Thiết kế và phân tích giải thuật

Nguyễn Đức Hùng, Đỗ Ngọc Dũng, Nguyễn Quang Minh

September 12, 2019

Outline

Thuật toán

- Các bước thiết kế thuật toán
- Hương tiếp cận top – down
- Phương pháp tinh chỉnh từng bước
- Phân loại thuật toán

Phân tích thuật toán

- Phân tích thuật toán về thời gian thực hiện
- Phân tích thuật toán về mặt không gian
- Phân tích với nhiều tham só

Thuật toán

Thuật toán

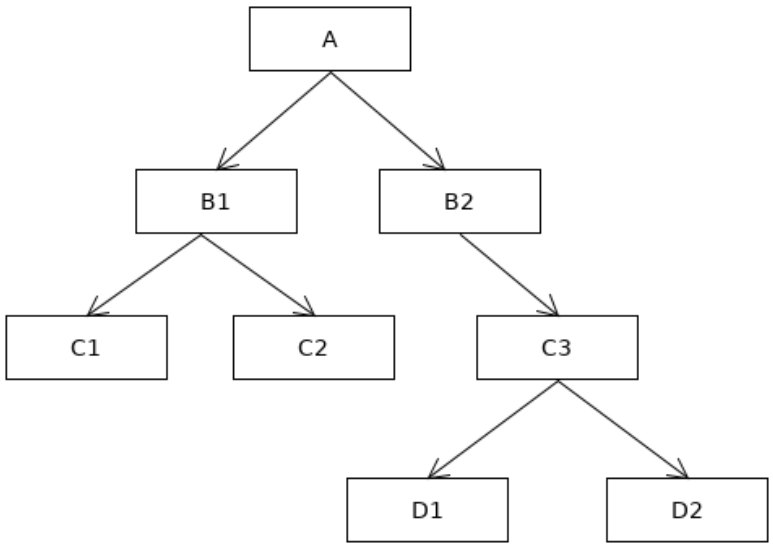
Thuật toán là tập hợp hữu hãn các chỉ dẫn nhằm giải quyết một vấn đề nào đó hoặc thực việc một công việc tính toán nào đó.

Thuật toán đại diện cho cách giải quyết một vấn đề, do đó độc lập với ngôn ngữ lập trình.

Các bước thiết kế thuật toán

1. Problem definition – Định ra vấn đề
2. Development of a model – Mô hình hóa
3. Specification of the algorithm – Xác định đặc trưng
4. Designing an algorithm – Thiết kế thuật toán
5. Checking the correctness of the algorithm – Kiểm tra tính đúng đắn thuật toán
6. Analysis of algorithm – Phân tích thuật toán
7. Implementation of algorithm – Triển khai cài đặt thuật toán
8. Program testing – Kiểm thử
9. Documentation preparation – Tài liệu về thuật toán

Hương tiếp cận top – down



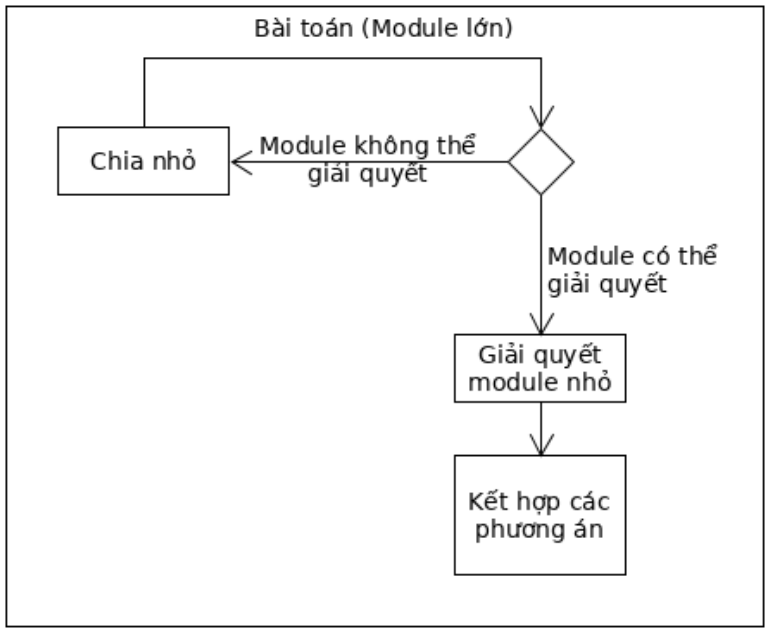


Figure: Chia nhỏ bài toán thành các module cơ bản

Phương pháp tinh chỉnh từng bước

Tinh chỉnh từng bước

Là phương pháp thiết kế giải thuật gắn liền với quá trình lập trình, phản ánh tinh thần của quá trình module hóa và thiết kế của top-down

Phương pháp tinh chỉnh từng bước

1. Thuật toán được thể hiện bằng lời lẽ tự nhiên của người thiết kế
2. Tinh chỉnh lời ý chi tiết và gần với ngôn ngữ lập trình hơn (mã giả)
3. Ngôn ngữ lập trình

Thuật toán

Một số tính chất:

- Chính xác: đảm bảo kết quả trả về là đúng
- Rõ ràng: các bước thực hiện minh bạch
- Tính dừng: các bước thực hiện thuật toán là hữu hạn và thuật toán có điểm dừng
- Phổ dụng: có thể dễ dàng sửa đổi để thích ứng với các bài toán trong cùng một lớp và làm việc trên các dữ liệu đầu vào khác nhau
- Tính khả thi: bộ nhớ yêu cầu đủ nhỏ, thuật toán có thể cài đặt được, thực hiện trong thời gian cho phép...

Cách biểu diễn thuật toán

- Lời nói
- Flowchart
- Mã giả (pseudo code)

Phân loại thuật toán

- Brute-force
- Divide and conquer
- Search and enumeration
- Randomized algorithm
- Reduction of complexity
- Back tracking
- Heuristic

Ví dụ

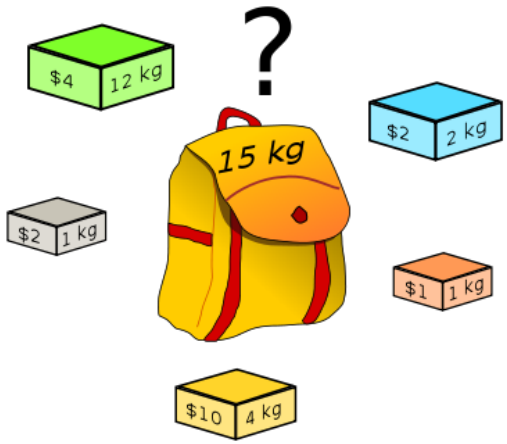


Figure: Bài toán cái túi

Algorithm 1 Giải thuật tham lam

1: **procedure** GREEDY(*value*, *weight*, W)
2: Tính $cost = \frac{value}{weight}$ của mỗi vật
3: **while** $TotalWeight < W$ **do**
4: Lấy vật có *cost* cao nhất
5: **end while**
6: **end procedure**

Phân tích thuật toán

- Phân tích tính đúng đắn
- Phân tích thời gian thực hiện
- Phân tích không gian (bộ nhớ cần dùng)

Phân tích thuật toán về thời gian thực hiện

Thời gian thực hiện thuật toán

Thời gian thực hiện được tính bằng số *phép tính cơ bản* được sử dụng để xử lý đầu vào với một *kích thước* nào đó.

Ví dụ:

phép cộng 2 số nguyên là một phép toán cơ bản.

phép cộng 2 vector *không* phải là một phép toán cơ bản.

Kí hiệu:

- Kích thước dữ liệu đầu vào n
- Thời gian thực hiện giải thuật $T(n)$

Phân tích thuật toán về thời gian thực hiện

n	$\log\log n$	$\log n$	n	$n \log n$	n^2	n^3	2^n
16	2	4	2^4	$4 \cdot 2^4 = 2^6$	2^8	2^{12}	2^{16}
256	3	8	2^8	$8 \cdot 2^8 = 2^{11}$	2^{16}	2^{24}	2^{256}
1024	≈ 3.3	10	2^{10}	$10 \cdot 2^{10} \approx 2^{13}$	2^{20}	2^{30}	2^{1024}
64K	4	16	2^{16}	$16 \cdot 2^{16} = 2^{20}$	2^{32}	2^{48}	2^{64K}
1M	≈ 4.3	20	2^{20}	$20 \cdot 2^{20} \approx 2^{24}$	2^{40}	2^{60}	2^{1M}
1G	≈ 4.9	30	2^{30}	$30 \cdot 2^{30} \approx 2^{35}$	2^{60}	2^{90}	2^{1G}

Figure: Số phép tính cơ bản với độ lớn của dữ liệu đầu vào tương ứng

Phân tích thuật toán về thời gian thực hiện

Phân tích tiệm cận:

- $f(n) \in O(g(n)) \iff \exists C_1, n_o, \ \forall n > n_o : |f(n)| \leq C_1 |g(n)|$
- $f(n) \in \Omega(g(n)) \iff \exists C_2, \ n_o, \forall n > n_o : |f(n)| \geq C_2 |g(n)|$
- $f(n) = \Theta(g(n)) \iff f(n) \in O(g(n)) \wedge f(n) \in \Omega(g(n))$

Các quy tắc:

- Quy tắc tổng
- Quy tắc nhân

Phân tích thuật toán về thời gian thực hiện

Thời gian thực hiện giải thuật trung bình

- Kịch bản tốt nhất
- Kịch bản xấu nhất
- Trường hợp trung bình

Phân tích thuật toán về mặt không gian

Space – time tradeoff

Không gian bộ nhớ chiếm dụng cũng rất quan trọng. Khi tối ưu không gian chiếm dụng, ta có thể gặp phải vấn đề space/time tradeoff – đánh đổi giữa không gian chiếm dụng và thời gian tính toán.

Ví dụ: Tính giai thừa của một số.

Algorithm 2 Tính chay

```
1: procedure GIAI THỪA(m)
2:     n = 1
3:     for i = 2 to m do
4:         n = n × i
5:     end for
6:     Return n
7: end procedure
```

Algorithm 3 Lookup

```
1: facts = [1, 2, 6, 24, ...]
2: procedure GIAI THỪA(n)
3:     Return facts[n]
4: end procedure
```

Phân tích với nhiều tham só

Đôi lúc bài giải thuật của chúng ta có độ phức tạp cần được đánh giá bằng nhiều tham số

Bài toán ví dụ

Cho một bức ảnh có n pixel với m màu khác nhau, đếm số lần xuất hiện của mỗi màu trong ảnh.

Algorithm 4 Đếm màu

```
1: for i = 1 : m do
2:     count[i] = 0
3: end for
4: for i = 1 : n do
5:     count[value(i)] ++
6: end for
```