Các câu hỏi về chia để trị tuy các bạn không hỏi nhưng chúng tôi vẫn trả lời

***. Chia để trị là gì?**
=> Chiến lược **chia để trị** là một chiến lược thiết kế thuật toán theo hướng **top down approach** và dựa trên **đệ quy**. Gồm 3 phần cơ bản là:

- **Chia**: Chia bài toán lớn thành nhiều bài toán con dễ giải quyết hơn (các bài toán con phải **đồng dạng với nhau** và với bài toán lớn và các bài toán con **độc lập** với nhau)
- **Trị**: Giải quyết các bài toán con đó
- **Hợp**: Kết hợp lời giải của các bài toán con để có được lời giải cuối cùng cho bài toán lớn.

1. Chương trình nào thích hợp với chia để trị
=> Chương trình có bộ xử lý đa luồng, chia để trị sẽ chia nhỏ bài toán ra cho từng bộ xử lý chạy song song với nhau rồi hợp kết quả của những bài toán con lại để cho ra kết quả cuối cùng

2. Framework của chia để trị
=>

```
DAC(P){
  if(P is small)
    Solve(p)
  else{
    Divide P into smaller subproblem p_n
    Apply DAC(p_n)
    Combine(DAC(p_n))
  }
}
```

3. Xử lý ma trận của numpy có phải chia để trị không?
=> Numpy dùng quy hoạch động nha (chi tiết xem ở link t post trên group - nó chỉ ra sự khác biệt của quy hoạch động và chia để trị)

4. Sự khác biệt của Branch and Bound với Divide and Conquer
=> BnB và DnC cùng chia nhỏ bài toán ra thành những bài toán nhỏ hơn nhưng BnB sau khi chia nhỏ sẽ loại bỏ những bài toán con không cho ra giải pháp tối ưu (DnC dùng tất cả bài toán con để có kết quả cuối cùng)

5. Bài toán nhỏ phải thỏa mãn tính chất gì?
=> Bài toán nhỏ phải đồng dạng với bài toán lớn (ví dụ: bài toán lớn là sắp xếp, bài toán nhỏ cũng phải là sắp xếp. Bài toán lớn là tìm max/min bài toán nhỏ cũng phải là tìm max/min)

6. Ý tưởng chung của chia để trị
=> Là chia nhỏ bài toán lớn thành các bài toán con nhỏ hơn, dễ giải hơn để giải quyết, nếu bài toán con vẫn chưa đủ nhỏ -> tiếp tục chia nhỏ thành bài toán con nhỏ hơn nữa

7. Chiến lược thiết kế thuật toán là gì?
=> Là một mô hình cách tiếp cận chung để giải quyết 1 bài toán ở các lĩnh vực tính toán khác nhau

8. Nhược điểm của chia để trị
=>

- Chậm hơn các giải thuật có cùng độ phức tạp nhưng sử dụng vòng lặp vì DnC sử dụng đệ quy => có thể được khắc phục bằng cách tận dụng khả năng xử lý song song.
  Can recursion be faster than loops under any specific conditions? - Quora
- Độ hiệu quả phụ thuộc vào tư duy thuật toán của người cài đặt.
- Dư thừa thao tác: Các thao tác chia nhỏ và hợp nhất có thể yêu cầu thêm tài nguyên và thời gian xử lý các bài toán con trùng lặp hoặc phụ thuộc vào nhau. Sự dư thừa này có thể gây ra chênh lệch khá đáng kể về hiệu suất đối với những bài toán lớn.
- Độ phức tạp lớn: Việc ta chia các bài toán lớn thành các bài toán con nhỏ hơn có thể làm tăng độ phức tạp của thuật toán. Điều này đặc biệt đúng khi các bài toán con phụ thuộc vào nhau và phải được giải quyết theo một trình tự nhất định.
- Khó thiết kế: Một số bài toán sẽ rất khó để chia thành các bài toán con hoặc là chúng cần một thuật toán cực kỳ phức tạp để có thể làm được việc đó. Trong trường hợp đó, việc cài đặt thuật toán theo hướng chia để trị thực sự là một thách thức (nên hãy chuyển qua một số chiến thuật khác dễ dùng hơn).
- Tốn tài nguyên bộ nhớ: Dùng đệ quy nên nếu đệ quy quá nhiều có thể gây ra tràn stack.
- Cách giải quyết bài toán con không hiệu quả: Dựa trên cách ta định nghĩa, phân chia và hợp nhất các bài toán con, thuật toán chia để trị sẽ không thường đưa ra cách giải quyết tối ưu nhất. Trong một số trường hợp, ta còn phải cài đặt thêm một số thuật toán tối ưu khác vào để cải thiện kết quả đầu ra.
- Khó khăn trong việc vận hành song song: Trong một số trường hợp, chia bài toán lớn ra thành nhiều bài toán con và xử lý chúng một cách độc lập có thể không thể thực hiện song song được, dẫn đến việc lãng phí tài nguyên phần cứng của thiết bị.

  Introduction to Divide and Conquer Algorithm - Data Structure and Algorithm Tutorials - GeeksforGeeks
  What are advantages and disadvantages of divide and conquer approach? - Quora

9. Ưu điểm của chia để trị:
=>

- Bài toán khó, xử lý dễ dàng (ví dụ: bài tháp Hà Nội)
- Xử lý nhanh: nhanh hơn brute-force và đa số các thuật toán khác

- Có thể giải quyết các bài toán con song song (trên các thiết bị có bộ xử lý đa luồng và bộ nhớ sử dụng chung) để tối ưu tài nguyên phần cứng và thời gian xử lý
- Thuật toán hiệu quả: những bài toán tưởng chừng rất tốn thời gian để xử lý như biến đổi Fourier thì đc giải quyết trong trong thời gian rất nhanh (chỉ O(NlogN)). Đồng thời chia để trị cũng là nền tảng để người ta tìm ra và xây dựng lên các thuật toán hiệu quả hơn (như quy hoạch động chẳng hạn)
- Sử dụng bộ nhớ hiệu quả: nếu bài toán đủ nhỏ, DnC sẽ sử dụng bộ nhớ cache thay vì bộ nhớ chính -> nhanh hơn nhiều.
- Cho ra kết quả chính xác hơn (chênh lệch thường nằm ở việc làm tròn) so với các giải pháp dùng vòng lặp thông thường.
  Divide & Conquer vs Dynamic Programming (afteracademy.com)
  Introduction to Divide and Conquer Algorithm - Data Structure and Algorithm Tutorials - GeeksforGeeks

10. Khi nào thì sử dụng chia để trị (DnC tốt cho bài toán nào?)

=> Ta nên dùng chia để trị khi:

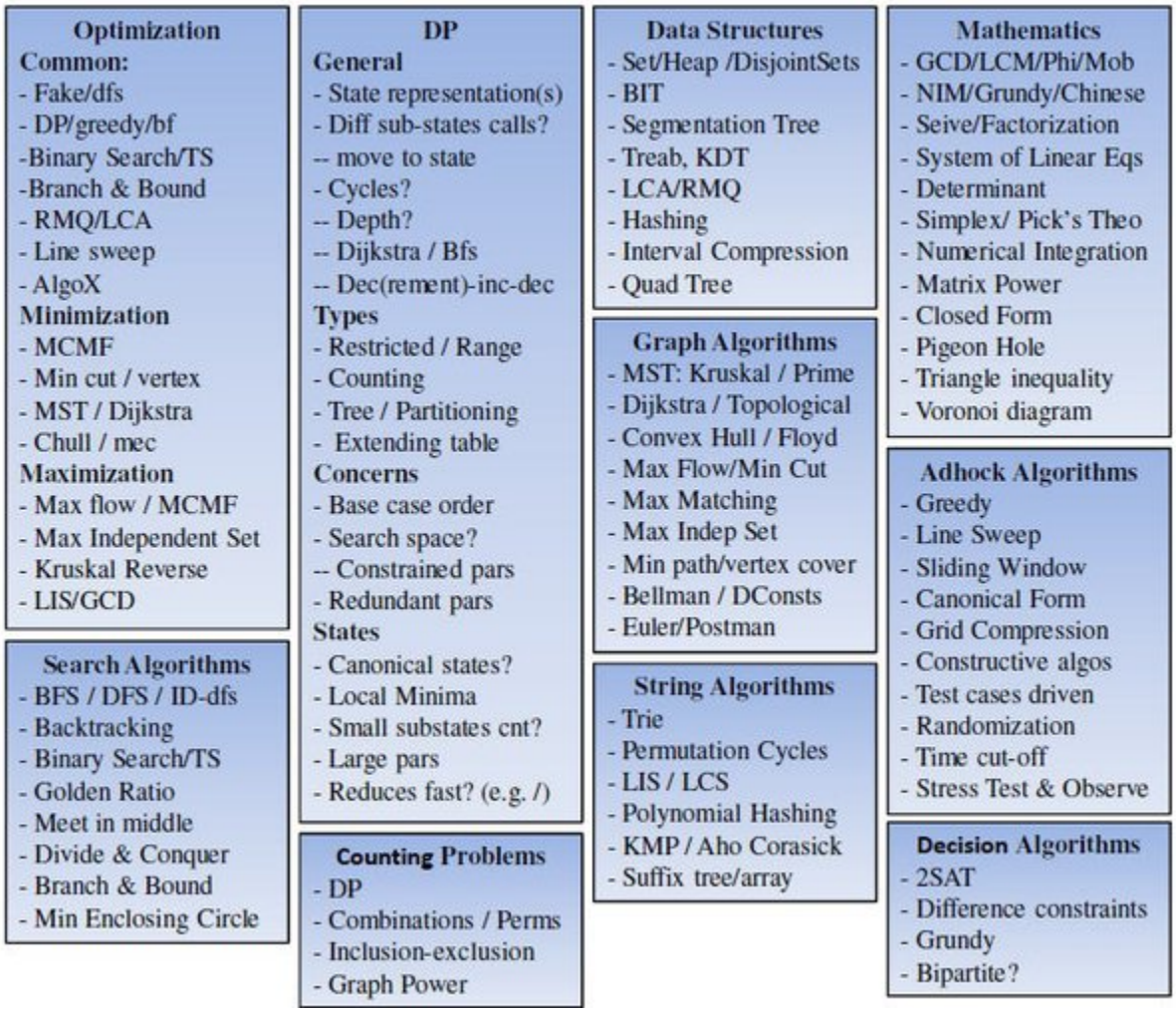
- Bài toán có thể giải quyết được nhờ đệ quy
- Bài toán lớn có thể được chia ra thành các bài toán con đồng dạng
- Các bài toán con độc lập với nhau và có thể được giải quyết đồng thời
- Kết quả của bài toán lớn có thể dễ dàng tìm được bằng cách kết hợp kết quả của cá bài toán con với nhau
- Các bài toán con của chúng ta không cần được tái tính toán (dùng đi dùng lại) nhiều lần -> nếu phải dùng lại nhiều lần dùng quy hoạch động
- Khi áp dụng chia để trị, số lượng phép toán cần sử dụng được giảm đi
  When do we use divide and conquer? - Quora
  How do you know when to use the 'Divide and Conquer' algorithm technique? - Quora

11. Các ứng dụng của DnC

=> Nhiều. Rất nhiều. Sau đâu là sơ sơ vài cái đọc cho vui:

- Các thuật toán phổ biến sử dụng chia để trị
  - Binary Search
  - Quick Sort
  - Merge Sort
  - Tìm cặp điểm gần nhất trong không gian
  - Phép nhân ma trận Strassen
  - Phép nhân số cực lớn Karatsuba
  - Thuật toán Cooley–Tukey cho phép biến đổi nhanh Fourier.
- Là nền tảng của nhiều thuật toán xịn xò hơn như quy hoạch động
- Dùng để tối ưu hóa quy hoạch động
  Divide and Conquer DP - Algorithms for Competitive Programming (cp-algorithms.com)

- Ứng dụng trong lập trình thi đấu: thường thường được dùng trong các thuật toán tìm kiếm.



- Dùng trong bảo mật: 1504679599_CIET_016.pdf (ijarse.com)
- Dùng trong xử lý ảnh: New Divide and Conquer Method on Endmember Extraction Techniques (thesai.org)
- Dùng trong model checking: Divide & Conquer Approach to Leads-to Model Checking | The Computer Journal | Oxford Academic (oup.com)
- Trong ML/DL: Divide and Conquer Networks | OpenReview
  Divide and Conquer: A Flexible Deep Learning Strategy for Exploring Metabolic Heterogeneity from Mass Spectrometry Imaging Data - PubMed (nih.gov)
- Và một mớ thứ khác: What kinds of problem can be solved using divide and conquer method? I mean why we use divide and conquer method? - Quora

12. Thiết thuật toán theo hướng DnC thì làm kiểu gì?

=> Thường là dùng đệ quy. Có thể dùng stack thay cho đệ quy. Có thể dùng vòng lặp nhưng sẽ phải đổi hướng tiếp cần của chia để trị từ top down thành bottom up.

13. Vấn đề nào không nên sử dụng chia để trị?

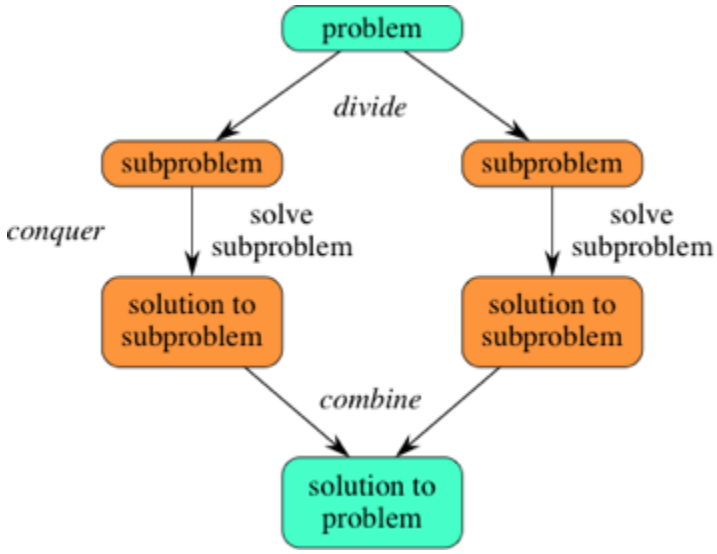
=>

- Chia để trị có thể không sử dụng được khi gặp các bài toán con trùng lặp hoặc phụ thuộc vào nhau -> tính toán lại kết quả của bài toán con nhiều lần -> thay vào đó ta nên dùng dynamic programing (quy hoạch động) để giải quyết thì sẽ có được hiệu quả tốt hơn.
- Hoặc là khi bài toán lớn của chúng ta không thể chia nhỏ ra thành các bài toán con đồng dạng với nhau và với bài toán lớn được.
- Hoặc là khi ta cần rất rất nhiều bài toán con để giải quyết 1 bài toán lớn.
  How do I understand a problem is divide and conquer? - Quora
  Where do divide and conquer algorithms fail? - Quora

15. Các bước của chia để trị:

=> Tùy vào bài toán:

- Trong quá trình sử dụng ta sẽ thấy có những vấn đề chia nhỏ ra vẫn còn lớn -> ta phải tiếp tục chia nhỏ -> cho đến khi vấn đề đủ nhỏ để có thể giải quyết được một cách độc lập thì thôi.
- Đôi khi còn có một số vấn đề không cần phải hợp lời giải của các bài toán con lại mà vẫn cho ra lời giải cuối cùng (ví dụ: thuật toán tìm kiếm nhị phân)
- TL;DR: Các bước của chia để trị phụ thuộc vào bài toán. Đôi khi ta phải chia nhiều lần. Đôi khi ta chả cần phải hợp.
  algorithm - Divide and Conquer approach - Stack Overflow



16. Có thể làm DnC mà không đệ quy không? Ví dụ?

=> Được, ta có thể dùng stack thay cho đệ quy hoặc không dùng stack và đệ quy luôn nhưng để làm được như vậy chúng ta phải thay đổi hoàn toàn hướng tiếp cận của chia để trị, từ top-down approach thành bottom-up approach

Divide and Conquer Method vs Dynamic Programming - javatpoint
Ví dụ: với merge sort, ta cho sẵn kích thước mảng là 4, ta sẽ thiết kế riêng thuật toán này để thao tác với mảng 4 phần tử đó -> chia nó thành 2 mảng 2 phần tử -> rồi sắp xếp 2 mảng -> hợp chúng lại -> lại sắp xếp. Nhưng khi ta đưa thuật toán này vào mảng có số phần tử khác, nó sẽ không hoạt động
Can divide and conquer algorithmic problems only be solved using recursion? - Quora

17. Độ phức tạp thời gian của divide and conquer
=> T(n) = aT(n/b) + f(n),
Với,
n = Kích thước của dữ liệu đầu vào
a = số bài toán con trong vòng lặp đệ quy
n/b = kích thước của các bài toán con.
f(n) = thời gian để làm các thao tác cơ bản của bài toán như chia bài toán ra thành các bài toán con và hợp chúng lại

*18. Thực tế người ta dùng divide and conquer không hay dùng theo dạng lai?
=> Người ta có dùng chia để trị theo dạng thuần cũng có theo dạng lai, có thể đọc thêm về ứng dụng của dạng lai chia để trị ở đây:
Divide-And-Conquer Hybrid Methods for Smaller Quantum Computers (berkeley.edu)
An efficient hybrid tridiagonal divide-and-conquer algorithm on distributed memory architectures - ScienceDirect
Hybrid divide-and-conquer approach for tree search algorithms~:~possibilities and limitations (quantum-journal.org)

19. Độ phức tạp tổng quát của DnC
https://www.csd.uwo.ca/~mmorenom/AM583/Lectures/IntroductionToAXIOM.html/node12.html

20. Các bài toán vận dụng có những bài toán nào?

- Divide and Conquer Optimization
- Matrix Exponentiation
- Quicksort , Merge Sort
- Strassen's Algorithm
- Closest Pair of Points
- Strassen's Algorithm
- Karatsuba algorithm for fast multiplication
- Cooley–Tukey Fast Fourier Transform (FFT) algorithm

21. Cách để làm mất những tính chất bất lợi của DnC để làm nó hiệu quả hơn
=> Không làm mất được. Chỉ tận dụng khả năng xử lý các bài toán con song song với nhau để xử lý nhanh hơn được thôi.

22. Những ứng dụng của chia để trị ngoài thiết kế thuật toán
(https://www.fearlessculture.design/blog-posts/dividing-people-is-the-best-way-to-lead
https://www.languagehumanities.org/what-is-a-divide-and-conquer-strategy.htm)

23. Tính độ phức tạp thuật toán bằng master theorem?
-https://csstudyfun.wordpress.com/2009/01/09/dinh-ly-tong-quat-master-theorem-tinh-do-phuc-tap-cac-thuat-toan-chia-de-tri/

24. So sánh những thuật toán sắp xếp dùng DnC và không DnC để làm nổi bật ưu điểm của chia để trị
(https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms)

*25. Sự giống và khác nhau của chia để trị và quy hoạch động?

- Tất cả đều là các kĩ thuật thuật toán được sử dụng để giải quyết các vấn đề
- Đều chia vấn đề cần giải quyết thành các bài toán con nhỏ hơn
- Chia để trị: giải quyết từng bài toán con một cách đệ quy và sau đó kết hợp các giải pháp của các bài toán con để tạo thành một giải pháp cho vấn đề ban đầu. Các bài toán con trong chia để trị thường độc lập với nhau.
- Quy hoạch động: giải quyết các vấn đề có các bài toán con chồng chéo. Lưu trữ các giải pháp cho các bài toán con này để tránh tính toán lặp lại, có thể được coi là một sự mở rộng của chia để trị, nơi mà các bài toán con không phải là độc lập mà thay vào đó là phụ thuộc vào nhau.

*26. Why balancing is necessary in divide and conquer?
=> Here's the idea:

- Solving a divide and conquer problem will cost you: cost/level * number of levels
- If, each level, you break each sub problem into two equal sized chunks, then it will take log_2(n) levels before the sub problem size is broken into chunks of size 1.
- However, if you break each of the the sub problems into one big chunk and one small chunk, then it can take almost n levels before the sub problems are broken into chunks of size 1.
- So, dividing each of the sub problems evenly will cost you: log_2(n) * cost/level
- And, dividing each of the sub problems into a big chunk, and small chunk will cost you somewhere around: n * cost/level (which is much worse)
- So for something like quick sort (which using the idea of divide and conquer):
- It costs n per level
- When it divides the problem evenly into nearly halves, it runs in: O(n * log_2(n))
- When it divides the problem into a big chunk and small chunk, it runs in: O(n*n) i.e. O(n^2)
  How evenly should one divide to conquer quickly? - ScienceDirect

Divide and conquer algorithms (article) | Khan Academy

27. Các yếu tố cần chú ý khi thiết kế một thuật toán chia để trị:

- Biểu thức quan hệ (relational formula)
- Điều kiện dừng
  Divide and Conquer Algorithm with Applications - TechVidvan
  https://www.javatpoint.com/divide-and-conquer-introduction

28. Chia để trị có thêm chi phí phát sinh trong quá trình thực hiện thuật toán không?
=> Gọi hàm đệ quy, càng nhiều đệ quy thì thêm càng nhiều bộ nhớ trong stack -> ngắn gọn là có

29. Học DnC còn còn có thể có thêm những kiến thức gì?
=> Có thể học thêm về quy hoạch động để sau này có thể dùng chia để trị để tối ưu thuật toán quy hoạch động.
(Đọc thêm:
Divide and Conquer Optimization in Dynamic Programming - GeeksforGeeks
What is divide and conquer optimization in dynamic programming? - Quora
Divide and Conquer Optimization | Jeffrey Xiao)

30. Tại sao chia để trị thường hiệu quả hơn Brute Force?
=> Có thể đọc thêm ở đây: complexity theory - Why do divide and conquer algorithms often run faster than brute force? - Stack Overflow

31. Tại sao thuật toán chia để trị không có 'cấu trúc tối ưu'?
=>

- Để trả lời câu hỏi này thì ta cần phải trả lời câu hỏi cấu trúc tối ưu là gì?
  => Cấu trúc tối ưu là một cấu trúc mà kết quả của nó có thể được giải quyết nhờ các bài toán con một cách tối ưu.
- Điều mà chia để trị không thể làm được là tối ưu kết quả của các bài toán con trùng lặp hoặc phụ thuộc nhau -> giải quyết lại 1 vấn đề nhiều lần -> lãng phí thời gian và tài nguyên
  How do I understand a problem is divide and conquer? - Quora

32. Tham lam hay chia để trị tốt hơn?
=> Tham lam vì tham lam không tái tính toán bài toán con bị trùng, chia để trị thì có; tham lam dùng vòng lặp, chia để trị thì dùng đệ quy (đệ quy chậm hơn vòng lặp)
Greedy vs Divide and Conquer Approach - CodeCrucks

33. Quy hoạch động là một bản mở rộng của chia để trị phải không?
=> Chuẩn!
Divide & Conquer vs Dynamic Programming (afteracademy.com)

34. Top down và bottom up approach là gì?
=>

- Top down: người lập trình bắt đầu với việc phân tích yêu cầu chung của cả bài toán, sau đó chia nó thành nhiều nhiệm vụ nhỏ hơn để giải quyết. Ví dụ: Trong hội họa, người ta vẽ một cái khung chung rồi mới phác thảo đến chi tiết.
    - Điểm mạnh: giúp người dùng tập trung và luôn biết được việc mình cần phải làm là gì và cứ bám theo mà hoàn thành.
- Bottom up: người lập trình xây dựng các thuật toán nhỏ, từ từ lên để giải quyết 1 bài toán lớn. Ví dụ: khi được yêu cầu lắp lego thành 1 tòa nhà thì đầu tiên mình đi tìm những miếng khớp với nhau, lặp đi lặp lại đến khi nó thành hình 1 tòa nhà.
    - Điểm mạnh: Giúp ta trong quá trình làm có thể phát hiện ra nhiều cách thức mới để giải quyết vấn đề đó.

  Tham khảo thêm:
  | Bottom-Up Vs. Top-Down Project Management 101 (monday.com)
  Top-down and bottom-up design - Wikipedia
  Top-Down vs. Bottom-Up Approach | Smartsheet
  dynamic programming - What is the difference between bottom-up and top-down? - Stack Overflow

35. Xử lý đồng bộ với chia để trị:
  Divide-and-conquer algorithms for multiprocessors (core.ac.uk)
  Divide and Conquer (intel.com)

36. Đôi thứ về decrease and conquer (giảm để trị):
- Là một biến thể của chia để trị
- Giống chia để trị ở chỗ cũng chia bài toán lớn thành bài toán con đồng dạng, xử lý chúng và từ đó cho ra kết quả của bài toán lớn.
- Khác ở chỗ là thay vì giải quyết tất cả các bài toán con được chia ra từ bài toán lớn, ta chỉ giải quyết 1 phần và lược bỏ phần còn lại.
- Có 3 dạng giảm để trị:
    - Giảm kích thước bài toán theo một hằng số.
    - Giảm kích thước bài toán theo 1 hệ số.
    - Giảm kích thước bài toán không cố định.
- Ví dụ: trong thuật toán tìm kiếm nhị phân (binary search)
    - Chúng ta thực hiện tìm kiếm trên một mảng đã sắp xếp (ở đây chúng ta đang xem xét mảng tăng dần)
    - B1: Ta lấy phần tử ở giữa mảng so sánh nó với phần tử cần tìm kiếm
    - B2: Nếu phần tử giữa mảng là phần tử cần tìm kiếm thì trả về vị trí
    - B3: Còn không, chúng ta chia mảng ra làm 2 nửa, 1 bên trái và 1 bên phải
    - B4: Nếu phần tử cần tìm lớn hơn phần tử ở giữa mảng thì tìm kiếm mảng con bên phải, còn không thì tìm kiếm ở mảng con bên trái
    - B5: Lặp lại từ bước 1 cho mảng con cho đến khi tìm thấy được phần tử cần tìm

- => Ở bước 3 và 4 chúng ta chia mảng ra làm 2 nửa giống chia để trị nhưng bỏ đi một nửa và chỉ thao tác ở một nửa còn lại => Việc này làm giảm các thao tác tìm kiếm và tính toán đi đáng kể => Hiệu năng xử lý tốt hơn chia để trị
- Ưu điểm:
    - Đơn giản
    - Hiệu quả
    - Chỉ áp dụng cho bài toán cụ thể (do áp dụng chỉ cho bài toán này nên có tối ưu hơn hầu hết các thứ khác)
- Nhược điểm:
    - Khó thiết kế
    - Chỉ áp dụng cho bài toán cụ thể (tính tái sử dụng của thuật toán không cao)
- Các ứng dụng:
    - Giảm kích thước bài toán theo hằng số
        - Insertion Sort
        - Thuật toán graph search: DFS, BFS
        - Topological sorting
        - Thuật toán tìm kiếm hoán vị, tổ hợp và dãy con
    - Giảm kích thước bài toán theo hệ số:
        - Binary Search
        - Russian peasant multiplication
    - Giảm kích thước bài toán không cố định:
        - Interpolation search
        - Euclid's algorithm

  Class_6.pdf (fsu.edu)
  Microsoft Word - enumeration_selection.doc (alaska.edu)
  algorithm - Decrease and Conquer in Real world - Stack Overflow
  Microsoft Word - Decrease & Conquer (nitjsr.ac.in)
  Decrease and Conquer - GeeksforGeeks

**. Đọc thêm để hiểu về chia để trị:

Advanced Divide and Conquer (hkoi.org)
ICS 311 #7: Divide & Conquer and Analysis of Recurrences (hawaii.edu)
dc1.pdf (ucsb.edu)
Chia để trị để giải quyết bài toán người thường nhân đi vòng vòng mà có quy mô lớn:
_pdf (jst.go.jp)

Phần phụ lục truyện vui đọc giải trí:
How is competitive programming different from real-life programming? - Quora

Fun fact: Bạn xem hết 36 câu và không nhận ra thiếu mất câu 14. 🙂