



HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.



ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

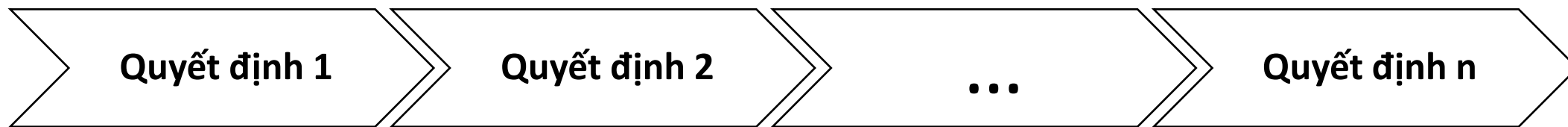
Sơ đồ thuật toán tham lam, chia để trị

ONE LOVE. ONE FUTURE.

A. Thuật toán tham lam

B. Thuật toán chia để trị

- Thuật toán tham lam (Greedy algorithm) là cách tiếp cận đơn giản và có tính trừu tượng cao để giải các bài toán tối ưu, đặc biệt là tối ưu tổ hợp.
- Quá trình tìm lời giải bằng thuật toán tham lam được chia thành nhiều giai đoạn



- Thuật toán tham lam (Greedy algorithm) là cách tiếp cận đơn giản và có tính trừu tượng cao để giải các bài toán tối ưu, đặc biệt là tối ưu tổ hợp.
- Quá trình tìm lời giải bằng thuật toán tham lam được chia thành nhiều giai đoạn

- Ký hiệu:
 - S : Lời giải cần tìm
 - C : Tập các ứng cử viên
 - $select(C)$: Hàm chọn ra ứng cử viên tiềm năng nhất
 - $solution(S)$: Hàm trả về TRUE nếu S là một lời giải hợp lệ, ngược lại hàm trả về FALSE
 - $feasible(S)$: Hàm trả về TRUE nếu S không vi phạm ràng buộc, ngược lại hàm trả về FALSE

```
Greedy() {  
    S =  $\emptyset$ ;  
    while C  $\neq \emptyset$  and not solution(S){  
        x = select(C);  
        C = C \ {x};  
        if feasible(S  $\cup$  {x}) {  
            S = S  $\cup$  {x};  
        }  
    }  
    return S;  
}
```

Sơ đồ chung thuật toán

Khởi tạo, tập lời giải rỗng

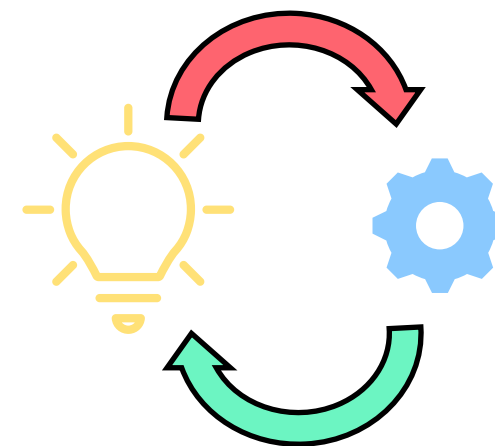
Chừng nào tập ứng cử viên còn khác rỗng
và S chưa phải là lời giải hợp lệ

Lựa chọn x là ứng cử viên tiềm năng nhất
và loại x khỏi tập ứng cử viên C

Nếu bổ sung x vào lời giải đang xây dựng
nếu không vi phạm ràng buộc

```
Greedy() {  
    S =  $\emptyset$ ;  
    while C  $\neq \emptyset$  and not solution(S){  
        x = select(C);  
        C = C \ {x};  
        if feasible(S  $\cup$  {x}) {  
            S = S  $\cup$  {x};  
        }  
    }  
    return S;  
}
```

- Thuật toán đơn giản → Dễ đề xuất và cài đặt
- Trong một số bộ dữ liệu đầu vào và bài toán, thuật toán tham lam cho lời giải tối ưu. Tuy nhiên, đa số trường hợp, đặc biệt là các bài toán thực tế có độ phức tạp cao, thuật toán này không đảm bảo lời giải trả ra là tối ưu
- Thuật toán khá hiệu quả trong những bài toán thực tế có nhiều ràng buộc và độ phức tạp cao.



Ví dụ: Người giao hàng

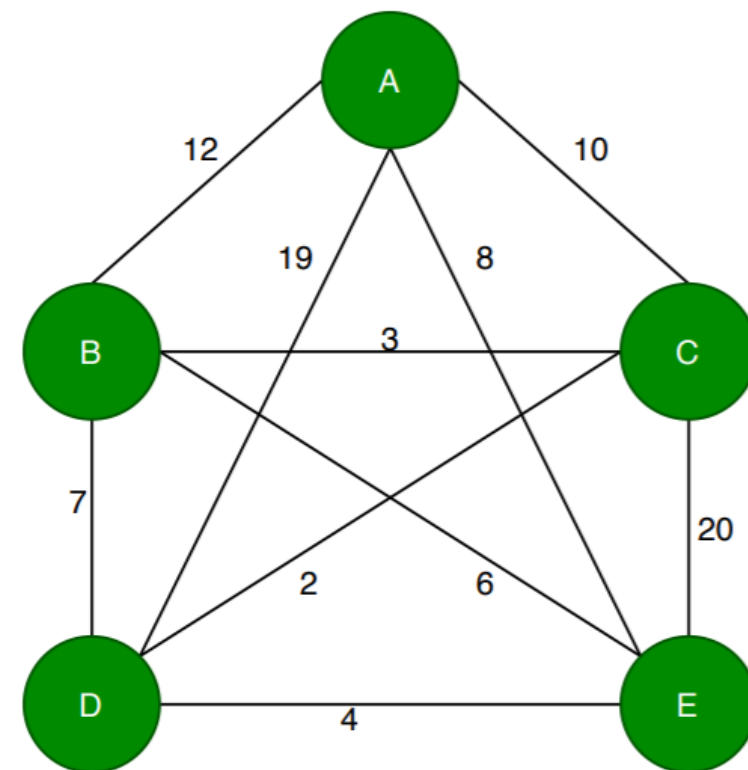
- Bài toán người bán hàng (Traveling Salesman Problem – TSP) là một bài toán tối ưu kinh điển trong lĩnh vực tối ưu tổ hợp
- Bài toán thuộc lớp NP-Hard (Chưa tồn tại thuật toán chạy trong thời gian đa thức giải hiệu quả)
- Phát biểu: Người bán hàng cần tìm một hành trình qua một tập hợp các thành phố sao cho hành trình có tổng chi phí di chuyển là nhỏ nhất và mỗi thành phố chỉ được ghé qua đúng một lần trước khi quay về thành phố xuất phát.



Ví dụ: Người giao hàng

- Thiết kế:

- S : Lời giải cần tìm
- C : Tập các các ứng cử viên
- $select(C)$: Hàm chọn ra ứng cử viên tiềm năng nhất
- $solution(S)$: Hàm trả về TRUE nếu S là một lời giải hợp lệ, ngược lại hàm trả về FALSE
- $feasible(S)$: Hàm trả về TRUE nếu S không vi phạm ràng buộc, ngược lại hàm trả về FALSE



Ví dụ: Người giao hàng

- **Bài toán:** Thiết kế thuật toán tham lam cho bài toán TSP

- **Thiết kế:**

Tập con các cạnh của đồ thị

Tập tất cả các cạnh của đồ thị

Cạnh có trọng số nhỏ nhất

Các cạnh trong S tạo thành một chu trình, mỗi đỉnh xuất hiện đúng 1 lần

Các cạnh trong S có khả năng tạo thành một chu trình khép kín, chứa mỗi đỉnh đúng một lần

- S : Lời giải cần tìm

- C : Tập các ứng cử viên

- $select(C)$: Hàm chọn ra ứng cử viên tiềm năng nhất

- $solution(S)$: Hàm trả về TRUE nếu S là một lời giải hợp lệ, ngược lại hàm trả về FALSE

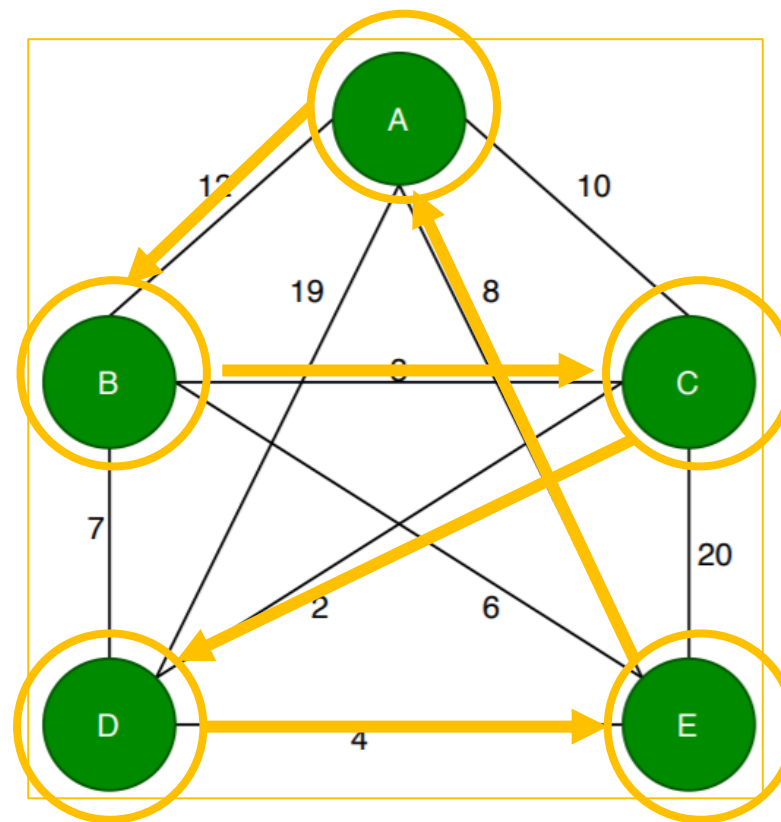
- $feasible(S)$: Hàm trả về TRUE nếu S không vi phạm ràng buộc, ngược lại hàm trả về FALSE

Ví dụ: Người giao hàng

- **Bài toán:** Thiết kế thuật toán tham lam cho bài toán TSP

- **Biểu diễn thuật toán:**

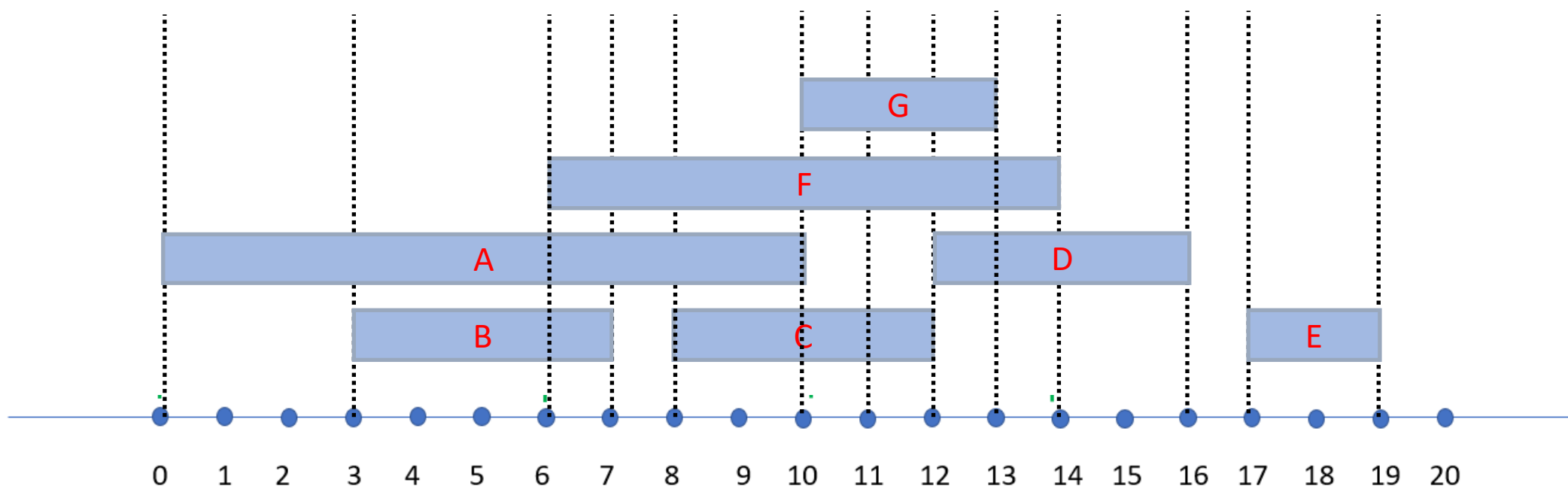
```
Greedy() {  
    S =  $\emptyset$ ;  
    while C  $\neq \emptyset$  and not  
        solution(S){  
        x = select(C);  
        C = C \ {x};  
        if feasible(S  $\cup$  {x}) {  
            S = S  $\cup$  {x};  
        }  
    }  
    return S;  
}
```



Xuất phát từ B: B \rightarrow C \rightarrow D \rightarrow E \rightarrow A \rightarrow B

Ví dụ: Bài toán tập đoạn con không giao nhau cực đại

- Bài toán:** Cho tập các đoạn thẳng $X = \{(a_1, b_1), \dots, (a_n, b_n)\}$ trong đó $a_i < b_i, \forall i \in \{1, \dots, n-1\}$ là tọa độ đầu mút của đoạn thứ i trên đường thẳng, với mọi $i \in \{1, \dots, n\}$. Tìm tập con các đoạn đôi một không giao nhau có số số lượng phần tử lớn nhất.



Lời giải tối ưu: $S = \{B, C, D, E\}$

Ví dụ: Bài toán tập đoạn con không giao nhau cực đại

▪ Thiết kế: Đề xuất thuật toán tham lam

Tập con các đoạn thẳng

Tập tất cả các đoạn thẳng

Đoạn thẳng có ... nhỏ nhất

Các đoạn thẳng không giao nhau

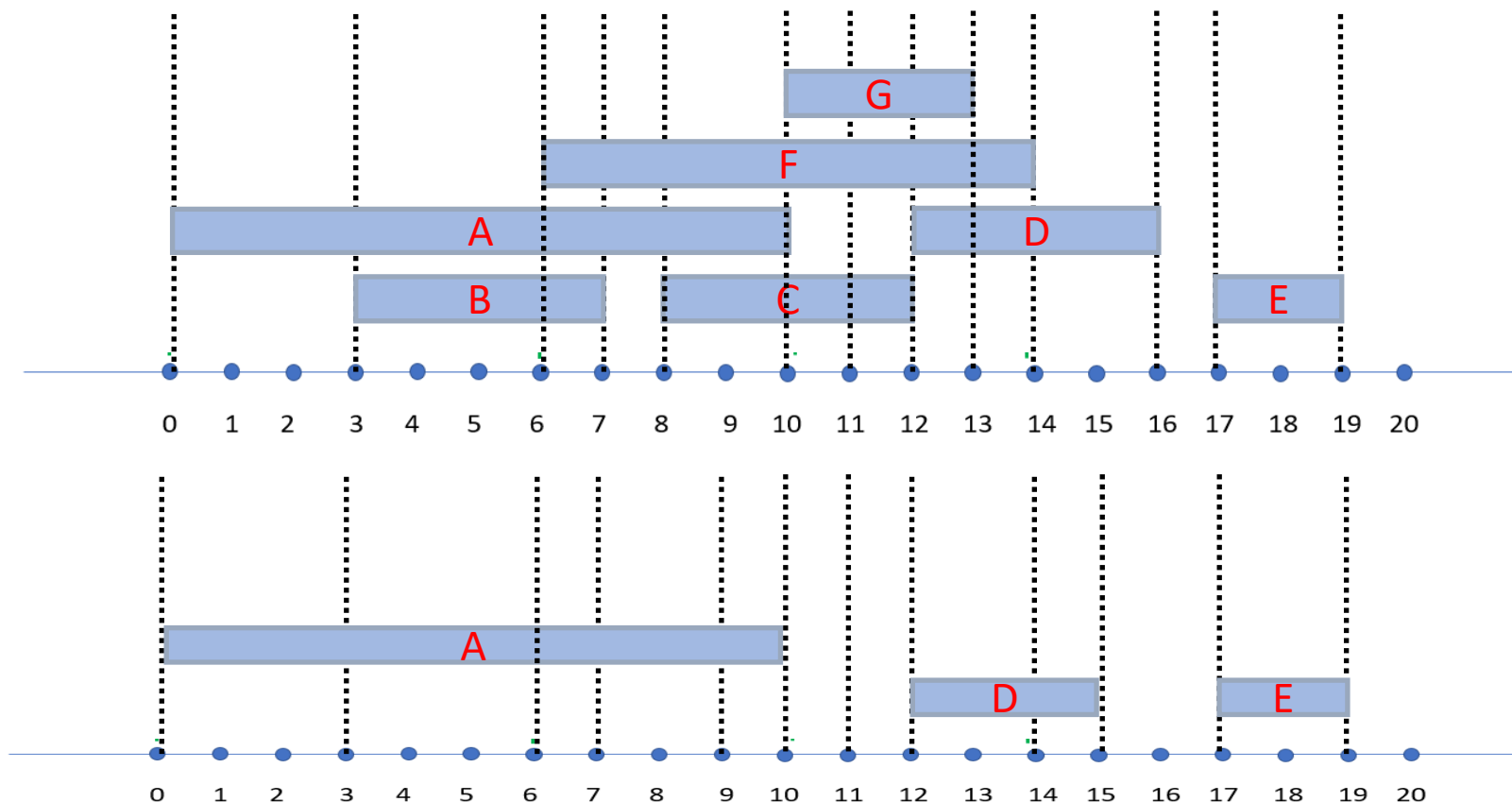
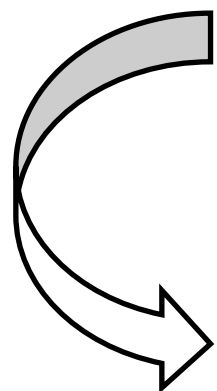
Các cạnh đoạn thẳng trong S không giao nhau

- S : Lời giải cần tìm
- C : Tập các ứng cử viên
- $select(C)$: Hàm chọn ra ứng cử viên tiềm năng nhất
- $solution(S)$: Hàm trả về TRUE nếu S là một lời giải hợp lệ, ngược lại hàm trả về FALSE
- $feasible(S)$: Hàm trả về TRUE nếu S không vi phạm ràng buộc, ngược lại hàm trả về FALSE

Ví dụ: Bài toán tập đoạn con không giao nhau cực đại

Đoạn thẳng có a_i nhỏ nhất

• $select(C)$: Hàm chọn ra ứng cử viên tiềm năng nhất

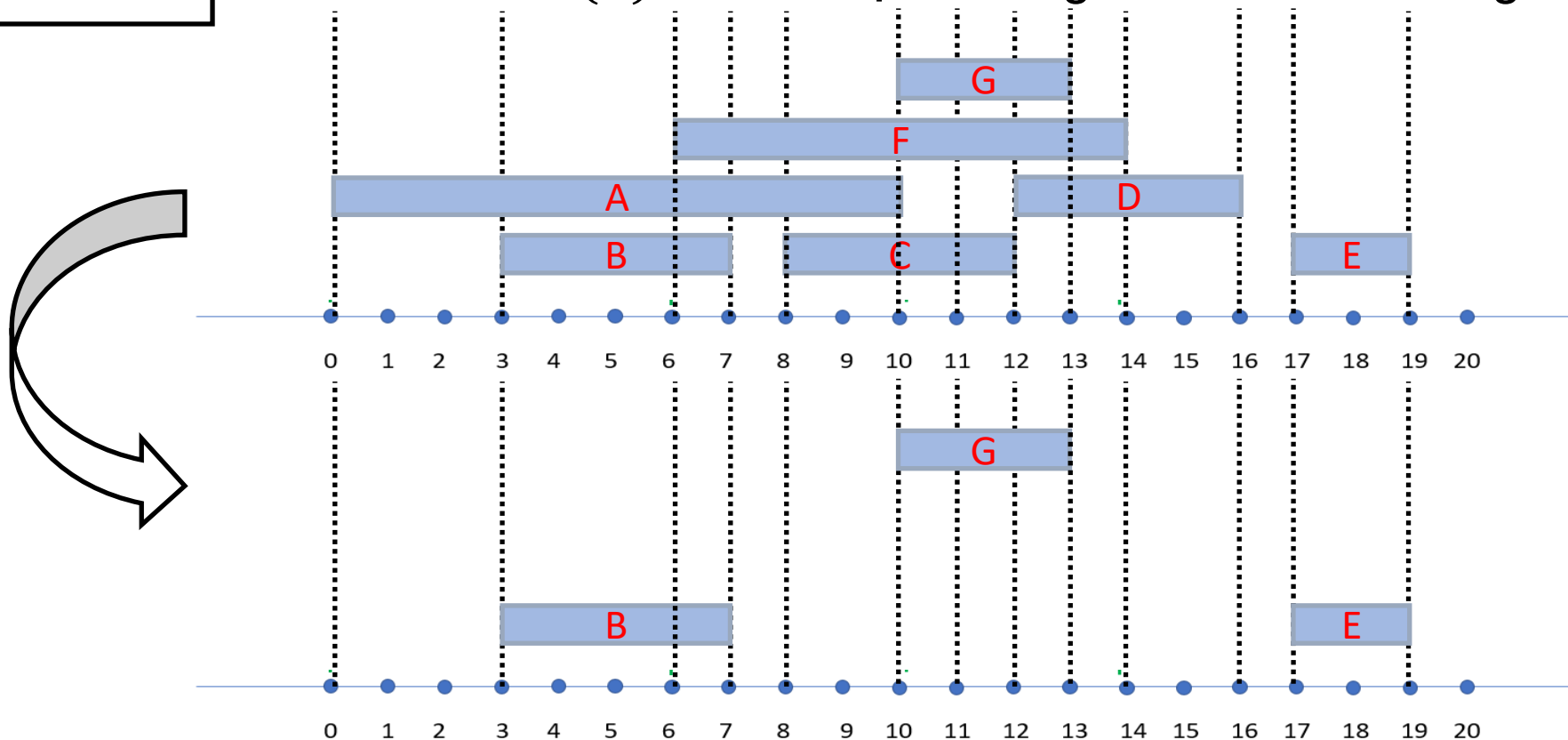


Lời giải này có 3 đoạn → Không tối ưu vì lời giải tối ưu có 4 đoạn ($S = \{B, C, D, E\}$)

Ví dụ: Bài toán tập đoạn con không giao nhau cực đại

Đoạn thẳng có $b_i - a_i$ nhỏ nhất

• *select*(C): Hàm chọn ra ứng cử viên tiềm năng nhất

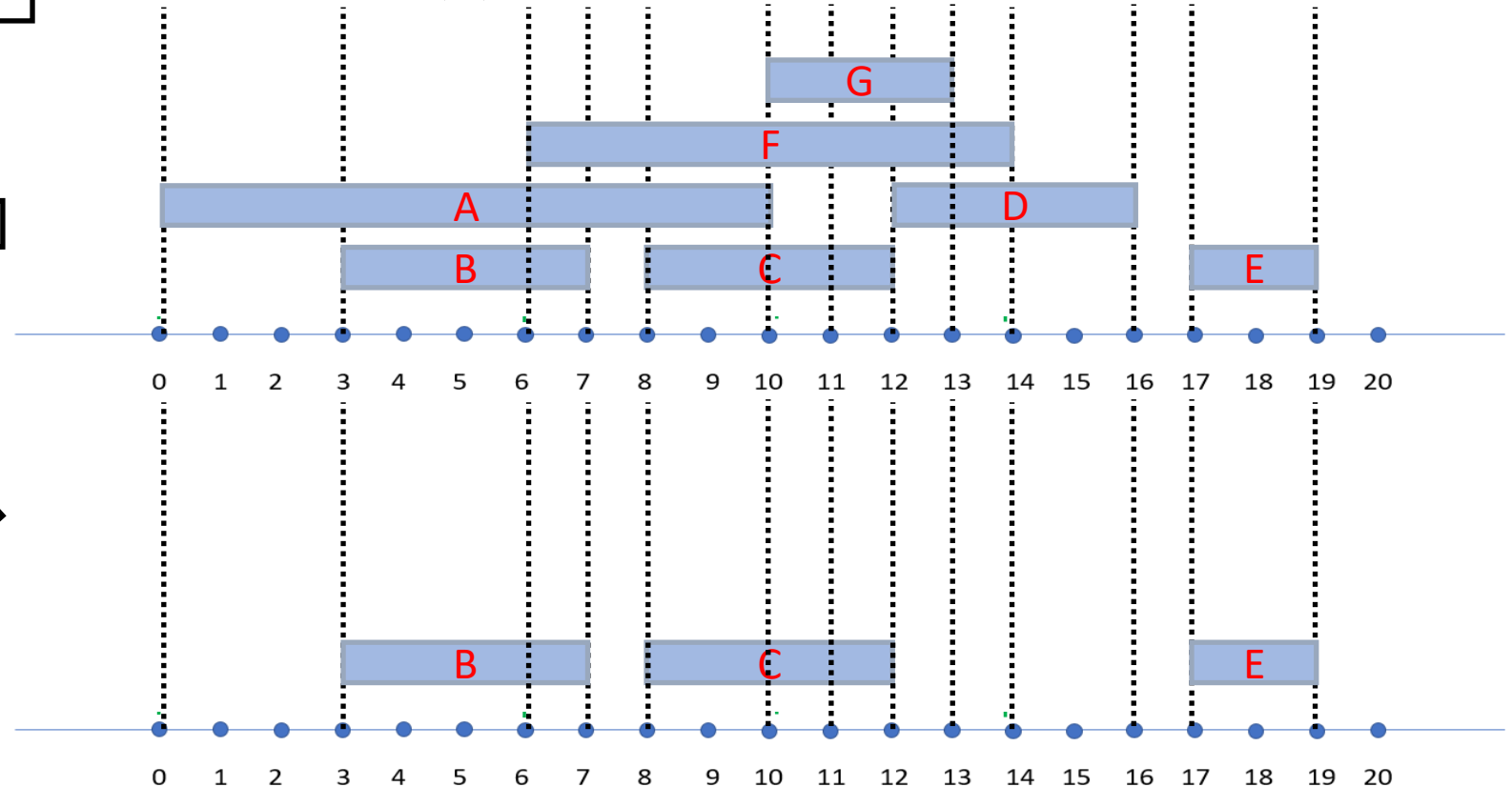
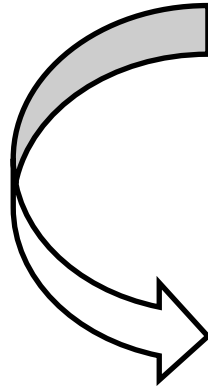


Lời giải này có 3 đoạn → Không tối ưu vì lời giải tối ưu có 4 đoạn ($S = \{B, C, D, E\}$)

Ví dụ: Bài toán tập đoạn con không giao nhau cực đại

Đoạn thẳng có b_i nhỏ nhất

• $select(C)$: Hàm chọn ra ứng cử viên tiềm năng nhất



Thuật toán tìm được lời giải tối ưu có 4 đoạn.

Bài tập về nhà: Chứng minh thuật toán tham lam này luôn trả ra lời giải tối ưu

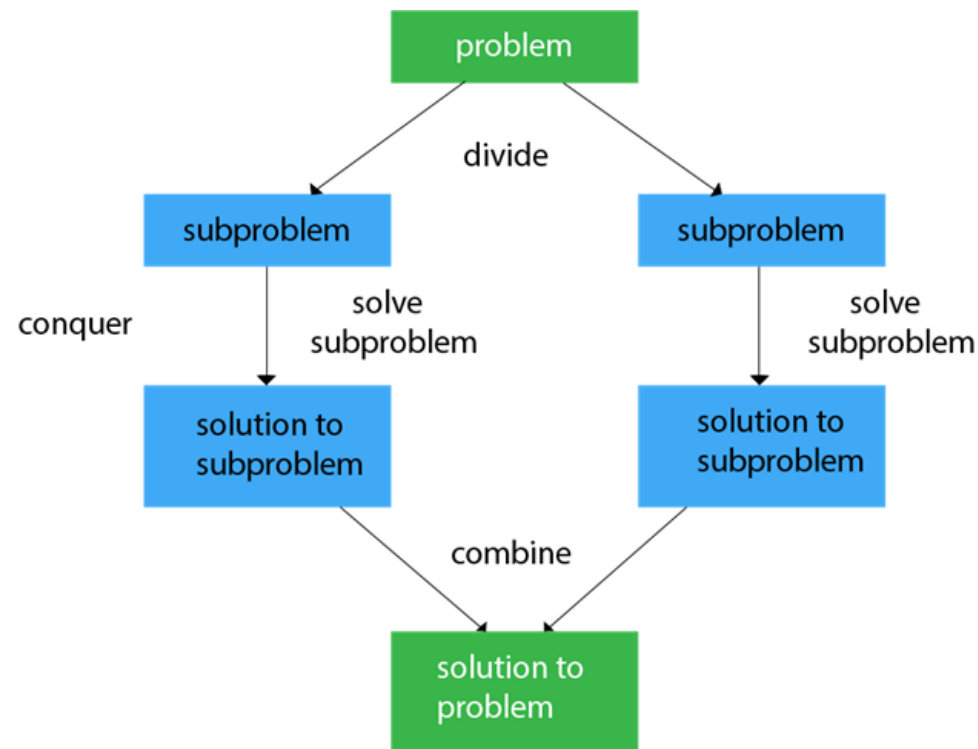
- **Đề bài:** Có n công việc $1, 2, \dots, n$. Công việc i có thời hạn hoàn thành là $d[i]$ và có lợi nhuận khi được đưa vào thực hiện là $p[i]$ ($i = 1, \dots, n$). Biết rằng chỉ được nhiều nhất 1 công việc được thực hiện tại mỗi thời điểm và khi thời gian thực hiện xong mỗi công việc đều là 1 đơn vị. Hãy tìm cách chọn ra các công việc để đưa vào thực hiện sao cho tổng lợi nhuận thu được là nhiều nhất đồng thời mỗi công việc phải hoàn thành trước hoặc đúng thời hạn.
- **Hình thức:** Làm tại nhà và nộp lại trên hệ thống chấm code

A. Thuật toán tham lam

B. Thuật toán chia để trị

Giới thiệu

- Thuật toán chia để trị (Divide and Conquer algorithm) là kỹ thuật mạnh trong Khoa học máy tính và Toán học.
- Ý tưởng: là làm dễ bài toán bằng cách chia nhỏ bài toán phức tạp thành những bài toán con đơn giản hơn (CHIA), giải riêng rẽ các bài toán con (TRỊ) và cuối cùng là tổng hợp các kết quả bài toán con để thu được lời giải bài toán ban đầu.



Nguồn: <https://www.javatpoint.com/divide-and-conquer-introduction>

- Sử dụng thuật toán: Sử dụng **Đệ quy quay lui** để thể hiện sơ đồ chung của thuật toán **chia để trị**
- ➔ Dễ hiểu và thể hiện rõ ý tưởng của thuật toán

```
DC( $P_n$ ) {  
    if  $n \leq n_0$  {  
        solve_problem( $p_n$ );  
    } else {  
         $SP = divide(P_n)$ ;  
        foreach  $p \in SP$  {  
             $s(p) = DC(p)$   
        }  
         $S = aggregate(s(p_1), \dots s(p_k))$   
    }  
    return  $S$ ;  
}
```

Sơ đồ thuật toán

```
DC( $P_n$ ) {  
  if  $n \leq n_0$  {  
    solve_problem( $p_n$ );  
  } else {  
     $SP = divide(P_n)$ ;  
    foreach  $p \in SP$  {  
       $s(p) = \mathbf{DC}(p)$   
    }  
     $S = aggregate(s(p_1), \dots, s(p_k))$   
  }  
  return  $S$ ;  
}
```

n_0 là kích thước nhỏ của bài toán, mà chúng ta có thể chỉ ra lời giải của bài một cách dễ dàng. Đây là bước cơ sở trong kỹ thuật **Đệ quy**.

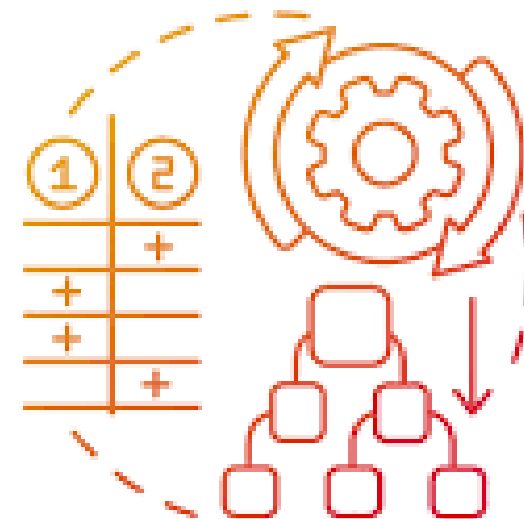
Chia nhỏ bài toán cha thành nhiều bài toán con

Giải từng bài toán con

Tổng hợp kết quả các bài toán con để xây dựng lời giải bài toán cha (bài toán ban đầu)

Đặc điểm

- Ý tưởng thuật toán “Chia nhỏ” rất tự nhiên và giống với cách tiếp cận giải quyết vấn đề của con người
- Các bài toán con độc lập, có nghĩa là việc giải một bài toán con không ảnh hưởng đến giải pháp của các bài toán con đồng mức khác
- Việc chia bài toán đảm bảo không mất nghiệm
- Một số ứng dụng nổi bật
 - Tính toán song song
 - Tìm kiếm



DIVIDE AND CONQUER

Ví dụ: Đoạn con có tổng lớn nhất

- **Phát biểu:** Cho dãy n số nguyên (a_1, a_2, \dots, a_n) , hãy tìm đoạn con bao gồm các phần tử liên tiếp của dãy sao cho tổng các phần tử được chọn là cực đại.

- **Ví dụ:** Cho dãy 7 số nguyên sau

-16	7	-3	0	-1	5	-4
-----	---	----	---	----	---	----

Đoạn con có tổng -12

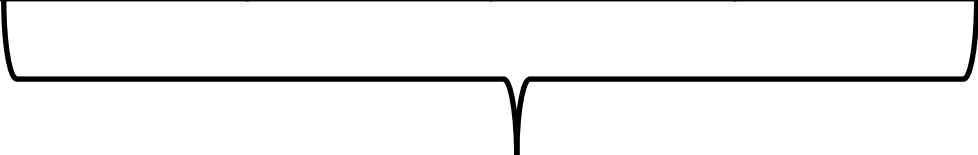
Đoạn con có tổng 0

- Lời giải tối ưu (dãy con có tổng cực đại bằng 8): 7, -3, 0, -1, 5
- Thuật toán vét cạn có độ phức tạp $O(n^2)$, chúng ta có thể làm tốt hơn với thuật toán chia để trị hay không?

Ví dụ: Đoạn con có tổng lớn nhất

- **Xác định bài toán con:** Bài toán con của bài toán tìm đoạn con cực đại của dãy (a_1, \dots, a_n) là bài toán tìm đoạn con cực đại của đoạn con $a_i, a_{i+1}, \dots, a_{i+j}, i \geq 1, i+j \leq n$.

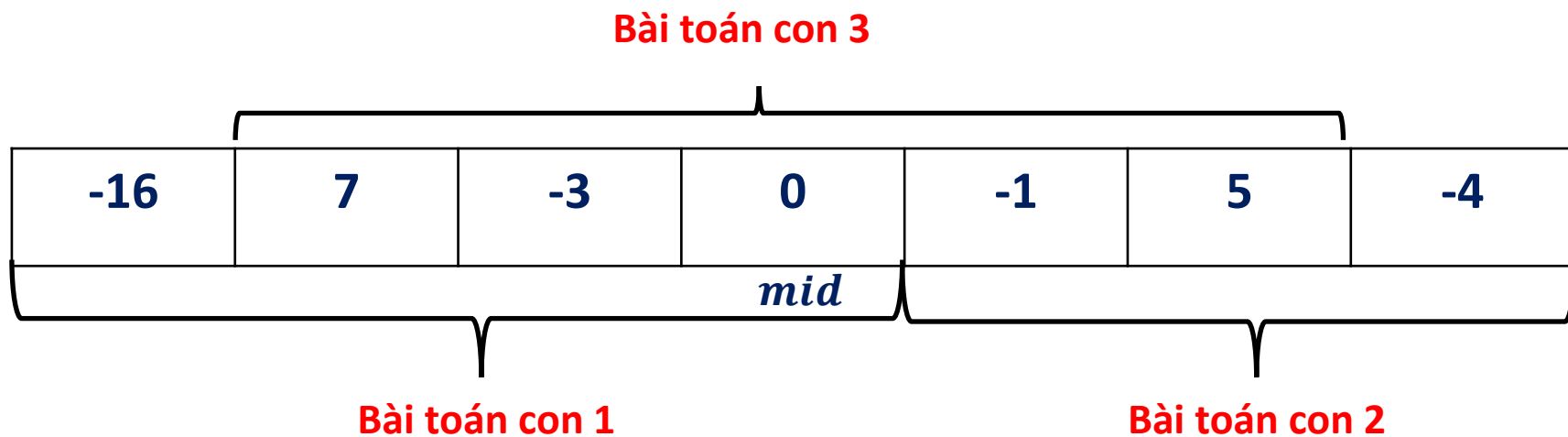
-16	7	-3	0	-1	5	-4
-----	---	----	---	----	---	----



Bài toán con: Tìm đoạn con cực đại của đoạn $(7, -3, 0, -1)$


Ví dụ: Đoạn con có tổng lớn nhất

- **Chia:** Chia đôi dãy n phần tử tại điểm dữa $mid = round(\frac{n+1}{2})$
 - Bài toán con 1: Giống bài toán cha, tìm dãy con cực đại của dãy $1, \dots mid$
 - Bài toán con 2: Giống bài toán cha, tìm dãy con cực đại của dãy $mid + 1, \dots n$
 - Bài toán con 3: Tìm dãy con cực đại có chứa phần tử mid và không biết thành phần đầu tiên và cuối cùng



Ví dụ: Đoạn con có tổng lớn nhất

- **Giải bài toán con:** Bài toán con 1 và 2 có thể giải bằng đệ quy. Bài toán con 3 giải được trực tiếp do có ràng buộc phải chứa phần tử *mid*. Lời giải bài toán 3 là hợp của đoạn con cực đại kết thúc tại *mid* và đoạn con cực đại bắt đầu từ *mid* + 1. Để giải 2 bài toán nhỏ này, chúng ta chỉ cần duyệt từ *mid* lùi về 1 và từ *mid* + 1 tiến về *n*, với độ phức tạp $O(n)$



-16	7	-3	0 <i>mid</i>	-1	5	-4
-----	---	----	-----------------	----	---	----

Ví dụ: Đoạn con có tổng lớn nhất

- **Xác định bài toán con:** Bài toán con của bài toán tìm đoạn con cực đại của dãy (a_1, \dots, a_n) là bài toán tìm đoạn con cực đại của đoạn con $a_i, a_{i+1}, \dots, a_{i+j}, i \geq 1, i + j \leq n$.
- **Chia:** Chia đôi dãy n phần tử tại điểm dữa $mid = \text{round}(\frac{n+1}{2})$
 - Bài toán con 1: Giống bài toán cha, tìm dãy con cực đại của dãy $1, \dots, mid$
 - Bài toán con 2: Giống bài toán cha, tìm dãy con cực đại của dãy $mid + 1, \dots, n$
 - Bài toán con 3: Tìm dãy con cực đại có chứa phần tử mid và không biết thành phần đầu tiên và cuối cùng
- **Giải bài toán con:** Dùng đệ quy và duyệt mảng độ phức tạp $O(n)$
- **Tổng hợp:** Lời giải là lời giải tốt nhất trong 3 lời giải của 3 bài toán con
- **Độ phức tạp thuật toán:** $O(\log(n))$

Ví dụ: Tính nhanh a^n

- **Phát biểu:** Tính a^n với a và n là các số nguyên dương
- **Phân tích:** Phương pháp trực tiếp, thực hiện n phép nhân, có độ phức tạp tính toán $O(n)$. Chúng ta có thể làm tốt hơn với Thuật toán chia để trị?
- **Thiết kế thuật toán chia để trị:**
 - **Bài toán con:** Tính $a^k, k < n$
 - **Chia:** Ý tưởng chia đôi, 2 bài toán con giống hệt nhau, chỉ cần giải một bài toán con
 - Nếu n chẵn, ta có $a^n = \left(a^{\frac{n}{2}}\right) \times \left(a^{\frac{n}{2}}\right)$
 - Nếu n lẻ, ta có $a^n = a \times a^{n-1}$
 - **Xử lý:** Sử dụng đệ quy để giải
 - **Tổng hợp:** Đơn giản, chỉ là một phép tính toán nhân
 - **Độ phức tạp thuật toán:** $O(\log(n))$

Ví dụ: Tính nhanh a^n

- Code minh họa

```
int Pow(int x, int n) {  
    if (n == 0) return 1;  
    if (n % 2 != 0) return x * pow(x, n - 1);  
    int res = Pow(x, n/2);  
    return res * res;  
}
```

- **Ý tưởng:** Cài đặt thuật toán Sắp xếp trộn (Merge sort)
- **Đề bài:** Sắp xếp mảng gồm n số nguyên sử dụng thuật toán chia để trị. Gợi ý, bài toán cha chia thành 2 bài toán con bằng nhau.
- **Hình thức:** Làm tại nhà và nộp lại trên hệ thống chấm code

A large graphic on the left side of the slide. It features a dark blue background with a circular pattern of red dots of varying sizes, creating a sense of depth and movement. The word "HUST" is centered within this pattern in a bold, white, sans-serif font.

HUST

THANK YOU !