**Tài liệu ôn tập Phân tích thiết kế thuật toán**

1. **Các khái niệm liên quan**

**Thuật toán là gì?**- Thuật toán là một dãy hữu hạn các bước không mập mờ và có thể thực thi được, quá trình hành động theo các bước này phải dừng và cho kết quả mong muốn.  
- Thỏa mãn 3 tiêu chuẩn: xác định, hữu hạn, đúng

**Ý nghĩa độ phức tạp**- Độ phức tạp của thuật toán là một khái niệm để đo lường lượng tài nguyên (thời gian và bộ nhớ) cần thiết để thực hiện thuật toán:

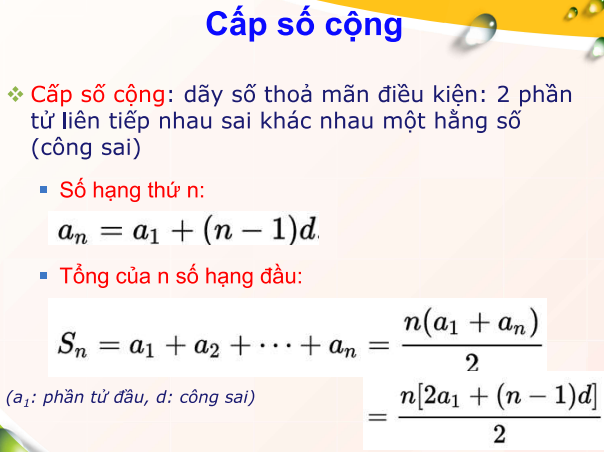
+ Độ phức tạp thời gian: các thuật toán cần một lượng thời gian hữu hạn để thực thi, thời gian mà thuật toán cần để giải bài toán đã cho gọi là độ phức tạp thời gian của thuật toán  
 + Độ phức tạp bộ nhớ: việc giải quyết các bài toán sử dụng máy tính yêu cầu bộ nhớ để lưu dữ liệu tạm thời hoặc kết quả cuối cùng trong khi thuật toán đang thực thi. Dung lượng bộ nhớ được yêu cầu bởi thuật toán để giải quyết bài toán được gọi là độ phức tạp bộ nhớ.

**Bậc tăng trưởng (Order of growth)**Để so sánh 2 thuật toán, cách tốt nhất là xác định thời gian thực hiện T(n) của 2 giải thuật rồi so sánh chúng. Tuy nhiên, thực tế, rất khó đế tính T(n) một cách chính xác, và nếu có tính được rồi thì có thể gặp khó khăn khác khi so sánh 2 hàm số với nhau 🡪 Vì vậy lựa chọn so sánh độ lớn của T1, T2 thay vì chính bản thân của chúng:  
- So sánh tương đối: hàm không sai biệt nhiều thì xem như xấp xỉ nhau về độ lớn.   
- Chỉ quan tâm đến những giá trị n đủ lớn   
- Khi n càng lớn (tiến tới vô cùng) thì hàm nào sẽ lớn hơn 🡪 Phân chia thành các bậc tăng trưởng  
Cách đo độ tăng của hàm số: hàm có bậc tăng tưởng lớn hơn 🡪 tăng nhanh hơn

**Các ký hiệu tiệm cận  
- Big(O):** biểu diễn giới hạn trên của độ phức tạp thuật toán, nó cho biết thời gian chạy của thuật toán trong trường hợp tệ nhất đối với một input có kích thước n. Dễ dàng tìm thấy giới hạn trên của một thuật toán. Giới hạn trên.  
- **Big(Ω):** biểu diễn giới hạn dưới của thuật toán, nó cho biết thời gian chạy tốt nhất có thể của thuật toán đối với một input có kích thước n. Tuy nhiên nó lại không thực sự hữu ích và ít được sử dụng nhất trong cả 3. Giới hạn dưới.  
**- Big(Ꙫ):** biểu diễn giới hạn một hàm từ trên xuống dưới, xác định hành vi tiệm cận chính xác. Nó cho biết thời gian chạy trung bình của thuật toán đối với một input có kích thước n.

1. **Kỹ thuật đếm sơ cấp**

**Một số công thức cần nhớ**



A picture containing text, screenshot, font, line

Description automatically generated

A picture containing text, font, handwriting, screenshot

Description automatically generated

A picture containing text, font, screenshot, line

Description automatically generated  
A picture containing text, font, receipt, white

Description automatically generated

**Trình bày mẫu bài toán đếm số phép toán sơ cấp**

**A picture containing text, screenshot, font, diagram

Description automatically generated**

A screenshot of a computer

Description automatically generated with low confidence  
A picture containing text, screenshot, font, diagram

Description automatically generated

1. **Phân tích thuật toán đệ quy**

**Thành lập phương trình đệ quy**

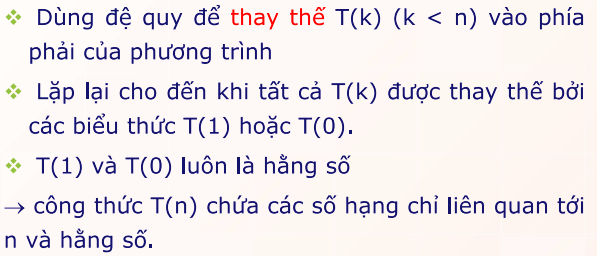
****

**A picture containing text, screenshot, font

Description automatically generated**

**Giải phương trình đệ quy**

* *Phương pháp truy hồi/thay thế*



A picture containing text, handwriting, font, screenshot

Description automatically generated

A picture containing text, font, handwriting, screenshot

Description automatically generated

* *Giải bằng Phương trình đặc trưng*

A picture containing text, font, handwriting, screenshot

Description automatically generated

A picture containing text, screenshot, font

Description automatically generated

A picture containing text, font, screenshot

Description automatically generated

Ví dụ

A picture containing text, screenshot, font

Description automatically generated

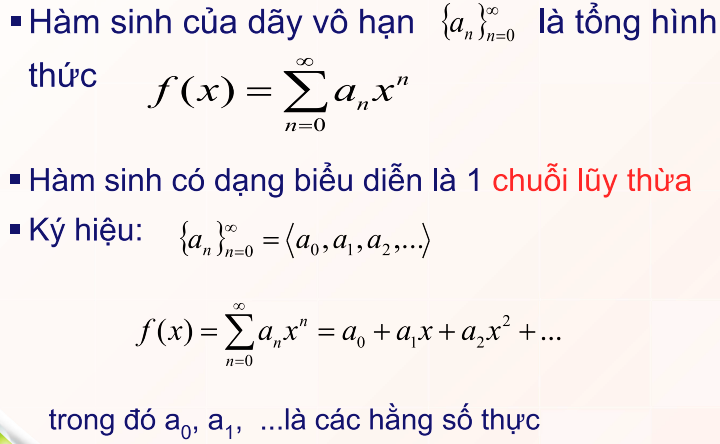
A picture containing text, font, screenshot

Description automatically generated

A picture containing text, font, handwriting, screenshot

Description automatically generated

* Phương pháp Hàm sinh (Generating function)

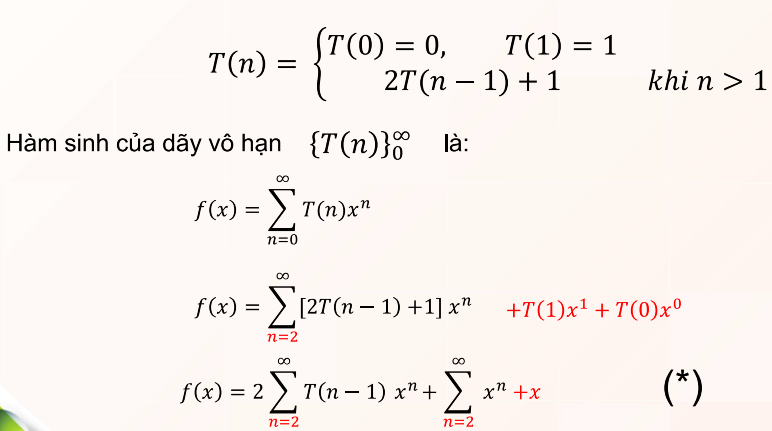


A picture containing text, font, handwriting, diagram

Description automatically generated

A picture containing text, font, handwriting, typography

Description automatically generated

  
A picture containing text, screenshot, font, diagram

Description automatically generated  
A picture containing text, font, screenshot, handwriting

Description automatically generated

1. **Độ phức tạp thuật toán**

**Big-O**

Ta nói T(n) có bậc tăng trưởng là f(n), T(n) có độ phức tạp của O là f(n). Định nghĩa:   
A picture containing text, font, handwriting, screenshot

Description automatically generated

* Kí hiệu: T(n) = O(f(n)). Bản chất T(n) ϵ O(f(n))
* Chú ý:   
  + Dấu = chỉ là ký hiệu hình thức  
  + O(f(n)) là tập hợp

A picture containing text, font, handwriting, calligraphy

Description automatically generated

A picture containing text, font, screenshot, number

Description automatically generated

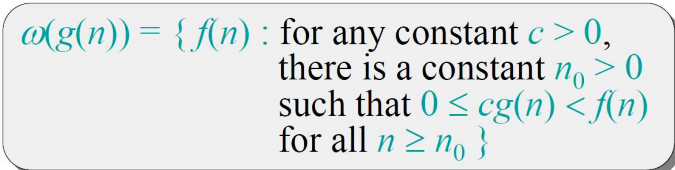
**Big-Ω**

Định nghĩa  
A picture containing text, font, handwriting, screenshot

Description automatically generated

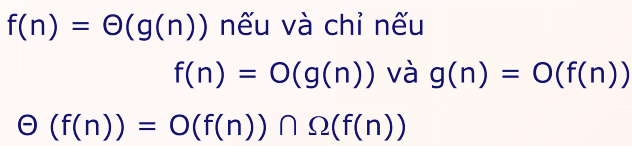
A screenshot of a computer

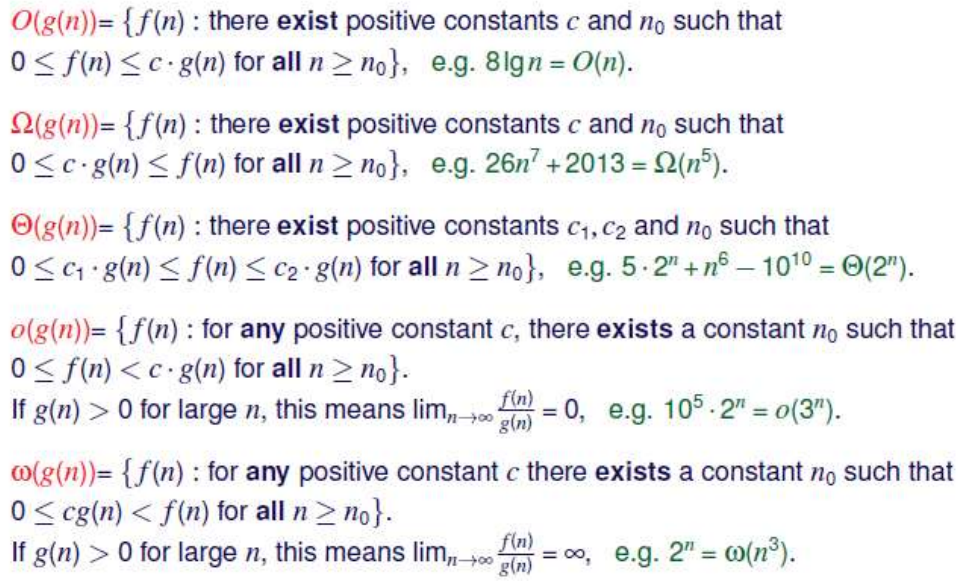
Description automatically generated with low confidence



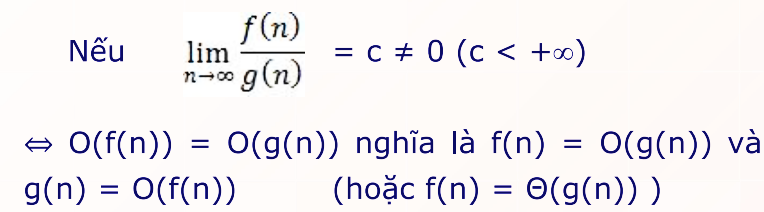
**Big-Ꙫ**

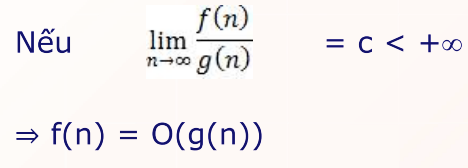
Định nghĩa  

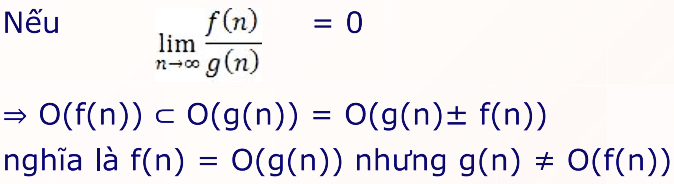



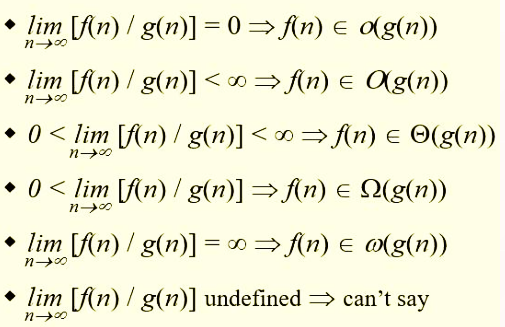
**Tóm tắt  
**

**Dùng lim để suy ra quan hệ**

****

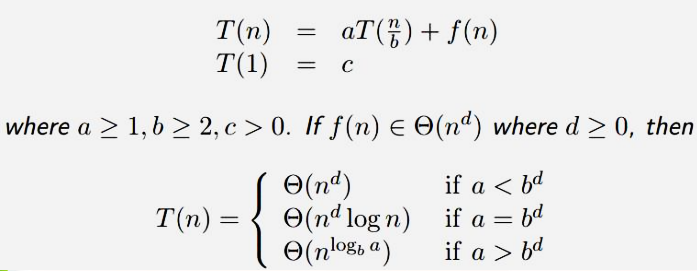
****

****

****

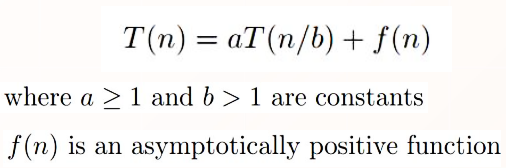
**Định lý Master**

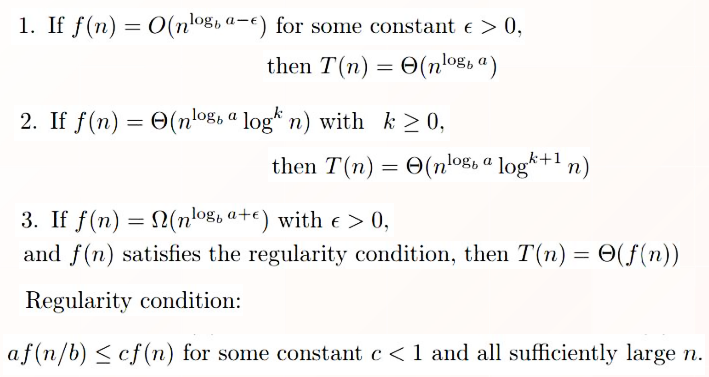
* Dạng đơn giản



Nhược điểm:   
 + Chỉ áp dụng cho phương trình đệ quy có dạng như trên   
 + Không thể dùng định lý Master nếu:   
 - T(n) không đơn điệu ( VD: sin(x) )   
 - f(n) không phải hàm đa thức ( VD: T(n) = 2T(n/2) + 2^n )  
 - b không thể biểu diễn như một hằng số

* Dạng tổng quát





1. **Thiết kế thuật toán**

**Bài toán tối ưu tổ hợp**

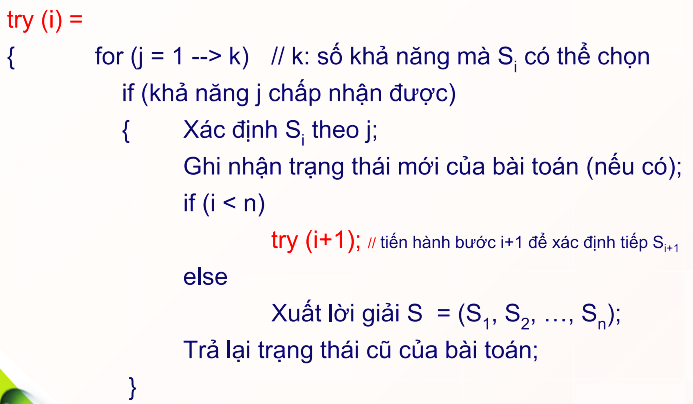
Có dạng tổng quát:  
 Cho hàm f(X) là hàm mục tiêu xác định trên 1 tập hữu hạn các phần tử D (tập các phương   
án)  
 Mỗi X ϵ D có dạng X = (X1, X2,…,Xn) gọi là 1 phương án, Xi ϵ P (tập các biến)   
 Cần tìm 1 phương án X “chấp nhận được” (thỏa mọi ràng buộc) sao cho f(X) đạt min (max) 🡪 phương án tối ưu

Cách giải quyết  
 - Vét cạn: xét tất cả các khả năng có thể có cho đến khi gặp giải pháp cho vấn đề cần giải quyết 🡪 thời gian mũ rất lớn  
 - Các thuật toán Quy hoạch tuyến tính  
 - Tối ưu cục bộ: phương pháp tham lam

**Phương pháp Backtracking**

Đặc trưng:  
 - Vét cạn  
 - Lời giải: được cấu thành bởi nhiều bước giải  
 - Quá trình tìm lời giải:  
 + Xây dựng từng bước giải thành phần  
 + Hoàn toàn dựa trên các phép thử

Mô hình hóa:  
- Ý tưởng:   
 Lời giải của bài toán là một bộ S = <S1, S2, S3,…,Sn> phải thỏa mãn điều kiện nào đó   
 Tại bước thứ i = xây dựng bước giải Si:  
 + Đã xây dựng xong các thành phần S1,…, S(i-1)  
 + Xây dựng Si bằng cách lần lượt thử tất cả các khả năng mà Si có thể chọn

Mô hình  


**Phương pháp Tham lam**

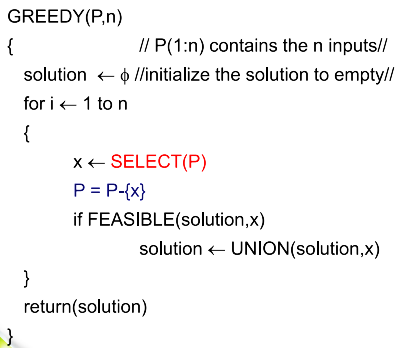
Lấy tiêu chuẩn tối ưu (trên phạm vi toàn cục) của bài toán, dựa vào đó chọn lựa hành động tốt nhất trong từng bước (hay từng giai đoạn) trong quá trình tìm kiếm lời giải

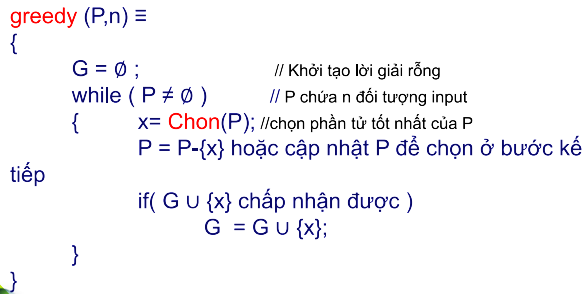
Mô hình hóa:  
 - Thường được vận dụng để giải bài toán tối ưu tổ hợp  
 - Xây dựng 1 cách giải phương án “tối ưu” X bằng cách:   
 + Chọn lựa từng thành phần Xi của X cho đến khi đủ n thành phần (hoàn thành)  
 + Với mỗi Xi, chọn Xi tối ưu  
 - Lời giải tối ưu (toàn cục) = tập hợp các lời giải tối ưu cục bộ  
 - Từng bước tói ưu cục bộ, hi vọng sẽ tối ưu toàn cục

Dấu hiệu nhận biết:   
 - Bài toán con tối ưu (Optimal Substructure)  
 - Thiết kế tiêu chuẩn tối ưu cục bộ (Greedy choice property = thiết kế tiêu chuẩn tối ưu cục bộ)

Ưu điểm:  
 - Đơn giản, dễ cài đặt  
 - Tốc độ nhanh (thời gian đa thức)

Khuyết điểm:  
 - Chưa chắc cho lời giải chính xác (cho phương án tốt chứ chưa chắc là tối ưu)  
 - Không phải luôn chấp nhận được (có thể cho lời giải tệ)  
 - Khó chứng minh tính đúng, nếu chứng minh được thì nó là thuật toán

Ý tưởng  


Mô hình  


**Thuật giải Heuristic**

Thuật toán là giải pháp viết dưới dạng thủ tục (các bước giải quyết vấn đề được diễn đạt một cách cụ thể) và thỏa 3 tiêu chuẩn: xác định (không mập mờ và có thể thực thi), hữu hạn (dừng), đúng (kết quả chính xác).

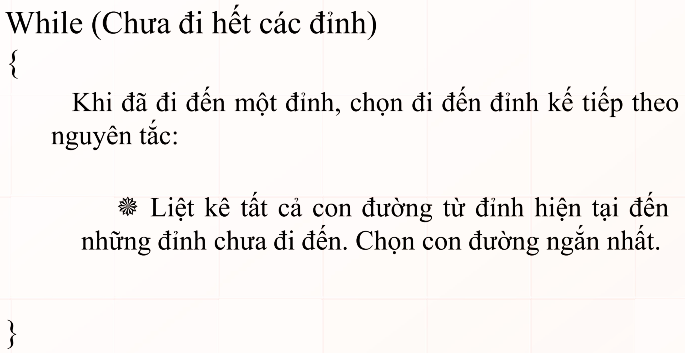
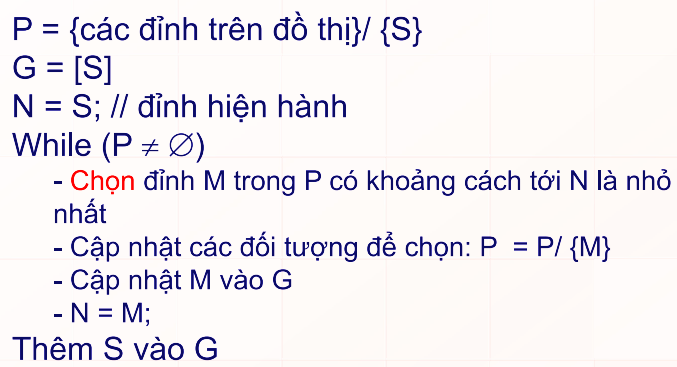
Thuật giải là giải pháp được diễn đạt tương tự như thuật toán nhưng không đòi hỏi các tiêu chuẩn như thuật toán. Tính xác định (giảm nhẹ tính tường minh đối với đệ quy và ngẫu nhiên) và tính đúng (chấp nhện các thuật giải đơn giản có thể cho kết quả đúng hay gần đúng nhưng có khả năng thành công cao hơn.

* Thuật giải Heuristic

Để có thể được chấp nhận, thuật giải phải thể hiện một giải pháp hợp lý nhất có thể trong tình huống hiện tại bằng cách:  
+ Tận dụng mọi thông tin hữu ích  
+ Sử dụng tri thức, kinh nghiệm, cảm tính trực giác của con người  
+ Tự nhiên đơn giản nhưng cho kết quả chấp nhận được

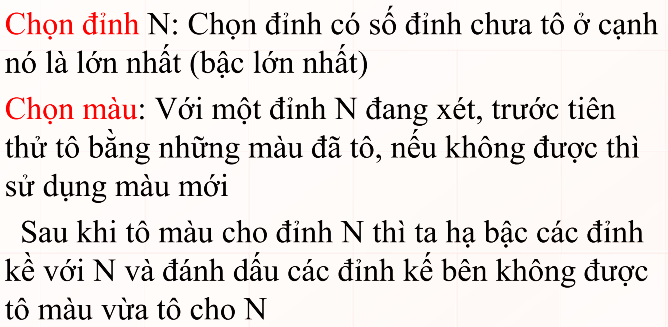
* Các nguyên lý của thuật giải Heuristic   
  + Vét cạn thông minh  
  + Nguyên lý thứ tự  
  + Nguyên lý tham lam   
  + Hàm Heuristic

*Ví dụ về cách giải bài toán tìm đường đi ngắn nhất cho người đưa thư sử dụng thuật giải Heuristic*

* Ý tưởng  
  
* Mã giả   
  

Thuật giải Heuristic cho bài toán này có độ phức tạp O(n2)

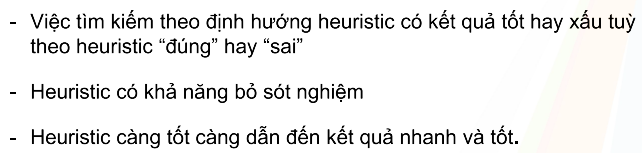
*Ví dụ về cách giải bài toán tô màu sử dụng thuật giải Heuristic*

* Ý tưởng  
  

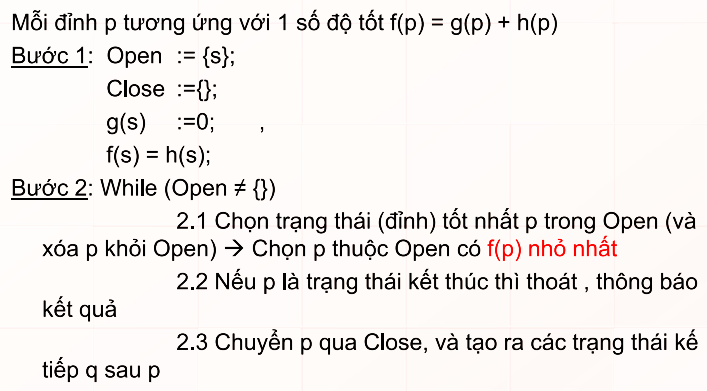
*Cấu trúc chung của bài toán tìm kiếm*

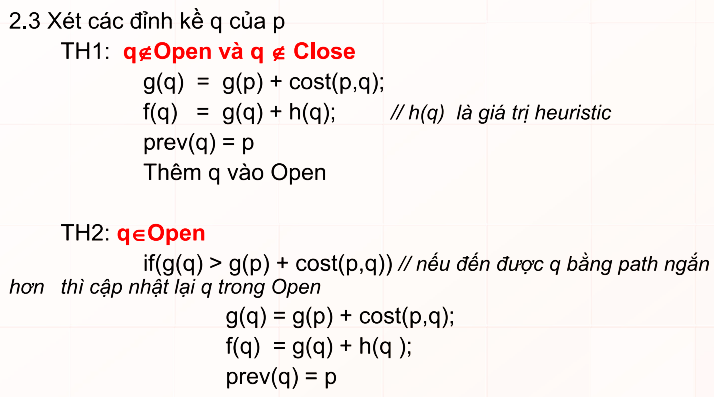


*Tìm kiếm Heuristic*

* Tìm theo Heuristic: Heuristic định hướng quá trình tìm kiếm theo hướng mà nó cho rằng khả năng đạt tới mục tiêu là cao nhất. Không sâu mà cũng không rộng.
* Kết quả của việc tìm kiếm với Heuristic   
  
* Hàm Heuristic (hàm đánh giá)  
  + f(u): đánh giá “sự gần đích” của trạng thái u (khoảng cách từ trạng thái u đến trạng thái đích)  
  + Tại mỗi bước tìm kiếm, chọn trạng thái để phát triển là trạng thái tốt nhất = có giá trị đánh giá nhỏ nhất (có nhiều hứa hẹn nhất tới đích)   
   Độ tốt f của một trạng thái được tính dựa theo 2 giá trị:  
   + h: một ước lượng về chi phí từ trạng thái hiện hành cho đến đích  
   + g: “chiều dài quãng đường” đã đi từ trạng thái ban đầu cho đến trạng thái hiện tại (chi phí thực sự)  
   Quy ước g và h đều không âm và càng nhỏ nghĩa là càng tốt

*Ví dụ với thuật giải A\**



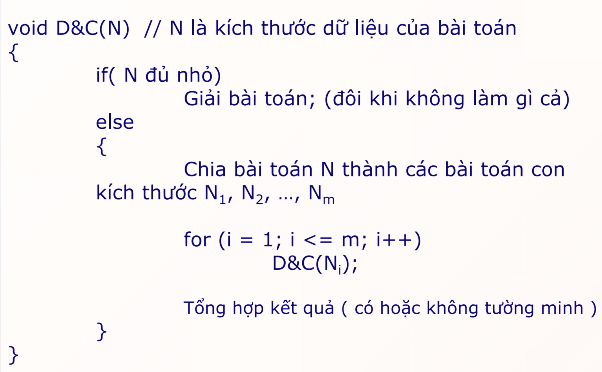


A screenshot of a computer

Description automatically generated with low confidence

**Chia để trị - Divide and Conquer**

* Ý tưởng:  
  + Để giải 1 bài toán có kích thước n, chia bài toán đã cho thành 1 số bài toán con có kích thước nhỏ hơn  
  + Giải các bài toán con rồi tổng hợp kết quả lại để được lời giải của bài toán ban đầu  
   ~ Giải các bài toán con: lại chia kích thước nhỏ hơn nữa  
   ~ Quá trình dẫn đến những bài toán mà lời giải hiển nhiên, dễ dàng thực hiện (bài toán cơ sở)
* Mô hình

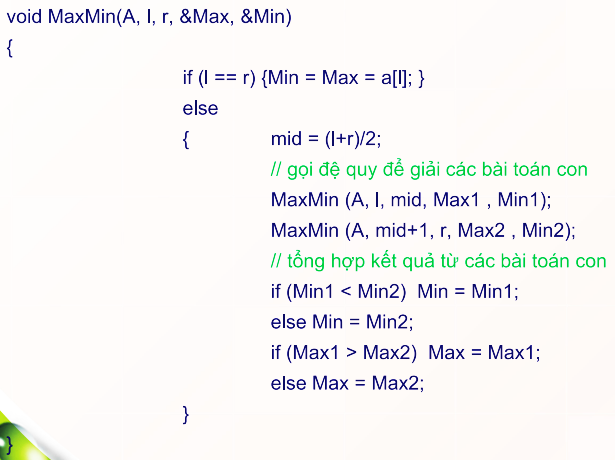


*Ví dụ bài toán tìm Max Min trong đoạn [l,r] của mảng A có n phần tử*

Chia  
 + Nếu l = r 🡪 giải trực tiếp  
 + Ngược lại, chia bài toán thành 2 bài toán con rời nhau

Trị  
 + Tìm kiếm max1, min1 trên bài toán con 1  
 + Tìm kiếm max2, min2 trên bài toán con 2

Tổng hợp kết quả

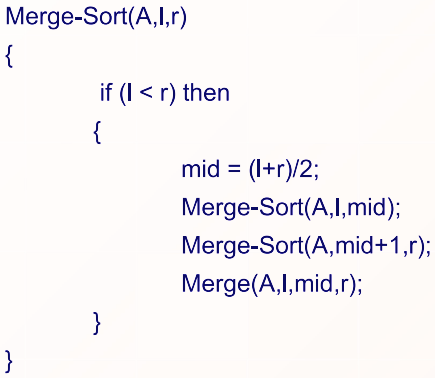


*Ví dụ bài toán thuật toán Merge Sort*

Chia  
 + Nếu mảng A rỗng hoặc chỉ có một phần tử thì trả về chính A ( đã có thứ tự)   
 + Ngược lại, chia A thành 2 mảng con

Đệ quy  
 + Sắp xếp 2 mảng con

Tổng hợp  
 + Xếp xen kẽ hai mảng con đã có thứ tự



*Ví dụ bài toán thuật toán Quick Sort*

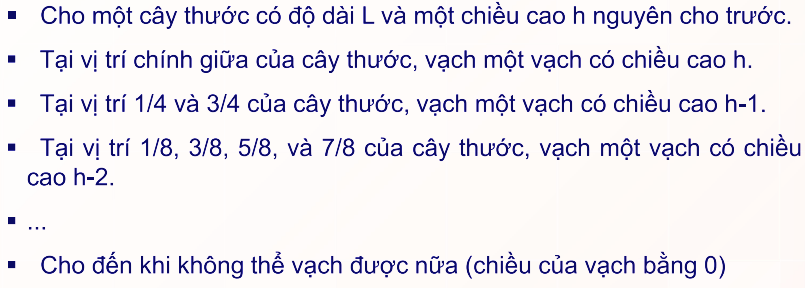
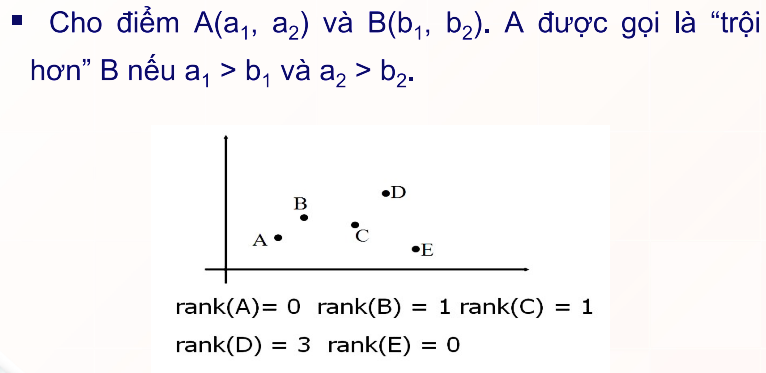
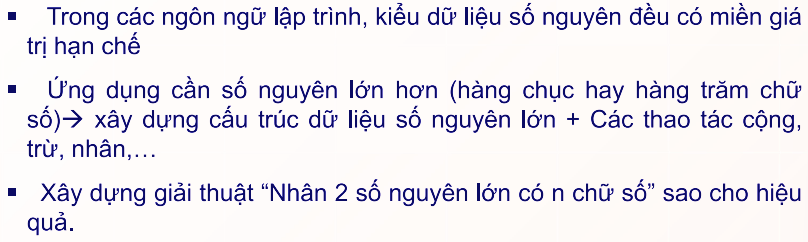
Chia  
 + Chọn 1 phần tử bất kì trong mảng làm nút trục, xác định vị trí hợp hệ cùa nút này trong mảng (vị trí pivot)  
 + Phân hoạch các phần tử còn lại sao cho từ vị trí 0 đến pivot – 1 đều có giá trị nhỏ hơn hoặc bằng nút trục, từ vị trí pivot + 1 đến n -1 lớn hơn nút trục

Đệ quy  
 + Sắp xếp 2 mảng con

Tổng hợp  
 + Không tổng hợp kết quả một cách tường minh (vì đã thực hiện trong quá trình phân hoạch)



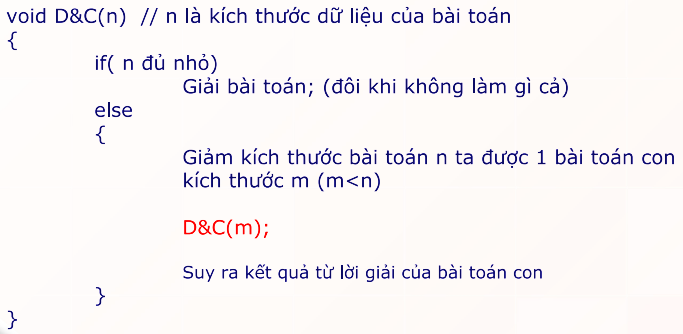
*Một số bài toán có thể giải bằng chia để trị*

1. Bài toán vạch thước  
   
2. Bài toán sắp xếp hạng trong không gian 2D   
   
3. Bài toán nhân 2 số nguyên lớn   
     
   A picture containing text, font, screenshot

   Description automatically generated

**Giảm để trị - Decrease and Conquer**

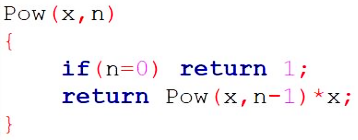
* Ý tưởng  
  + Để giải 1 bài toán P(n) có kích thước n, ta giảm kích thước của nó, giải 1 bài toán con có kích thước nhỏ hơn   
  + Giải bài toán con rồi tìm cách suy ra lời giải của bài toán ban đầu
* Mô hình  
  + Top-down (1 giải thuật đệ quy)  
  + Bottom-up (giải thuật lặp)



* 3 trường hợp giảm kích thước:  
  + Giảm theo hằng số: -1, -2, -3,...  
  + Giảm theo hệ số: ½, 1/3,...  
  + Giảm theo kích thước của biến

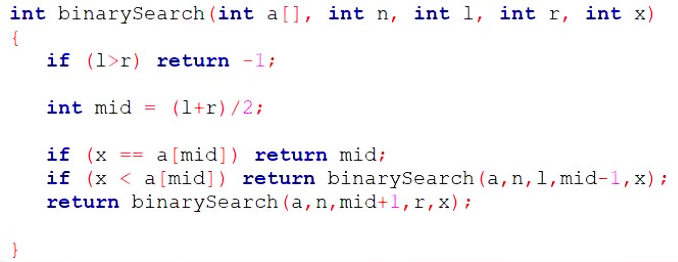
*Ví dụ bài toán tính lũy thừa xn*

Công thức đệ quy: xn = xn-1.x  
Giảm kích thước bởi cùng 1 hằng số (không đổi) là 1

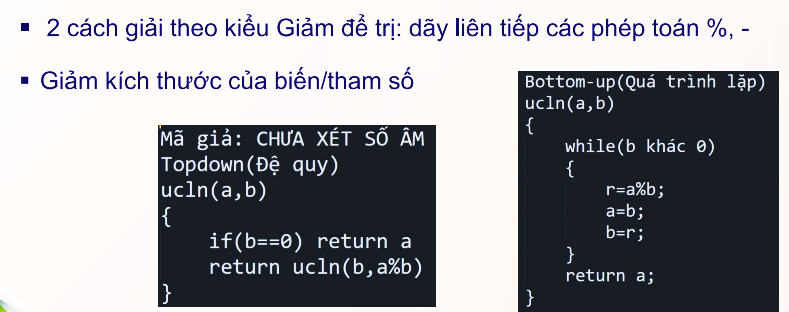


*Ví dụ bài toán Binary Search*

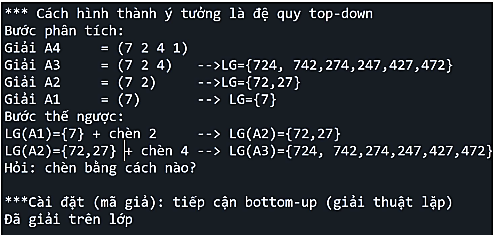
+ Giảm kích thước bởi cùng một hệ số là 1/2  
+ binarySearch(a,n) chuyển thành tìm kiếm trên mảng con có kích thước giảm đi một nửa



*Ví dụ tìm UCLN(a,b)*



*Ví dụ tìm n! hoán vị của 1 tập gồm n phần tử a = (a1, a2, a3,…,an)*



**Phương pháp Quy hoạch động**

\* Chia để trị: một số bài toán con nào đó có thể giải nhiều lần 🡪 chi phí cao, có thể O(cn)  
\* Quy hoạch động: như divide and conquer (kết hợp lời giải của các bài toàn con) 🡪 nâng cấp, khắc phục nhược điểm của D&C  
\* Thường được vận dụng để giải bài toán tối ưu, bài toán có công thức truy hồi

* Dấu hiệu bài toán có thể áp dụng QHĐ  
  + Optimal substructure: một lời giải tối ưu cho một bài toán bao gồm nhiều lời giải tối ưu cho bài toán con   
  + Các bài toán con trùng lặp: một lời giải đệ quy chứa 1 số các bài toán con khác nhau và chúng được lặp lại nhiều lần

\* Khắc phục việc giải dư thừa 1 số bài toán con: lưu trữ kết quả + tra cứu  
 + Tạo ra 1 bảng (vector) để lưu trữ kết quả của các bài toán con   
 + Sử dụng kết quả đã lưu trong bảng mà không cần giải lại bài toán con   
 + Giải các bài toán con theo hướng bottom-up (từ nhỏ đến lớn)

\* Các bước thực hiện  
 + Bước 1: phân tích đặc trưng của cấu trúc lời giải tối ưu: xác định hàm quy hoạch động  
 + Bước 2: tạo bảng: lấp đầy bảng theo một quy luật nào đó  
 ~ Gán giá trị cho 1 số ô nào đó  
 ~ Xác định giá trị của 1 số ô khác nhờ vào giá trị các ô trước đó  
 + Bước 3: tra bảng và truy xuất/xây dựng lời giải của bài toán

* Ưu, nhược điểm