

# 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

# C BASIC

ĐỆ QUY QUAY LUI

ONE LOVE. ONE FUTURE.

# NỘI DUNG

- Bài toán Sudoku (P.02.06.01)
- Bài toán Queen (P.02.06.02)
- Bài toán TSP (P.02.06.03)



# BÀI TOÁN SUDOKU (P.02.06.01)

- Cho bảng vuông  $9 \times 9$  được chia thành 81 ô vuông con, đồng thời bảng cũng được chia thành 9 bảng vuông con mỗi bảng kích thước  $3 \times 3$  (xem Hình bên). Một số ô của bảng đã được điền 1 số nguyên từ 1 đến 9. Hãy điền vào các ô còn lại (ô có giá trị 0), mỗi ô một giá trị từ 1 đến 9 thỏa mãn: các số trên mỗi hàng, mỗi cột và mỗi bảng vuông con  $3 \times 3$  đều khác nhau
- Dữ liệu
  - 9 dòng, mỗi dòng là 9 phần tử của 1 hàng trên bảng
- Kết quả
  - Ghi ra số lượng phương án điền số

1	0	0	4	0	0	7	0	9
0	5	0	0	0	0	0	2	0
0	8	9	1	2	3	4	0	6
2	0	4	3	6	5	8	0	7
0	6	5	8	0	0	2	1	4
8	9	7	2	1	4	3	6	5
0	0	0	6	0	2	9	0	8
6	0	0	9	7	8	5	0	1
0	0	0	0	0	1	6	0	0

# BÀI TOÁN SUDOKU

stdin	stdout
0 0 3 4 0 0 0 8 9 0 0 6 7 8 9 0 2 3 0 8 0 0 2 3 4 5 6 0 0 4 0 6 5 0 9 7 0 6 0 0 9 0 0 1 4 0 0 7 2 0 4 3 6 5 0 3 0 6 0 2 0 7 8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	64

1	0	0	4	0	0	7	0	9
0	5	0	0	0	0	0	2	0
0	8	9	1	2	3	4	0	6
2	0	4	3	6	5	8	0	7
0	6	5	8	0	0	2	1	4
8	9	7	2	1	4	3	6	5
0	0	0	6	0	2	9	0	8
6	0	0	9	7	8	5	0	1
0	0	0	0	0	1	6	0	0



# BÀI TOÁN SUDOKU – THIẾT KẾ THUẬT TOÁN, MÃ GIẢ

- Đánh số thứ tự
  - Các hàng, cột của bảng được đánh số thứ tự 0, 1, ..., 8
  - Mỗi bảng vuông con  $3 \times 3$  được đặc trưng bởi chỉ số theo hàng  $i$  và theo cột  $j$  (mỗi chỉ số hàng  $i$ , cột  $j$  này tương ứng với 3 hàng liên tiếp và 3 cột liên tiếp của bảng,  $i, j = 0, 1, 2$ )
- Biểu diễn phương án:  $X[0..8, 0..8]$
- Mảng đánh dấu:
  - $\text{markR}[r, v] = 1$ : nghĩa là giá trị  $v$  đã xuất hiện trên hàng  $r$ , với  $r = 0, \dots, 8, v = 1, \dots, 9$
  - $\text{markC}[c, v] = 1$ : nghĩa là giá trị  $v$  đã xuất hiện trên cột  $c$ , với  $c = 0, \dots, 8, v = 1, \dots, 9$
  - $\text{MarkS}[i, j, v] = 1$ : nghĩa là giá trị  $v$  đã xuất hiện trên bảng vuông con  $3 \times 3$  ở tọa độ  $(i, j)$ , với  $i, j = 0, 1, 2, v = 1, \dots, 9$

	0	1	2	3	4	5	6	7	8
0	1	0	0	4	0	0	7	0	9
1	0	5	0	0	0	0	0	2	0
2	0	8	9	1	2	3	4	0	6
3	2	0	4	3	6	5	8	0	7
4	0	6	5	8	0	0	2	1	4
5	8	9	7	2	1	4	3	6	5
6	0	0	0	6	0	2	9	0	8
7	6	0	0	9	7	8	5	0	1
8	0	0	0	0	0	1	6	0	0

The table represents a 9x9 Sudoku grid. Red lines divide the grid into 3x3 subgrids. Red numbers at the bottom indicate row and column indices. Red curly braces on the right indicate subgrid indices.

Row indices: 0, 1, 2, 3, 4, 5, 6, 7, 8

Column indices: 0, 1, 2, 3, 4, 5, 6, 7, 8

Subgrid indices: 0, 1, 2

# BÀI TOÁN SUDOKU – THIẾT KẾ THUẬT TOÁN, MÃ GIẢ

- Thứ tự duyệt các biến: từ trên xuống dưới và từ trái qua phải
- Hàm try(r, c): thử giá trị cho X[r, c]
  - Xét các giá trị v từ 1 đến 9
- Hàm check(v, r, c):
  - Giá trị v chỉ hợp lệ khi nó chưa xuất hiện
  - Trên hàng r: markR[r, v] = 0
  - Trên cột c: markC[c, v] = 0
  - Trên bảng 3 x 3 ở tọa độ (r/3, c/3): markS[r/3, c/3, v] = 0

```
try(r, c){  
    if x[r, c] > 0 then {  
        if r = 8 and c = 8 then solution();  
        else { if c = 8 then try(r+1, 0); else try(r, c+1); }  
        return;  
    }  
    for v = 1 to 9 do {  
        if check(v, r, c) then {  
            x[r, c] = v;  
            markR[r,v] = 1; markC[c,v] = 1; markS[r/3,c/3,v] = 1;  
            if r = 8 and c = 8 then solution();  
            else { if c = 8 then try(r+1, 0); else try(r, c+1); }  
            markR[r,v] = 0; markC[c,v] = 0; markS[r/3,c/3,v] = 0;  
            x[r, c] = 0;  
        }  
    }  
}
```



# BÀI TOÁN SUDOKU – CODE

```
#include <stdio.h>
int X[9][9];
int markR[9][10];
int markC[9][10];
int markS[3][3][10];
int cnt;
int check(int v, int r, int c){
    if(markR[r][v]) return 0;
    if(markC[c][v]) return 0;
    if(markS[r/3][c/3][v]) return 0;
    return 1;
}
void Try(int r, int c){
    if(X[r][c] > 0){
        if(r == 8 && c == 8) cnt++;
        else{ if(c == 8) Try(r+1,0); else Try(r,c+1); }
        return;
    }
    for(int v = 1; v <= 9; v++){
        if(check(v,r,c)){
            X[r][c] = v;
            markR[r][v] = 1; markC[c][v] = 1; markS[r/3][c/3][v] = 1;
            if(r == 8 && c == 8) cnt++;
            else{ if(c == 8) Try(r+1,0); else Try(r,c+1); }
            markR[r][v] = 0; markC[c][v] = 0; markS[r/3][c/3][v] = 0;
            X[r][c] = 0;
        }
    }
}
```



# BÀI TOÁN SUDOKU – CODE

```
void init(){
    for(int v = 1; v <= 9; v++){
        for(int r = 0; r <= 8; r++)
            markR[r][v] = 0;
        for(int c = 0; c <= 8; c++)
            markC[c][v] = 0;
        for(int i = 0; i <= 2; i++)
            for(int j = 0; j <= 2; j++)
                markS[i][j][v] = 0;
    }
}
```

```
int main(){
    init();
    for(int r = 0;r <= 8; r++){
        for(int c = 0; c <= 8; c++){
            int v;      scanf("%d",&v);
            X[r][c] = v;
            if(v > 0){
                markR[r][v] = 1;    markC[c][v] = 1;
                markS[r/3][c/3][v] = 1;
            }
        }
        cnt = 0;
        Try(0,0);
        printf("%d",cnt);
        return 0;
    }
}
```



# BÀI TOÁN QUEEN (P.02.06.02)

- Trên bàn cờ quốc tế kích thước  $n \times n$ , có  $k$  quân hậu ( $0 \leq k < n$ ). Trạng thái bàn cờ được biểu diễn bởi ma trận  $A_{nxn}$  trong đó  $A(i, j) = 1$  nghĩa là ô hàng  $i$  cột  $j$  có quân hậu và  $A(i, j) = 0$  nghĩa là ô hàng  $i$ , cột  $j$  không có quân hậu. Hãy đếm số lượng  $Q$  các cách đặt  $n - k$  quân hậu khác trên bàn cờ sao cho không có 2 quân hậu nào ăn được nhau
- Dữ liệu
  - Dòng 1 ghi số nguyên  $n$  ( $1 \leq n \leq 15$ )
  - Dòng  $i + 1$  ( $i = 1, 2, \dots, n$ ): ghi hàng thứ  $i$  của ma trận  $A$
- Kết quả
  - Ghi ra giá trị  $Q$

stdin	stdout
8	3
00000000	
00000000	
01000000	
00000000	
00000000	
00000000	
00010000	
00000000	
00000000	



# BÀI TOÁN QUEEN – THIẾT KẾ THUẬT TOÁN, MÃ GIẢ

- Biểu diễn phương án:  $x[1..n]$ , trong đó  $x[i]$  là chỉ số hàng của quân hậu trên cột  $i$
- Ràng buộc
  - $x[i] \neq x[j]$
  - $x[i] + i \neq x[j] + j$
  - $x[i] - i \neq x[j] - j$
- Mảng đánh dấu:
  - $\text{markR}[r] = 1$ : hàng  $r$  có quân hậu
  - $\text{markD1}[d] = 1$ : nghĩa là có quân hậu trên hàng  $r$ , cột  $k$  với  $d = n - k + r$
  - $\text{MarkD2}[d] = 1$ : nghĩa là có quân hậu trên hàng  $r$ , cột  $k$  với  $k + r = d$

```
check(r, k){  
    return (mark[r] = 0) and (markD1[n+k-r] = 0) and (markD2[k+r] = 0);  
}  
  
try(k){  
    if x[k] > 0 then {  
        if k = n then cnt = cnt + 1; else try(k+1);  
        return;  
    }  
    for r = 1 to n do {  
        if check(r, k) then {  
            x[k] = r; mark[r] = 1; markD1[n+k-r] = 1; markD2[k+r] = 1;  
            if k = n then cnt = cnt + 1;  
            else try(k+1);  
            x[k] = 0; mark[r] = 0; markD1[n+k-r] = 0; markD2[k+r] = 0;  
        }  
    }  
}
```



# BÀI TOÁN QUEEN – CODE

```
#include <stdio.h>
#define N 100
int n;
int x[N];
int a[N];
int mark[N];
int markD1[2*N];
int markD2[2*N];
int cnt;
int check(int v, int k){
    if(mark[v] == 1) return 0;
    if(markD1[n + k - v]==1) return 0;
    if(markD2[k + v]==1) return 0;
    return 1;
}
```

```
void Try(int k){
    if(x[k] > 0){
        if(k == n) cnt += 1; else Try(k+1);
        return;
    }
    for(int r = 1; r <= n; r++){
        if(check(r,k)){
            x[k] = r; mark[r] = 1; markD1[n+ k - r] = 1; markD2[k + r] = 1;
            if(k == n) cnt += 1; else Try(k+1);
            x[k] = 0; mark[r] = 0; markD1[n+ k - r] = 0; markD2[k + r] = 0;
        }
    }
}
```



# BÀI TOÁN QUEEN – CODE

```
void input(){
    for(int i = 1; i < N; i++) mark[i] = 0;
    for(int i = 0; i < 2*N; i++){    markD1[i] = 0; markD2[i] = 0;  }
    scanf("%d",&n);
    for(int i = 1; i <= n; i++) x[i] = 0;
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= n; j++){
            int e;  scanf("%d",&e);
            if(e == 1){
                x[j] = i;
                mark[i] = 1; markD1[n + j - i] = 1; markD2[i+j] = 1;
            }
        }
    }
}
```

```
int main(){
    input();
    cnt = 0;
    Try(1);
    printf("%d",cnt);
    return 0;
}
```



# BÀI TOÁN TSP (P.02.06.03)

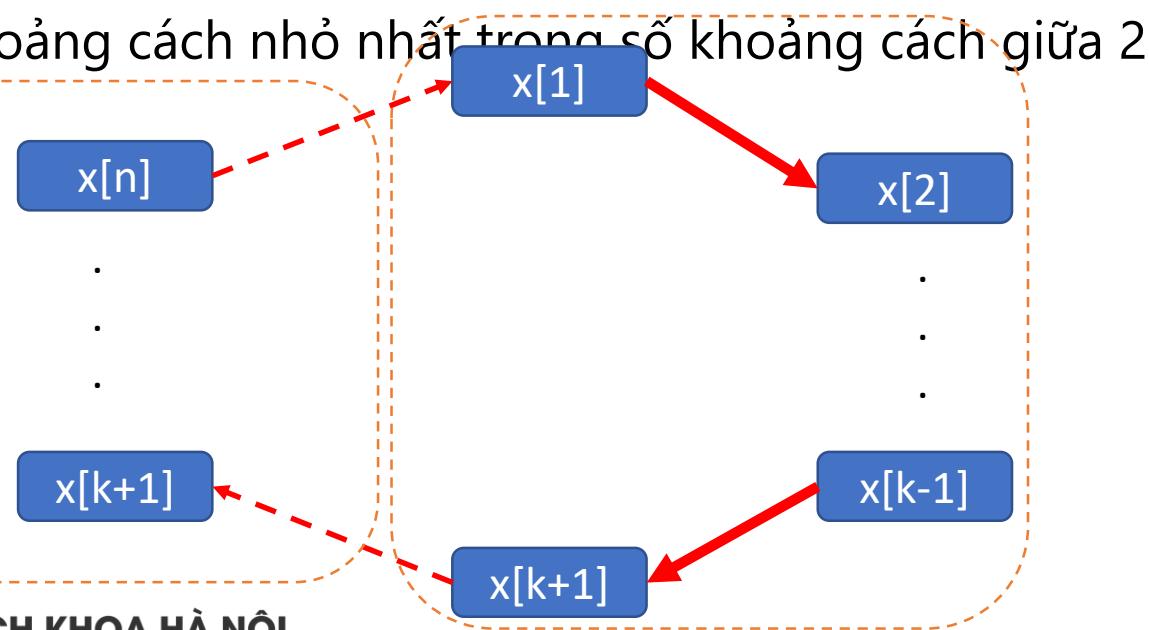
- Cho  $n$  điểm  $1, 2, \dots, n$ . Khoảng cách di chuyển từ điểm  $i$  đến điểm  $j$  là  $d(i, j)$ , với  $i, j = 1, 2, \dots, n$ . Hãy tìm chu trình xuất phát từ điểm 1, đi qua các điểm khác, mỗi điểm đúng 1 lần và quay về điểm 1 với tổng độ dài nhỏ nhất
- Dữ liệu
  - Dòng 1 ghi số nguyên  $n$  ( $1 \leq n \leq 20$ )
  - Dòng  $i + 1$  ( $i = 1, 2, \dots, n$ ): ghi hàng thứ  $i$  của ma trận  $d$
- Kết quả
  - Ghi ra độ dài của chu trình tìm được

stdin	stdout
4 0 1 1 9 1 0 9 3 1 9 0 2 9 3 2 0	7



# BÀI TOÁN TSP – THIẾT KẾ THUẬT TOÁN, MÃ GIẢ

- Biểu diễn phương án:  $x[1, \dots, n]$  trong đó  $x[i]$  là điểm thứ  $i$  trên hành trình,  $i = 1, 2, \dots, n$ . Chu trình sẽ là  $x[1] \rightarrow x[2] \rightarrow \dots \rightarrow x[n] \rightarrow x[1]$
- Mảng đánh dấu:
  - $\text{mark}[v] = 1$ : nghĩa là điểm  $v$  đã xuất hiện trên hành trình
- Nhánh và cận
  - Cm: khoảng cách nhỏ nhất trong số khoảng cách giữa 2 điểm

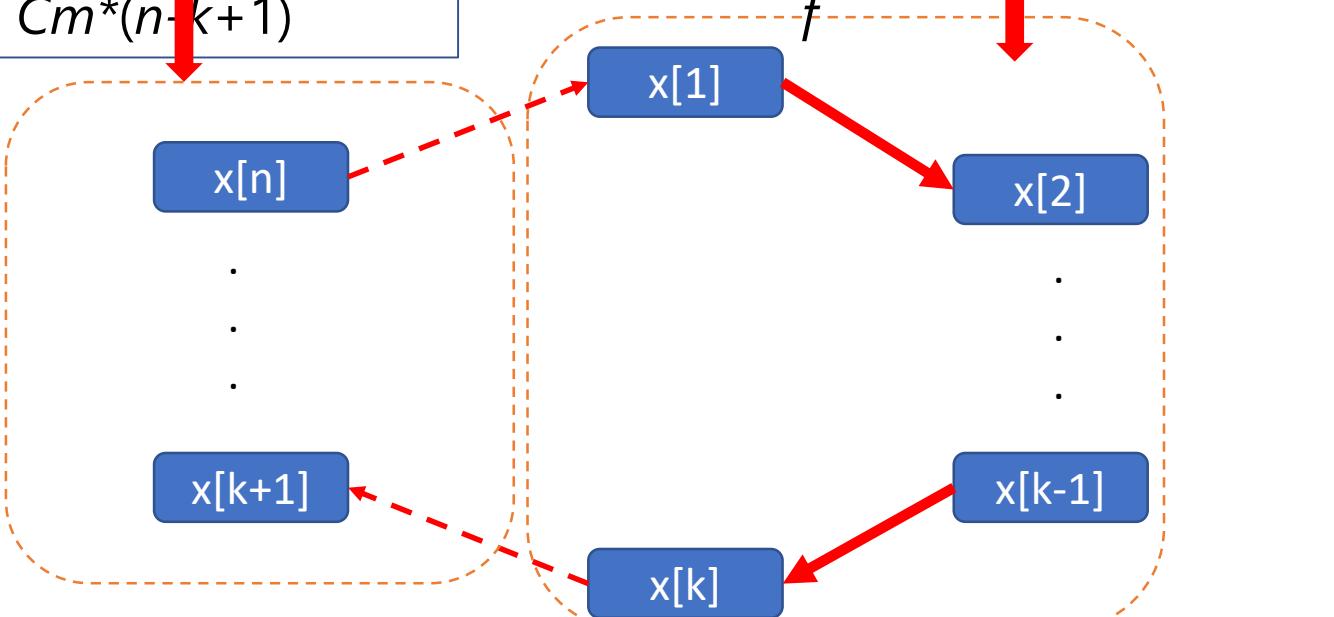


```
try(k){  
    for v = 1 to n do {  
        if mark[v] = 0 then {  
            x[k] = v;  
            f = f + d[x[k-1], v]; mark[v] = 1;  
            if k = n then {  
                if fmin > f + d[x[n],x[1]] then  
                    fmin = f + d[x[n],x[1]];  
            } else{  
                if f + Cm*(n-k+1) < fmin then  
                    try(k+1);  
            }  
            f = f - d[x[k-1], v]; mark[v] = 0;  
        }  
    }  
}
```

# BÀI TOÁN TSP – THIẾT KẾ THUẬT TOÁN, MÃ GIẢ

- $Cm$ : khoảng cách nhỏ nhất trong số khoảng cách giữa 2 điểm

Hành trình chưa đi qua, gồm  $n-k+1$  chặng độ dài  $\geq Cm^*(n-k+1)$



Cận dưới độ dài hành trình phát triển tiếp từ  $x[1], x[2], \dots, x[k]$  là:  $f + Cm^*(n-k+1)$

```
try(k){  
    for v = 1 to n do {  
        if mark[v] = 0 then {  
            x[k] = v;  
            f = f + d[x[k-1], v]; mark[v] = 1;  
            if k = n then {  
                if fmin > f + d[x[n],x[1]] then  
                    fmin = f + d[x[n],x[1]];  
            }else{  
                if f + Cm*(n-k+1) < fmin then  
                    try(k+1);  
            }  
            f = f - d[x[k-1], v]; mark[v] = 0;  
        }  
    }  
}
```

# BÀI TOÁN TSP – CODE

```
#include <stdio.h>

int n;// so thanh pho

int d[100][100];

int x[100];

int mark[100];

int f;// do dai cua lo trinh di duoc

int f_min;// do dai lo trinh ngan nhat (ky luc)

int Cm;
```

```
void Try(int k){

    for(int v = 1; v<= n; v++){

        if(mark[v]==0){

            x[k] = v;

            f = f + d[x[k-1]][v]; mark[v] = 1;

            if(k == n){

                if(f_min > f + d[x[n]][x[1]])

                    f_min = f + d[x[n]][x[1]];

            }else{

                if (f + Cm*(n-k+1) < f_min)

                    Try(k+1);

            }

            mark[v] = 0; f = f - d[x[k-1]][v];

        }

    }

}
```

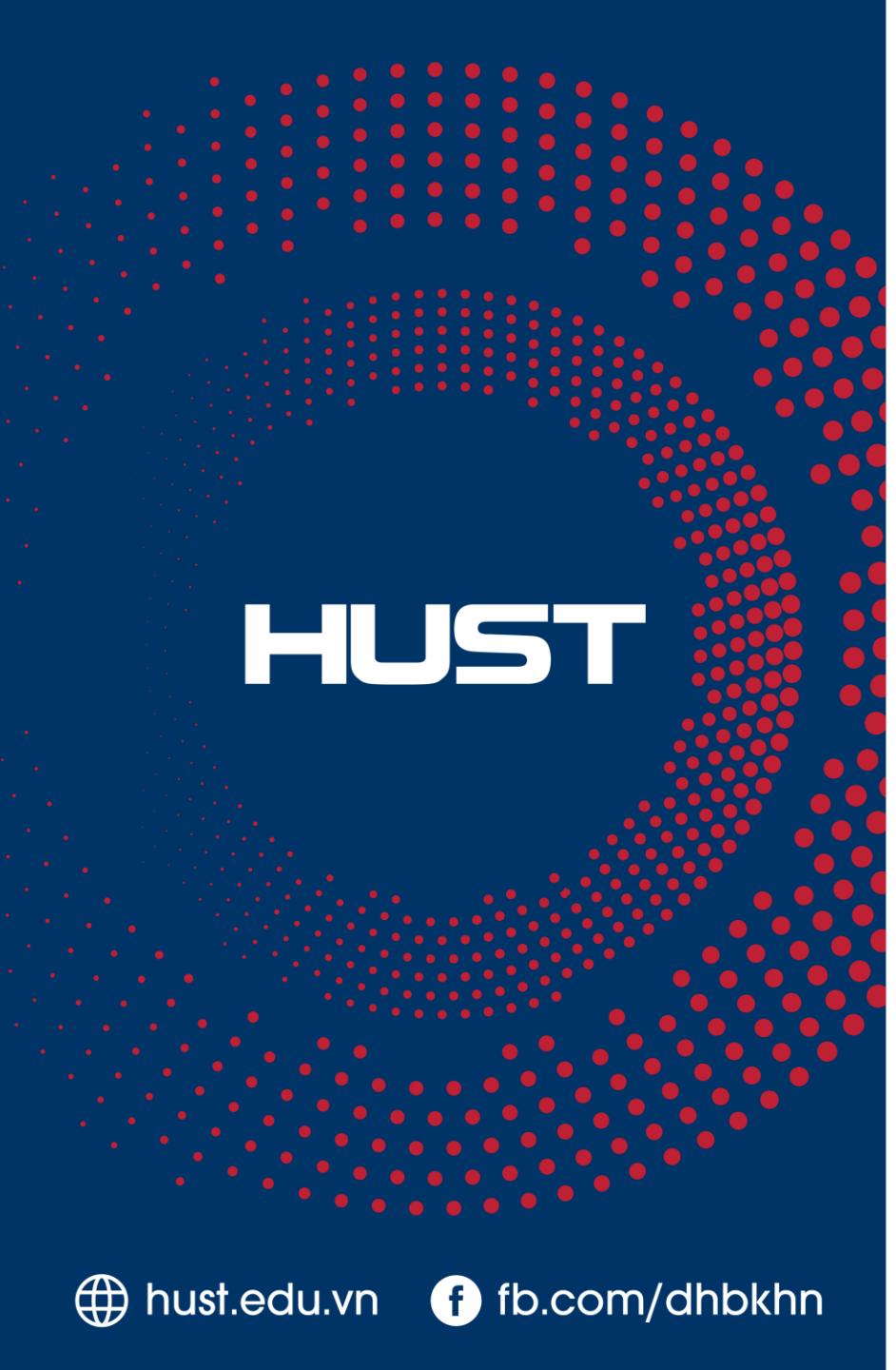


# BÀI TOÁN TSP – CODE

```
void input(){
    scanf("%d",&n);
    Cm = 10000000;
    for(int i = 1; i <= n; i++){
        for(int j =1; j <= n; j++){
            scanf("%d",&d[i][j]);
            if(i != j && d[i][j] < Cm) Cm = d[i][j];
        }
    }
}
```

```
int main(){
    input();
    for(int v = 1; v <= n; v++) mark[v] = 0;
    x[1] = 1; mark[1] = 1;
    f = 0; f_min = 10000000;
    Try(2);
    printf("%d",f_min);
    return 0;
}
```





**HUST**

**THANK YOU !**