CANTHO UNIVERSITY

Thực hành
KỸ THUẬT THAM ĂN

BÀI TOÁN NGƯỜI GIAO HÀNG (TSP)

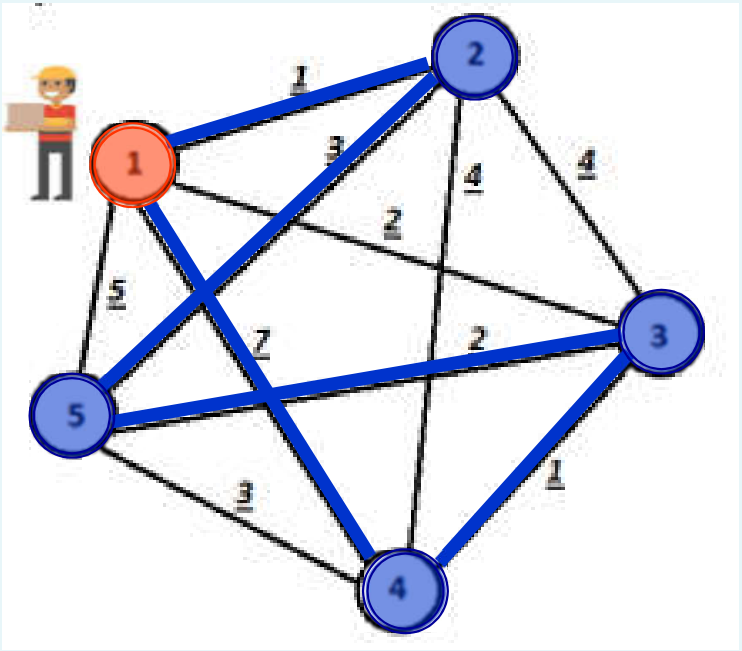
f(X) → MIN

Võ Huỳnh Trâm

CANTHO UNIVERSITY

Bài toán đường đi của người giao hàng (TSP - Travelling Salesman Problem) : Mô tả

- **Bài toán**: Một người giao hàng cần đi giao hàng tại **n** thành phố. Xuất phát từ một thành phố, đi qua các thành phố khác và trở về thành phố ban đầu.
- **Yêu cầu**: Tìm một chu trình sao cho tổng độ dài các cạnh là nhỏ nhất hay một phương án có giá nhỏ nhất ?



TSP: KỸ THUẬT THAM ĂN

2 phương pháp tiếp cận bằng thuật toán Tham ăn:

(1) **Phương pháp 1**: Xây dựng phương án bằng cách chọn những cạnh có độ dài nhỏ nhất phù hợp.

(2) **Phương pháp 2**: Xây dựng phương án bằng cách xuất phát từ một thành phố, đi theo cạnh nhỏ nhất để đến thành phố khác, …

$\Rightarrow$ Phần này cài đặt thuật toán theo **Phương pháp 1**

TSP: KỸ THUẬT THAM ĂN – SẮP XẾP ĐỘ DÀI CẠNH

(1) Sắp xếp các cạnh theo thứ tự tăng của *độ dài*.

(2) Xét các cạnh có *độ dài* từ nhỏ đến lớn để đưa vào ***phương án***

(3) Một cạnh sẽ được đưa vào ***phương án*** nếu:

- **Không tạo thành một đỉnh có cấp ≥ 3**
- **Không tạo thành một chu trình thiếu**

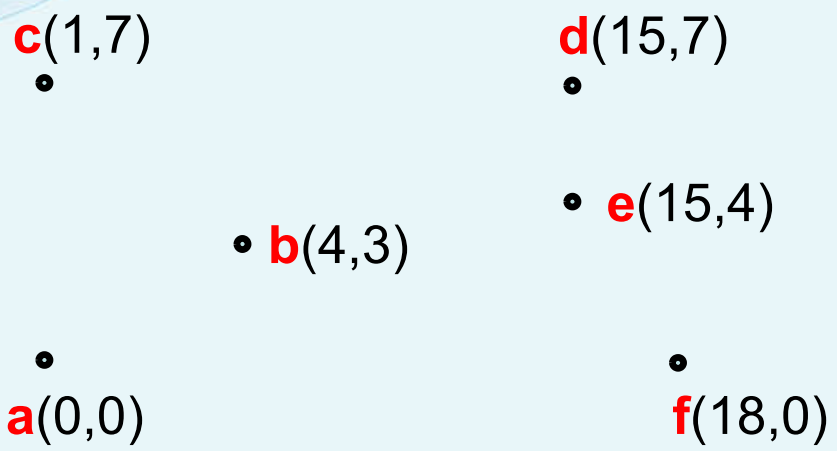
(4) Lặp lại bước (3) cho đến khi chọn được (n-1) cạnh

(5) Tìm cạnh thứ n tạo thành **chu trình** với (n-1) cạnh đã chọn.

* Chỉ chọn trong **n(n-1)/2** cạnh ứng với n đỉnh $\Rightarrow T(n) = \mathbf{O(n^2)}$

CANTHO UNIVERSITY

TSP: Ví dụ



TT	Cạnh	X_1	Y_1	X_2	Y_2	Độ dài
1	de	15	7	15	4	3.00
2	ab	0	0	4	3	5.00
3	bc	4	3	1	7	5.00
4	ef	15	4	18	0	5.00
5	ac	0	0	1	7	7.07
6	df	15	7	18	0	7.62
7	be	4	3	15	4	11.05
8	bd	4	3	15	7	11.70
9	cd	1	7	15	7	14.00
10	bf	4	3	18	0	14.32
11	ce	1	7	15	4	14.32
12	ae	0	0	15	4	15.52
13	ad	0	0	15	7	16.55
14	af	0	0	18	0	18.00
15	cf	1	7	18	0	18.38

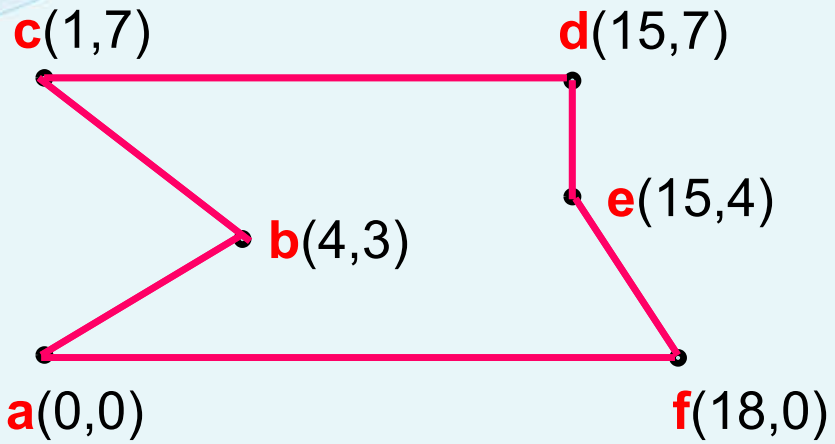
TSP: n = 6 ; Nếu $A(X_1, Y_1)$ và $B(X_2, Y_2)$ thì độ dài cạnh = $\sqrt{(X_1 - X_2)^2 + (Y_1 - Y_2)^2}$

VD: **b(4,3)** và **c(1,7)** thì độ dài cạnh **bc** là:

$$=\sqrt{(4-1)^2+(3-7)^2}=\sqrt{(3)^2+(-4)^2}=\sqrt{25}=5$$

CANTHO UNIVERSITY

TSP: Ví dụ



Tổng độ dài chu trình:

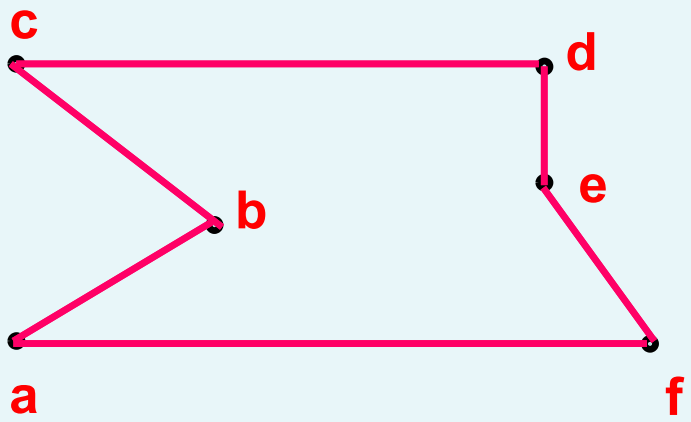
(a-b-c-d-e-f-a)

de + ab + bc + ef + cd + af

= 3 + 5 + 5 + 5 + 14 + 18 = **50**

TT	Cạnh	X_1	Y_1	X_2	Y_2	Độ dài
1	de	15	7	15	4	3.00
2	ab	0	0	4	3	5.00
3	bc	4	3	1	7	5.00
4	ef	15	4	18	0	5.00
5	ac	0	0	1	7	7.07
6	df	15	7	18	0	7.62
7	be	4	3	15	4	11.05
8	bd	4	3	15	7	11.70
9	cd	1	7	15	7	14.00
10	bf	4	3	18	0	14.32
11	ce	1	7	15	4	14.32
12	ae	0	0	15	4	15.52
13	ad	0	0	15	7	16.55
14	af	0	0	18	0	18.00
15	cf	1	7	18	0	18.38

TSP: Kết quả

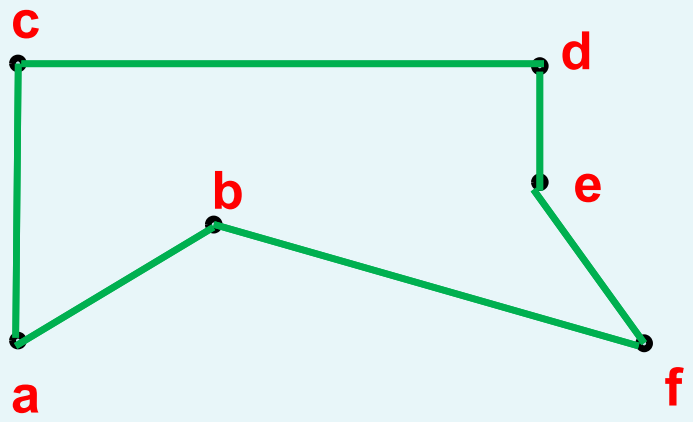


Phương án tốt (Greedy)

a-b-c-d-e-f-a = 50

- Số phép chọn: **n(n-1)/2**

$\Rightarrow$ T(n) = **O(n^2)**



Phương án tối ưu (Vét cạn)

a-c-d-e-f-b-a = **48.39**

- Số chu trình: **(n-1)! /2**

$\Rightarrow$ T(n) = **O(n!)**

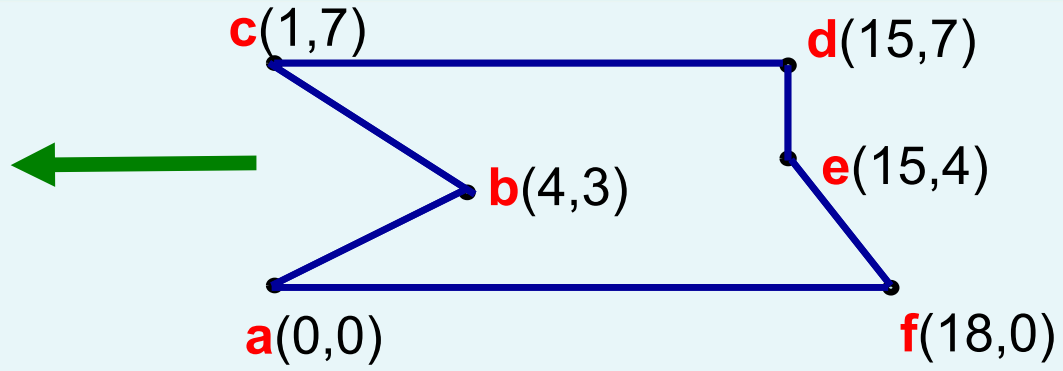
TSP: Thuật toán tham ăn

```
void TSP( ) {
    /*E là tập hợp các cạnh, Chu_trinh là tập hợp các cạnh được chọn để đưa
    vào chu trình, khởi đầu Chu_trinh rỗng*/
    /*Sắp xếp các cạnh trong E theo thứ tự tăng của độ dài*/

    Chu_Trinh = Φ;
    Gia = 0.0;
    while (E != Φ) {
            if (cạnh e có thể chọn) {
                Chu_Trinh = Chu_Trinh + [e];
                Gia = Gia + độ dài của e; }
            E := E - [e];
            }
```

CANTHO UNIVERSITY

MA TRẬN TRỌNG SỐ ĐỘ DÀI CẠNH

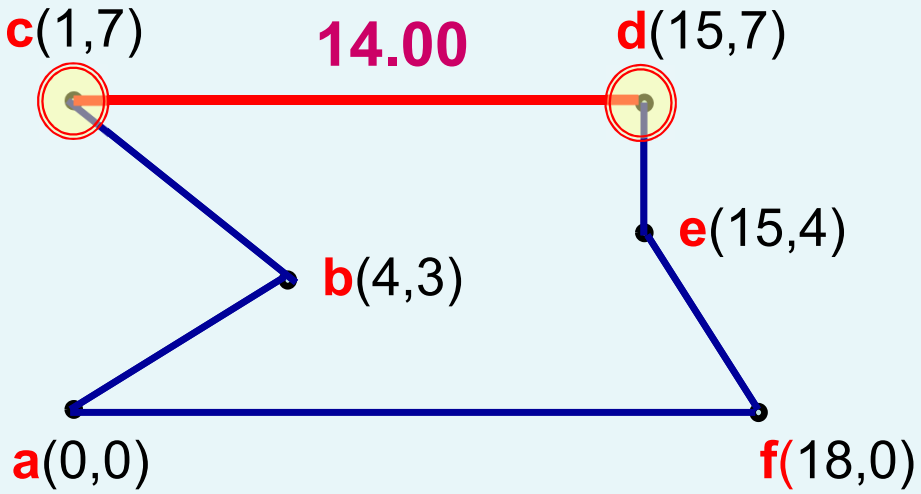
TT	Cạnh	X_1	Y_1	X_2	Y_2	Độ dài
1	de	15	7	15	4	3.00
2	ab	0	0	4	3	5.00
3	bc	4	3	1	7	5.00
4	ef	15	4	18	0	5.00
5	ac	0	0	1	7	7.07
6	df	15	7	18	0	7.62
7	be	4	3	15	4	11.05
8	bd	4	3	15	7	11.70
9	cd	1	7	15	7	14.00
10	bf	4	3	18	0	14.32
11	ce	1	7	15	4	14.32
12	ae	0	0	15	4	15.52
13	ad	0	0	15	7	16.55
14	af	0	0	18	0	18.00
15	cf	1	7	18	0	18.38



	a	b	c	d	e	f
	6					
a	0	5.00	7.07	16.55	15.52	18.00
b	5.00	0	5.00	11.70	11.05	14.32
c	7.07	5.00	0	14.00	14.32	18.38
d	16.55	11.70	14.00	0	3.00	7.62
e	15.52	11.05	14.32	3.00	0	5.00
f	18.00	14.32	18.38	7.62	5.00	0

KHAI BÁO CẠNH

```
typedef struct {
        float do_dai;
        int dau, cuoi;
} canh;
```



Số đỉnh = **n** ⇒ Số cạnh = **n(n-1)/2**

VD: n = 6 ⇒ 6(6-1)/2 = 15 cạnh

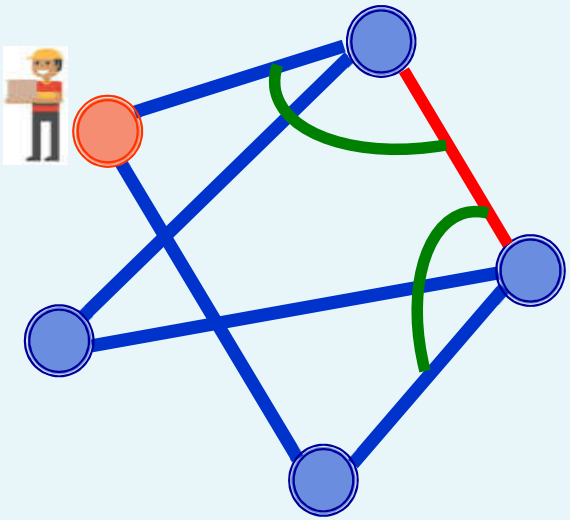
THIẾT KẾ HÀM KIỂM TRA ĐỈNH CẤP 3

Giả sử ta đã xây dựng được một phương án thành phần (PA), có k cạnh → Cần thêm vào cạnh mới.

- Xét các cạnh của PA, *từ 1 đến k*:

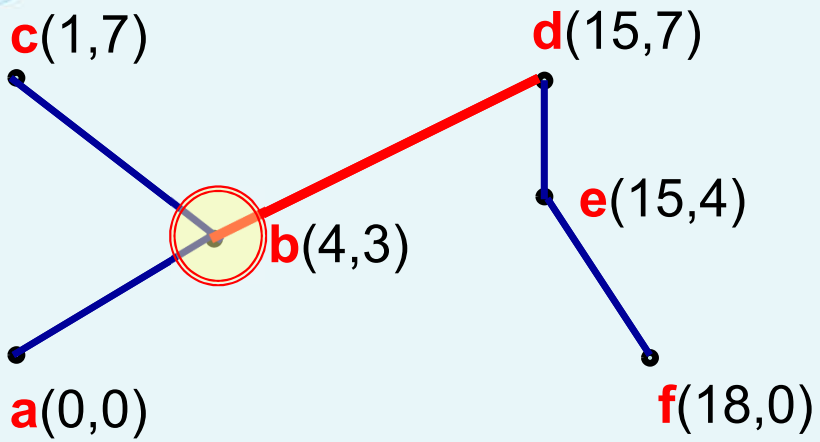
- Nếu đầu/cuối của cạnh trong PA trùng 2 lần với **đầu** của cạnh mới ⇒ **đỉnh cấp 3**

- Nếu đầu/cuối của cạnh trong PA trùng 2 lần với **cuối** của cạnh mới ⇒ **đỉnh cấp 3**



CANTHO UNIVERSITY

VÍ DỤ ĐỈNH CẤP 3



. PA có 4 cạnh: **de, ab, bc, ef**
→ Cần thêm cạnh **bd**

. *dem = 1*, xét đầu **bd** là **b**

- so sánh **b** với đầu và cuối **de**
- so sánh **b** với đầu và cuối **ab**
  ⇒ *dem = 2*
- so sánh **b** với đầu và cuối **bc**
  ⇒ *dem = 3*
- ngưng vì *dem = 3*
- vì *dem = 3* ⇒ có đỉnh cấp 3

HÀM KIỂM TRA ĐỈNH CẤP 3

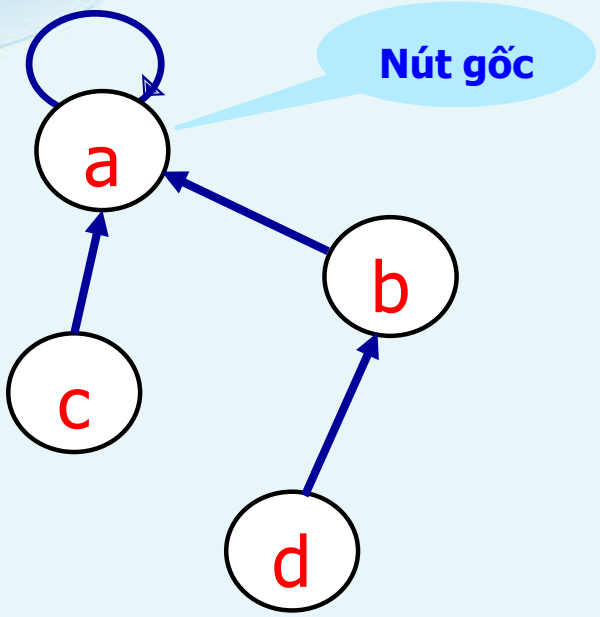
```
int dinh_cap3(canh PA[], int k, canh moi){
    int i,dem;
    i=0;
    dem=1;
    while (i<k && dem <3){
        if (moi.dau==PA[i].dau || moi.dau==PA[i].cuoi)
        dem++;
    i++;
    }
    if (dem==3) return 1;

    i=0;
    dem=1;
    while (i<k && dem <3){
        if (moi.cuoi==PA[i].dau || moi.cuoi==PA[i].cuoi)
        dem++;
    i++;
    }
    return dem==3;
}
```

THIẾT KẾ HÀM KIỂM TRA CHU TRÌNH: KIỂM SOÁT BẰNG RỪNG CÂY

Khởi đầu rừng có **n cây**, mỗi cây chỉ bao gồm **1 nút**.

- Cạnh được chọn vào PA thì **hợp nhất 2 cây** chứa 2 đỉnh của cạnh đó
- Nếu 2 đỉnh của một cạnh cùng thuộc 1 cây thì thêm cạnh đó vào PA sẽ tạo thành **chu trình.**
- Nếu 2 đỉnh của một cạnh thuộc 2 cây khác nhau thì không tạo chu trình thiếu $\Rightarrow$ có thể chọn cạnh vào PA

TỔ CHỨC CÂY



. **parent**[a] = a; **parent**[c] = a

. **parent**[b] = a; **parent**[d] = b

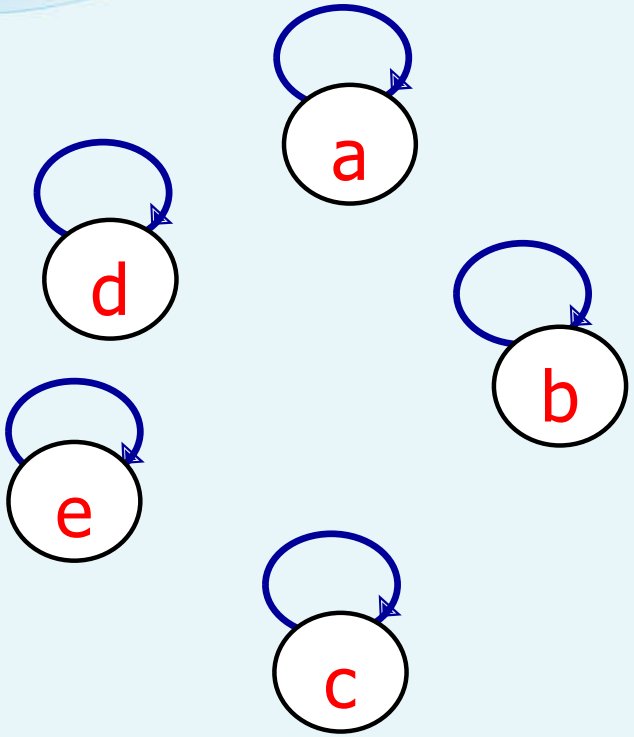
. Mỗi cây đại diện bởi **nút gốc**

. Mỗi nút của cây đều hướng về **nút cha** (*parent*) của nó.

. Nút gốc là cha của chính nó

⇒ Lưu trữ cây trong mảng **parent**, mỗi phần tử lưu trữ cha của nó.

TỔ CHỨC RỪNG CÂY



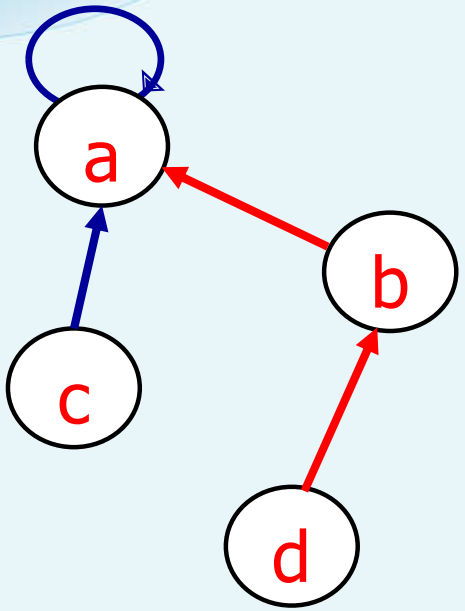
. Rừng cũng chính là mảng **parent**

. Khởi đầu mỗi phần tử của **parent** là một cây chỉ gồm một nút.

```
void init_forest(int parent[], int n){
    int i;
    for(i=0; i<n; i++)
        parent[i]=i;
}
```

CANTHO UNIVERSITY

HAI NÚT THUỘC MỘT CÂY



→ **c và d thuộc cùng 1 cây**

. 2 nút thuộc cùng 1 cây ↔ chúng có cùng gốc

. Để xác định gốc của nút, ta tìm về tổ tiên của nó (lần tìm theo nút cha), cho đến **nút có cha là chính nó**.

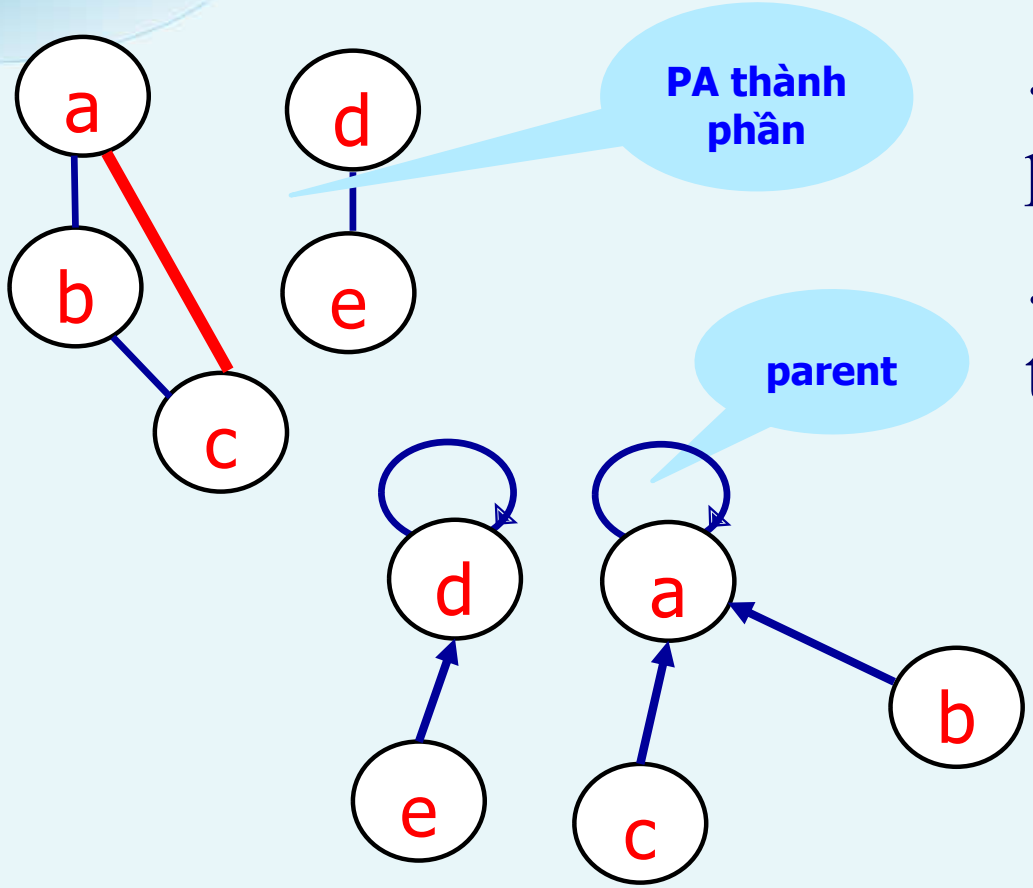
. **parent**[c] = a

. **parent**[d] = b; **parent**[b] = a; **parent**[a] = a;

```
int find_root(int parent[], int u){
    while (u != parent[u])
        u = parent[u];
        return u;
}
```

CANTHO UNIVERSITY

THÊM MỘT CẠNH TẠO CHU TRÌNH



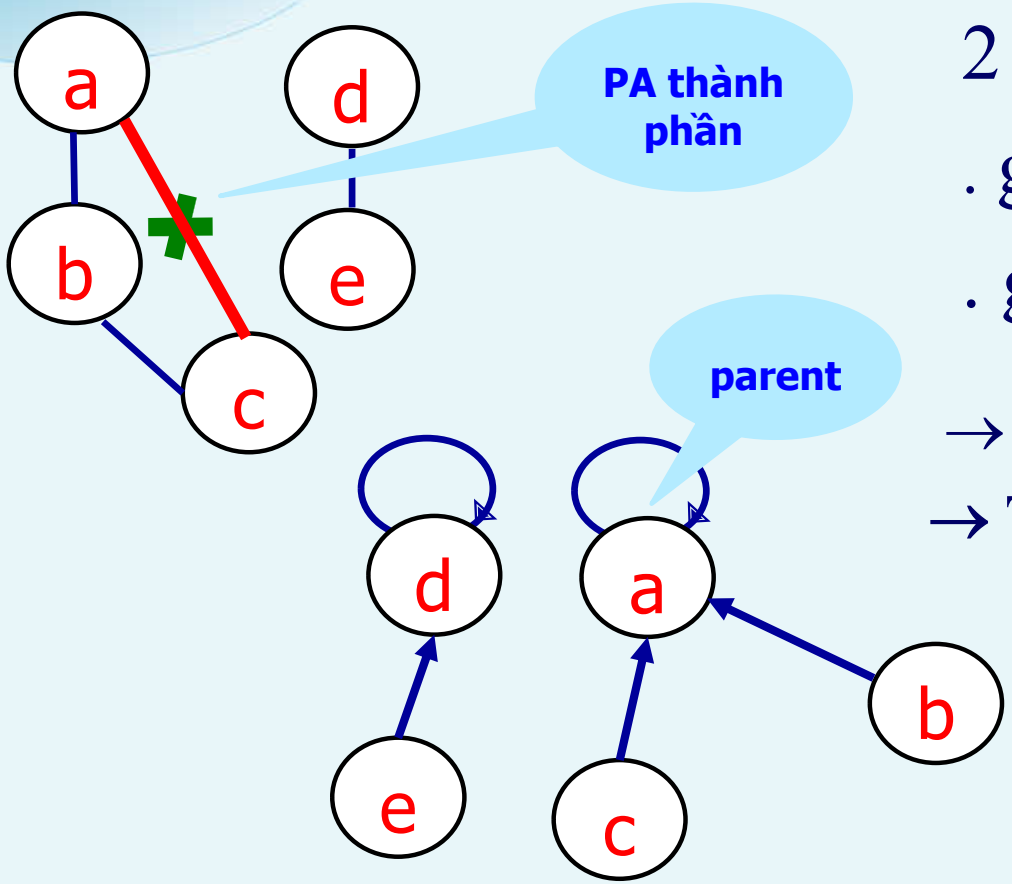
. Đã xây dựng PA thành phần gồm 3 cạnh **de, ab, bc**

. Nếu thêm cạnh **(a,c)** sẽ tạo thành **chu trình**. Vì sao ?

→ Vì 2 đỉnh của cạnh **(a, c)** nằm trên cùng 1 cây (có chung 1 gốc)

CANTHO UNIVERSITY

HÀM KIỂM TRA TẠO CHU TRÌNH



2 đỉnh **a, c** có chung 1 gốc:

. gốc của **a** là **a** (**parent**[a] = **a**)

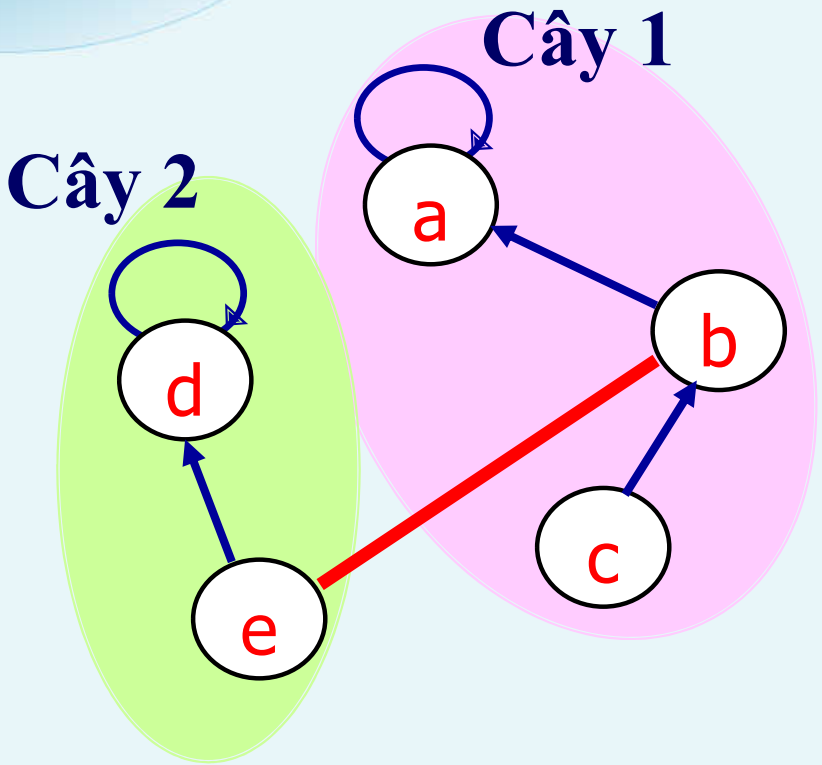
. gốc của **c** là **a** (**parent**[c] = **a**)

→ **a và c có cùng gốc → thuộc cùng cây**

→ **Thêm ac sẽ tạo thành chu trình thiếu**

```
int chu_trinh(int r_dau, int r_cuoi){
    return (r_dau == r_cuoi);
}
```

HỢP NHẤT 2 CÂY



. **Cây 1** có gốc là a

. **Cây 2** có gốc là d

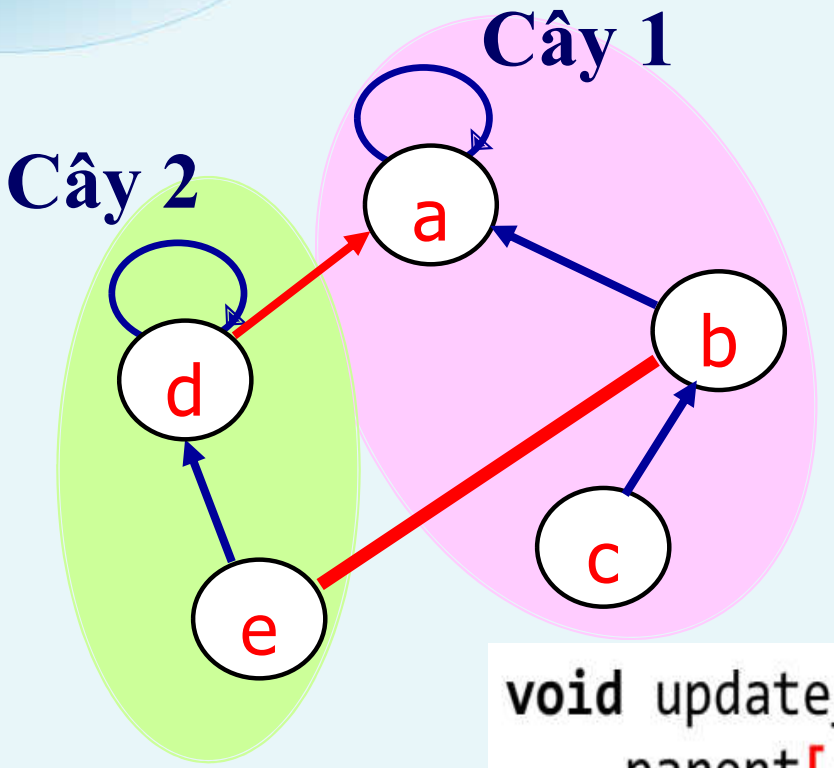
. Thêm cạnh **(b,e)** vào đường đi

→ Lấy gốc của **Cây 1** làm gốc của Cây 2, *hoặc:*

→ Lấy gốc của **Cây 2** làm gốc của Cây 1

CANTHO UNIVERSITY

HỢP NHẤT 2 CÂY



. **Cây 1** có gốc là **a**

. **Cây 2** có gốc là **d**

. Thêm cạnh **(b,e)** vào đường đi

→ Lấy gốc của **Cây 1** làm gốc của Cây 2 : **parent**[d] = a;

```
void update_forest(int parent[], int r1, int r2){
    parent[r2]= r1; //Hop nhat hai cay voi nhau
}
```