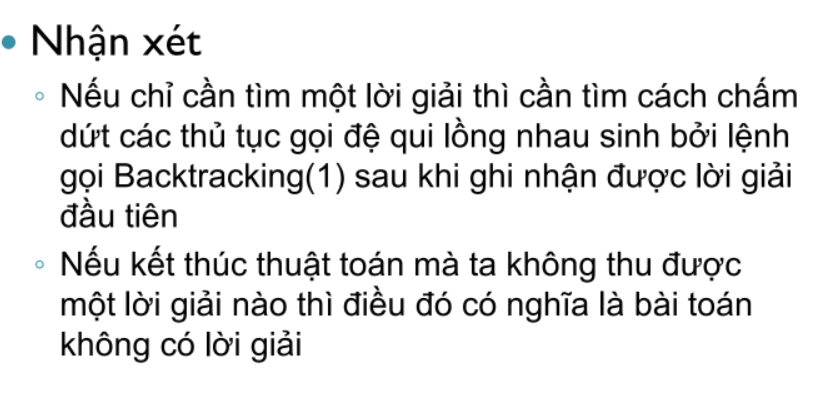
Quay lại bước trước

Xóa bỏ ghi nhận

Thử các lựa chọn còn lại tại bước này

◦ Phù hợp với bài toán liệt kê cấu hình dạng X[1, ..., n]





***5.Phương pháp tham lam***

**Mục đích:** Tìm một lời giải tốt trong thời gian chấp nhận được (độ phức tạp đa thức)

**Ý tưởng:** Dựa vào sự đánh giá tối ưu cục bộ địa phương (local optimum) để đưa ra quyết định tức thì tại mỗi bước lựa chọn, với hy vọng cuối cùng sẽ tìm ra được phương án tối ưu toàn cục (global optimum).

** Kỹ thuật**

◦ Sắp xếp các lựa chọn cho bước đó theo thứ tự nào đó “có lợi” (tăng dần hoặc giảm dần)

◦ Chọn lựa chọn tốt nhất rồi đi tiếp bước kế (không quay lui)

** Đặc điểm**

◦ Có trường hợp luôn tìm ra phương án tối ưu

◦ Trường hợp không tìm ra phương án tối ưu thì thu được một phương án khả dĩ chấp nhận được

◦ Thường có tốc độ tốt hơn hẳn so với các thuật toán tối ưu tổng thể.

***6.Phương pháp phân tích thiết kế quy hoạch động***

**Một số thuật ngữ**

◦ Bài toán giải theo phương pháp quy hoạch động được gọi là bài toán quy hoạch động

◦ Công thức phối hợp nghiệm của các bài toán con để có nghiệm của bài toán lớn gọi là công thức truy hồi của quy hoạch động

◦ Tập các bài toán nhỏ nhất có ngay lời giải để từ đó giải quyết các bài toán lớn hơn gọi là cơ sở của quy hoạch động

◦ Không gian lưu trữ lời giải các bài toán con để tìm cách phối hợp chúng gọi là bảng quy hoạch hay bảng phương án của quy hoạch động

**Ba tính chất**

◦ Tính chất thứ nhất và thứ hai là điều kiện cần của một bài toán quy hoạch động.

◦ Tính chất thứ ba nêu lên đặc điểm của một bài toán mà cách giải bằng phương pháp quy hoạch động hiệu quả hơn hẳn so với phương pháp giải đệ quy thông thường

◦ Khác với thuật toán đệ quy, phương pháp quy hoạch động thêm vào cơ chế lưu trữ nghiệm hay một phần nghiệm của mỗi bài toán khi giải xong nhằm mục đích sử dụng lại, hạn chế những thao tác thừa trong quá trình tính toán

**Mục đích**: Cải tiến thuật toán chia để trị hoặc quay lui vét cạn để giảm thời gian thực hiện

**Ý tưởng**

◦ Lưu trữ các kết quả của các bài toán con trong bảng quy hoạch

◦ Đổi bộ nhớ lấy thời gian

**Kỹ thuật thiết kế**

◦ Phân tích bài toán dùng kỹ thuật chia để trị/quay lui

Chia bài toán thành các bài toán con

Xác định cơ sở của quy hoạch động

Tìm quan hệ giữa kết quả của bài toán lớn và kết quả của các bài toán con -> công thức truy hồi

◦ Lập bảng quy hoạch: Quy hoạch động = Chia để trị + Cơ chế lưu trữ nghiệm để sử dụng lại

**Lập bảng quy hoạch**

◦ Số chiều = số biến trong công thức truy hồi

◦ Xuất phát từ cơ sở của quy hoạch động

->Thiết lập quy tắc điền kết quả vào bảng quy hoạch theo công thức truy hồi

◦ Tra bảng tìm kết quả

◦ Lần vết trên bảng để tìm lời giải tối ưu

