HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

C BASIC

C BASIC

RECURSIVE BACKTRACKING

CONTENT

- Sudoku problem (P.02.06.01)
- Queen problem (P.02.06.02)
- TSP problem (P.02.06.03)

SUDOKU PROBLEM (P.02.06.01)

- Let the 9 x 9 square board be divided into 81 sub-squares, and the board is also divided into 9 sub-square panels each measuring 3 x 3 (see Figure below). Some cells of the table have been filled with an integer from 1 to 9. Fill in the remaining cells (cells with value 0), each cell with a value from 1 to 9 to satisfy: numbers on each row, each column, and each pair of 3 x 3 squares are different
- Data
  - 9 lines, each line is 9 elements of 1 row on the table
- Result
  - Write down the number of number filling options

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

SUDOKU PROBLEM

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

SUDOKU PROBLEM – PSEUDOCODE

- Numbering
  - The rows and columns of the table are numbered 0, 1, . . ., 8
  - Each 3 x 3 subsquare table is characterized by an index in row i and column j (each index in row i, column j corresponds to 3 consecutive rows and 3 consecutive columns of the table, i, j = 0 , 1, 2
- Representing solutions: X[0..8, 0..8]
- Marked array:
  - markR[*r*, *v*] = 1: value v does appear in row *r*, with *r* = 0, ..., 8, *v* = 1,..., 9
  - markC[*c*, *v*] = 1: value v does appear in *column c*, with *c* = 0,..., 8, *v* = 1,..., 9
  - MarkS[*i*, *j*, *v*] = 1: value v does appear in a subsquare 3 x 3 at coordinate (i, j), với *i*, *j* = 0, 1, 2, *v* = 1, ..., 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**	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**	1
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**	2
8	**0**	**0**	**0**	**0**	**0**	**1**	**6**	**0**	**0**	
		0			1			2		

SUDOKU PROBLEM – PSEUDOCODE

- Order to iterate: from top to down and from left to right
- Function try(r, c): try values for X[r, c]
  - Consider values of v from 1 to 9
- Function check(v, r, c):
  - Value v is valid when it has not appeared
  - Row r: markR[r, v] = 0
  - Column c: markC[c, v] = 0
  - Subsquare 3 x 3 at coordinate (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;
    }
  }
}
```

QUEEN PROBLEM (P.02.06.02)

- On an international chess board of size n x n, there are k queens (0 <= k < n). The state of the chessboard is represented by the matrix A_{nxn} in which A(i, j) = 1 means that row i, column j has a queen and A(i,j) = 0 means row i, column j does not have a queen. queen. Count the number Q of ways to place n - k other queens on the chessboard so that no two queens can attack each other.
- Data
  - Line 1: An integer *n* (1 <= *n* <= 15)
  - Line *i* + 1 (*i* = 1, 2, ..., *n*): row i^{th} of *A*
- Result
  - Write the value of *Q*

stdin	stdout
8 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	3

QUEEN PROBLEM – PSEUDOCODE

- Representing solutions: $x[1..n]$, where $x[i]$ is the row index of the queen in column i
- Constraints
  - $x[i] \neq x[j]$
  - $x[i] + i \neq x[j] + j$
  - $x[i] - i \neq x[j] - j$
- Marked arrays:
  - markR[r] = 1: row r has a queen
  - markD1[d] = 1: there is a queen in row r, column k with $d = n - k + r$
  - MarkD2[d] = 1: there is a queen in row r column k with $k + r = d$

```
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;
    }
  }
}
```

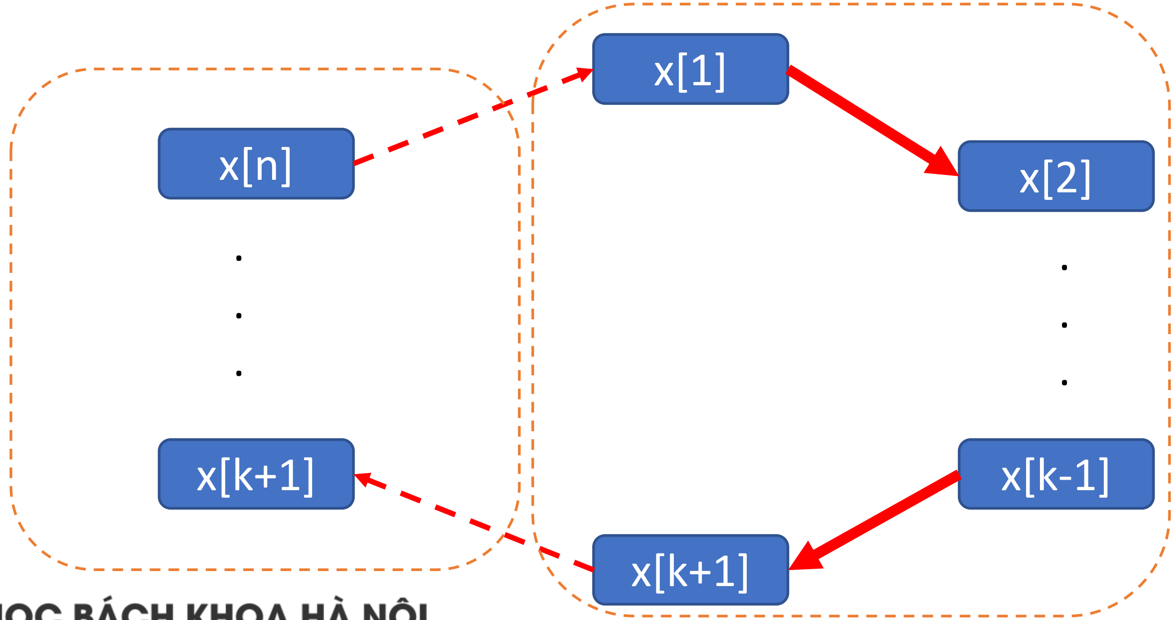
TSP PROBLEM (P.02.06.03)

- Given n points 1, 2, . . ., n. The travel distance from point i to point j is d(i, j), with i, j = 1, 2, …, n. Find the trip starting from point 1, passing through other points, each point exactly once and returning to point 1 with the smallest total length.
- Data
  - Line 1: An integer *n* (1 <= *n* <= 20)
  - Line *i* + 1 (*i* = 1, 2, …, *n*): Row i of the matrix d
- Result
  - The length of the found trip

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

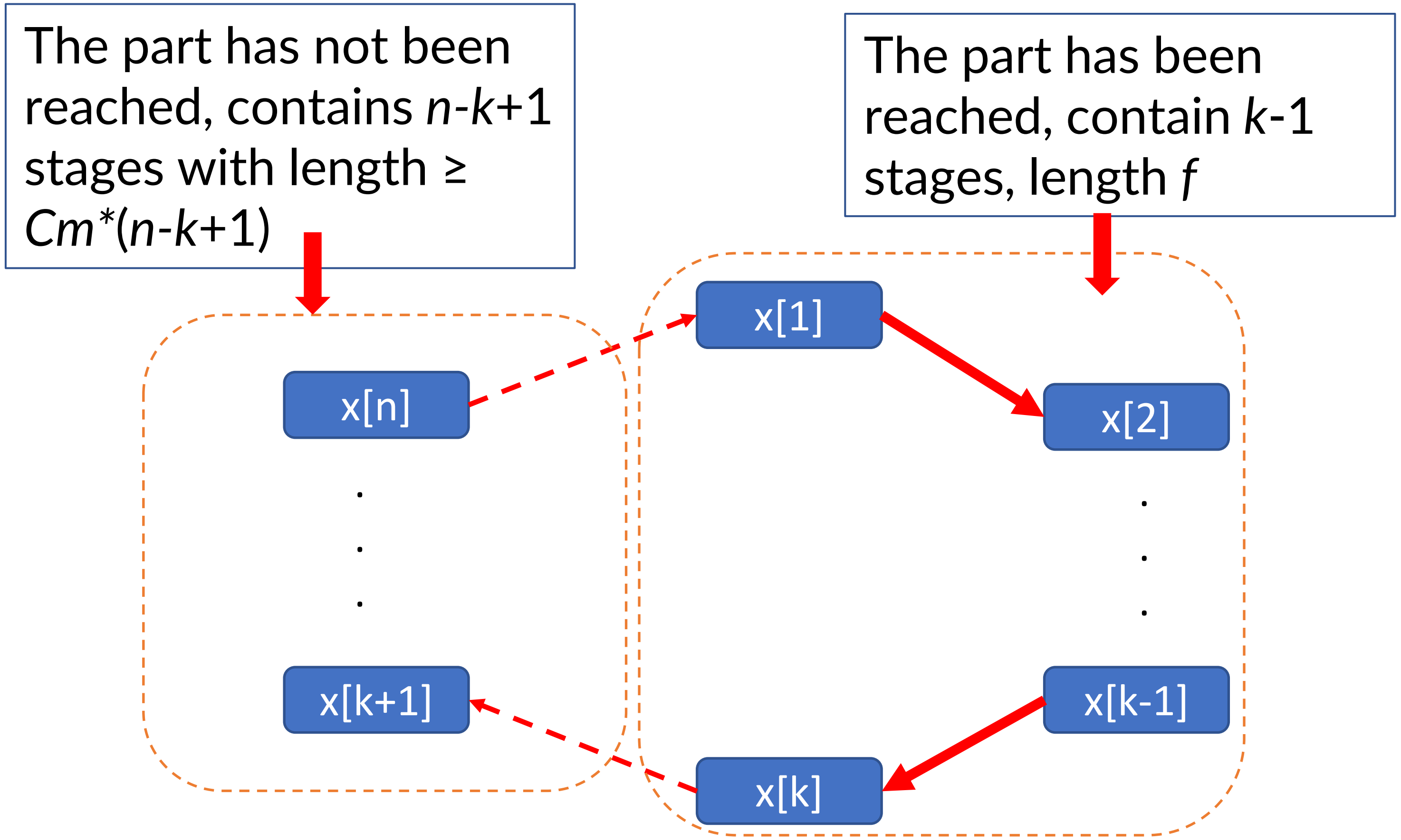
TSP PROBLEM - PSEUDOCODE

- Representing solutions: x[1, . . ., *n*], where x[*i*] is the *point i^th^* of the trip, *i* = 1, 2, ..., *n*. The trip is: x[1] → x[2] → . . . → x[n] → x[1]
- Marked array:
  - mark[v] = 1: v does appear in the trip
- Branch and Bound
  - Cm: the minimal distance between two points




```
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;
     }
  }
}
```

TSP PROBLEM - PSEUDOCODE



Lower bound $f + Cm^*(n-k+1)$

```
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;
     }
  }
}
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !