



# Luyện tập thiết kế thuật toán

Nhóm 11 – CS112.N22.KHCL  
GV: Thầy Nguyễn Thanh Sơn



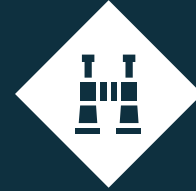
# Thành viên



Bùi Hạ Khánh  
21520282



Huỳnh Võ  
Ngọc Thanh  
21520449



Hồ Yến Nhi  
21520380



# Tổng quan



01

Ôn tập

02

Bài toán 1:  
Cắt giấy

03

Bài toán 2:  
Tìm số nhỏ nhất

04

Bài toán 3:  
Karatsuba





Ôn tập





Các phương pháp thiết kế thuật toán mà chúng ta đã thảo luận ở những buổi học trước và các thuật toán ấy được áp dụng cho những bài toán nào?





- **Vét cạn (Bruteforce)**: dùng khi các phương pháp khác không cho ra lời giải tối ưu về mặt thời gian chạy hay không gian bộ nhớ.
- **Quay lui (Backtracking)**: dùng cho các bài toán yêu cầu lời giải thỏa 1 số điều kiện nào đó, thay vì yêu cầu toàn bộ giải pháp. Thường điều kiện để quay lui là các điều kiện bài toán đưa ra.
- **Nhánh và cận (Branch and bound)**: tương tự với quay lui nhưng điều kiện để quay lui sử dụng 1 hàm tính cận.





- **Chia để trị (Divide and Conquer)**: dùng cho các bài toán có thể được giải bằng cách giải các bài toán con với giá trị đầu vào nhỏ hơn và tổng hợp lại lời giải.
- **Tham lam (Greedy)**: dùng cho các bài toán tương tự như chia để trị, tuy nhiên bài toán phải thỏa điều kiện là lời giải tối ưu nhất được đưa ra từ lời giải tối ưu nhất của bài toán con (cấu trúc con tối ưu - Optimal substructure).
- **Quy hoạch động (Dynamic programming)**: tương tự với giải thuật tham lam, bên cạnh đó bài toán cũng phải có các bài toán con tương tự nhau trong khi chia (Overlapping subproblem).





02

# Bài toán 1: Cắt giấy







# Bài toán

Cho một tờ giấy có kích thước  $A \times B$  ( $1 \leq A, B \leq 140$ ).  
Nhiệm vụ là cắt tờ giấy thành những hình vuông có kích thước bất kỳ. Tìm số ô vuông ít nhất có thể cắt được từ tờ giấy đó.

INPUT	OUTPUT
13 29	9
36 30	5





# Bài toán

Giải thích:

Với  $A = 13$  và  $B = 29$ , có thể cắt được 2 (hình vuông có kích thước  $13 \times 13$ ) + 4 (hình vuông có kích thước  $3 \times 3$ ) + 3 (hình vuông có kích thước  $1 \times 1$ ) = 9

Với  $A = 30$  và  $B = 36$ , có thể cắt được 3 (hình vuông có kích thước  $12 \times 12$ ) + 2 (hình vuông có kích thước  $18 \times 18$ ) = 5





Bài toán có thể được giải bằng phương pháp nào và tại sao có thể dùng phương pháp ấy?  
(Ngoại trừ vét cạn)





Có thể dùng **tham lam** hoặc **quy hoạch động** do việc tìm lời giải tối ưu từ các mảnh giấy nhỏ hơn để tìm ra lời giải cho bài toán lớn.



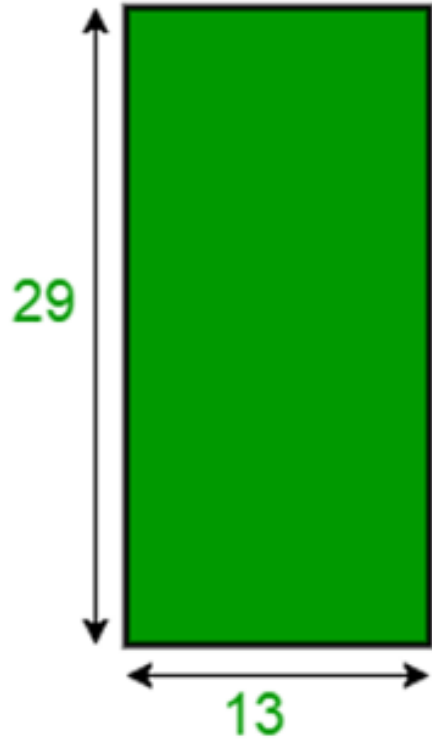


Để thực hiện giải thuật có thể chia mảnh giấy như thế nào ?





Chia thành  $n$  ô vuông có cạnh là chiều rộng của mảnh giấy và 1 hình chữ nhật (hình chữ nhật sử dụng lại cho bài toán con).



Ví dụ:

Nếu tờ giấy có kích thước  $13 \times 29$ , thì hình vuông có cạnh 13  $\Rightarrow$  có thể cắt 2 hình vuông có kích thước  $13 \times 13$  ( $29/13 = 2$  dư 3).

Bây giờ tờ giấy còn lại sẽ có kích thước  $3 \times 13$ . Tương tự, chúng ta có thể cắt tờ giấy còn lại bằng cách sử dụng 4 hình vuông có kích thước  $3 \times 3$  và 3 hình vuông có kích thước  $1 \times 1$ .

$\Rightarrow$  Tối thiểu 9 hình vuông có thể được cắt từ tờ giấy có kích thước  $13 \times 29$ .





Liệu việc chia mảnh giấy thành  $n$  ô vuông (với  $n$  là số các ô vuông tối đa mà mảnh giấy có thể chia) và 1 hình chữ nhật luôn hợp lý ?











Không, vì có trường hợp không thỏa điều đó.





Với trường hợp tờ giấy có kích thước  $36 \times 30$ , phương pháp trên sẽ cho ra kết quả là 6, nhưng có thể cắt tờ giấy thành 5 ô vuông bằng cách cắt 3 hình vuông (kích thước  $12 \times 12$ ) và 2 hình vuông (kích thước  $18 \times 18$ ).



- 
- 
- Dùng **quy hoạch động** xây dựng mảng 2 chiều ( $dp[][]$ ) với kích thước dài x rộng.
  - Thực hiện vòng lặp để tính số lần chia tối ưu cho từng mảnh giấy có kích thước  $1 \times 1$  đến chiều dài x chiều rộng:
    - + Nếu mảnh giấy là một hình vuông ( $i = j$ ), thì gán  $dp[i][j]$  bằng 1 vì không cần phải chia mảnh giấy.
    - + Ngược lại, thử tất cả các đường cắt dọc và ngang để tìm số lần chia tối ưu.
- 
- 



# Độ phức tạp

- Vòng lặp chạy theo chiều dài:  $m$
  - Vòng lặp chạy theo chiều rộng:  $n$
  - So sánh với từng phần tử trước theo chiều dài và chiều rộng:  $(m + n)$
- $\Rightarrow O(m * n * (m + n))$
- 
- 



03



## Bài toán 2: Tìm số nhỏ nhất





# Bài toán

Cho một chuỗi “str” gồm các chữ số và một số nguyên  $n$  ( $0 \leq n \leq \text{str.length} \leq 10^5$ ), hãy tìm số nhỏ nhất có thể bằng cách loại bỏ  $n$  chữ số từ chuỗi mà không thay đổi thứ tự các chữ số ban đầu.

Input: + Dòng đầu là chuỗi “str”

+ Dòng thứ hai là giá trị  $n$

Output: Số nhỏ nhất sau khi đã xóa  $n$  phần tử trong str

INPUT	OUTPUT
121198 2	1118
765028321 5	221





Bài toán có thể được giải bằng phương pháp nào và tại sao có thể dùng phương pháp ấy?  
(Ngoại trừ vét cạn)





Có thể sử dụng **Tham lam**. Cách tiếp cận tham lam hoạt động bằng cách chọn lựa các giá trị tối ưu tại mỗi bước để xây dựng kết quả cuối cùng. Trong bài toán này, để tạo ra một số mới ít hơn  $n$  chữ số, ta cần tìm cách chọn lựa tối ưu từng chữ số trong dãy và xóa các chữ số còn lại.





Làm sao có thể biết được cần phải xóa những chữ số nào?







Để tạo ra số nhỏ nhất, các  $n$  chữ số lớn hơn các số sau nó ở đầu dãy phải bị xóa.

Ví dụ, với số 32879 và  $n = 2$

=> 279 (vì  $3 > 2$  và  $8 > 7$  nên xóa 3 và 8)



Tuy nhiên, nếu không có đầy đủ  $n$  chữ số lớn hơn, thì buộc phải xóa các chữ số lớn ở phía sau.

Ví dụ, với số 132479 và  $n = 2$

=> 12479 và  $n = 1$  (nếu áp dụng cách trên).

Sau đó, tiến hành xóa chữ số 9 ở cuối

=> 1247





Để xác định số gần nhất không thỏa điều kiện  
thì dùng cấu trúc nào?





Có thể thấy đây là LIFO (Last in, First out)  
=> Có thể dùng Stack.

Ví dụ, với  $str = "765028321"$ ,  $n = 5$ :

+ Đầu tiên, thêm phần tử đầu của  $str$  vào  $stack = [7]$

+ Sau đó duyệt từng phần tử tiếp theo trong  $str$  (giả sử  $c$  là giá trị tại vị trí đang xét):

- Với  $c = 6$  ( $str[1]$ ), có  $stack[-1] = 7 > 6 \Rightarrow$  pop 7 ra khỏi  $stack$  và thêm 6 vào đồng thời giảm  $n$  đi 1  $\Rightarrow stack = [6]$ ,  $n = 4$

- Tương tự với 6 và 5 sau đó ta được: Với  $c = 0 \Rightarrow$  pop 5 ra khỏi  $stack$  và thêm 0 vào đồng thời giảm  $n$  đi 2  $\Rightarrow stack = [0]$ ,  $n = 2$

- Với  $c = 2$ , có  $stack[-1] = 0 < 2 \Rightarrow$  thêm 2 vào  $\Rightarrow stack = [2, 0]$

- Với  $c = 8$ , có  $stack[-1] = 2 < 8 \Rightarrow$  thêm 8 vào  $\Rightarrow stack = [8, 2, 0]$


- Với  $c = 3$ , có  $stack[-1] = 8 > 3 \Rightarrow$  pop 8 ra khỏi  $stack$  và thêm 3 vào đồng thời giảm  $n$  đi 1  $\Rightarrow stack = [3, 2, 0]$ ,  $n = 1$

- Với  $c = 2$ , có  $stack[-1] = 3 > 2 \Rightarrow$  pop 3 ra khỏi  $stack$  và thêm 2 vào đồng thời giảm  $n$  đi 1  $\Rightarrow stack = [2, 2, 0]$ ,  $n = 0$





# Độ phức tạp

- Duyệt từng chữ số có trong dãy  $O(\log_{10}(\text{len}(\text{str})))$
  - So sánh với chữ số trước đó  $O(1)$
  - Tiến hành xóa từng chữ số còn lại  $O(n)$
- $\Rightarrow O(\text{len}(\text{str}) + n)$
- 



04



## Bài toán 3: Karatsuba



Phép nhân thông thường trên 2 số (đều có  $n$  chữ số) thì sẽ có độ phức tạp theo thời gian chạy là ?

$\Rightarrow O(n^2)$

Có thể giảm thời gian chạy bằng cách chia nhỏ 2 số ra, tính từng phép nhân và cộng lại hay không ?







Có thể được, hầu hết các thuật toán nhân hiện đại đều sử dụng phương pháp chia để trị (sử dụng tính đồng thời của CPU) cho việc tính toán.

Một số thuật toán:

Karatsuba, Schonhage-Strassen, Furer, Tom-cook





# Bài toán

Thuật toán Karatsuba là một thuật toán nhân nhanh. Nó được Anatoly Karatsuba phát hiện vào năm 1960 và xuất bản năm 1962. Đây là một thuật toán dùng chia để trị để làm giảm phép nhân của hai số có  $n$  chữ số thành ba phép nhân của  $n/2$  chữ số.





# Bài toán

Thuật toán Karatsuba thực hiện như sau:

+ Giả sử, có hai số  $x$  và  $y$  đều có  $n$  chữ số, có thể chia mỗi số thành hai phần bằng nhau, với mỗi phần có  $n/2$  chữ số. Sau đó, biểu diễn  $x$  và  $y$  như sau:

$x = 10^{n/2} * a + b$ ,  $y = 10^{n/2} * c + d$  (Trong đó  $a$  và  $c$  là nửa đầu của  $x$  và  $y$ , còn  $b$  và  $d$  là nửa sau của  $x$  và  $y$ )

+ Sau đó, tính tích của  $x$  và  $y$  bằng cách sử dụng công thức sau:

$$x * y = 10^n * a * c + 10^{n/2} * (a * d + b * c) + b * d$$

+ Đặc biệt là chỉ cần tính ba phép nhân với các số có kích thước  $n/2$ :

$a * c$ ,  $b * d$  và  $(a + b) * (c + d)$  để tìm  $(a * d + b * c)$

Tính  $(a * d + b * c)$  bằng cách lấy  $(a + b) * (c + d) - a * c - b * d$ .





# Bài toán


Input: + Dòng đầu là chuỗi  $x$

+ Dòng thứ hai là chuỗi  $y$

Output: Kết quả phép tính  $x * y$

INPUT	OUTPUT
4154 51454	213739916
6541541541514545454154 15454	4154962260395530977724 3716069997997007620439
6351656156315631654514 5146514654	937711509062916





Ví dụ với  $x = "4154"$  và  $y = "51454"$ :

+ Chia mỗi số thành hai phần bằng nhau:

$$x = 41 * 10^2 + 54$$

$$y = 514 * 10^2 + 54$$

+ Sau đó, chúng ta tính tích của  $x$  và  $y$  bằng công thức:

$$x * y = 10^n * a * c + 10^{n/2} * (a * d + b * c) + b * d$$

$$= 41 * 514 * 10^4 + 10^2 * (41 * 54 + 54 * 514) + 54 * 54$$



+ Chỉ cần tính ba phép nhân với các số có kích thước  $n/2$ :

1.  $a * c = 41 * 514$ . Tiếp tục dùng Karatsuba để tính  $\Rightarrow a * c = 21074$

2.  $b * d = 54 * 54$ . Tương tự với  $a * c \Rightarrow b * d = 2916$

3.  $(a + b) * (c + d) = (41 + 54) * (514 + 54) = 95 * 568$ . Tiếp tục dùng Karatsuba để tính  $\Rightarrow (a + b) * (c + d) = 53960$ . Chúng ta tính  $(a * d + b * c)$  bằng cách lấy  $(a + b) * (c + d) - (a * c) - (b * d) = 53960 - 21074 - 2916 = 29970$

$$\Rightarrow 4154 * 51454 = 10^n * a * c + 10^{n/2} * (a * d + b * c) + b * d$$

$$= 10^4 * 21074 + 10^2 * 29970 + 2916 = 213739916$$



# Độ phức tạp

Vì ta chia số ra làm 2 phần và thực hiện đồng thời 3 lời gọi đệ quy cho 3 phép tính nên độ phức tạp là  $n^{\log_2(3)}$  (theo master theorem)





Buổi thảo luận của nhóm mình đến đây là hết.  
Cảm ơn thầy và các bạn đã lắng nghe !!!

# Thanks!

