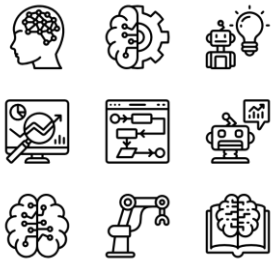
Computer Science for Practicing Engineers

Các phương pháp thiết kế thuật toán



TS. Huỳnh Bá Diệu
Email: dieuhb@gmail.com
Phone: 0914146868

1

Các phương pháp thiết kế thuật toán

1. **Phương pháp Vét cạn (Brute-force or exhaustive search)**
2. **Phương pháp quay lui (Backtracking)**
3. Phương pháp Chia để trị (Divide and Conquer)
4. Phương pháp Qui hoạch động (Dynamic Programming)
5. Phương pháp tham lam (Greedy Algorithms)
6. Phương pháp nhánh cận (Branch and Bound Algorithm)
7. *Phương pháp ngẫu nhiên (Randomized Algorithm)*

2

Phương pháp thiết kế vét cạn *(Brute-force or exhaustive search)*

Brute force is a straightforward approach to solving a problem, usually directly based on the problem statement and definitions of the concepts involved.

The "force" implied by the strategy's definition is that of a computer and not that of one's intellect. "Just do it!" would be another way to describe the prescription of the brute-force approach. And often, the brute-force strategy is indeed the one that is easiest to apply.

3

Phương pháp thiết kế vét cạn *(Brute-force or exhaustive search)*

- Giải bài toán bằng cách liệt kê tất cả các trường hợp là ứng viên của lời giải.
- Kiểm tra từng ứng viên để chọn ra lời giải.

Ưu điểm: Đơn giản, dễ chứng minh tính đúng đắn

Hạn chế: Không có chiến lược nào trong quá trình tìm lời giải.

Khi kích thước dữ liệu lớn thì thời gian giải là rất lâu do phải xét tất cả các ứng viên.

4

Phương pháp thiết kế vét cạn

Ví dụ: Tìm dãy con liên tiếp có tổng lớn nhất

- Liệt kê tất cả các dãy con có thể có *//tất cả các ứng viên*
- Tính tổng từng dãy con *// tính toán*
- So sánh và tìm ra dãy có tổng lớn nhất *// kiểm tra để tìm lời giải*

5

Phương pháp thiết kế vét cạn

Bài toán ví dụ:

Cho robot có thể thực hiện được 2 kiểu bước đi có độ dài là 1 m và 2 m. Cho đoạn đường AB dài ll m. Hỏi có bao nhiêu cách robot đi vừa hết đoạn đường AB nhưng thực hiện không quá 2 bước đi 1 m.

Giải pháp theo pp vét cạn:

- Liệt kê tất cả cách đi hết ll m
- Kiểm tra trong mỗi cách đi, có cách nào không đi quá 2 bước 1 thì in ra

ll=4 m

1 1 1 1

1 1 2

1 2 1

2 1 1

2 2

6

Phương pháp thiết kế vét cạn

Bài toán Robot

Liệt kê tất cả các cách đi:

Cần mảng m[] để lưu các bước đi

Biến sb lưu số bước.

Biến sc: số cách đi

7

Phương pháp thiết kế vét cạn: Robot

```
void phantich(int ll)  // các cách để đi hết đoạn đường độ dài ll
  {
     Nếu đã đi hết đoạn đường thì {đếm số bước đi độ dài 1, nếu ít hơn 3 thì in ra cách đi}
     Ngược lại:
        Duyệt các cách đi có thể
         Nếu như đoạn đường phải đi ít hơn hoặc bằng độ dài bước đi thì
         {
            Ghi nhận bước đi có độ dài được chọn;
            Phân tích tiếp cách đi độ dài đoạn đường còn lại
            Lùi lại 1 bước để chọn cách đi khác
         }
}
```

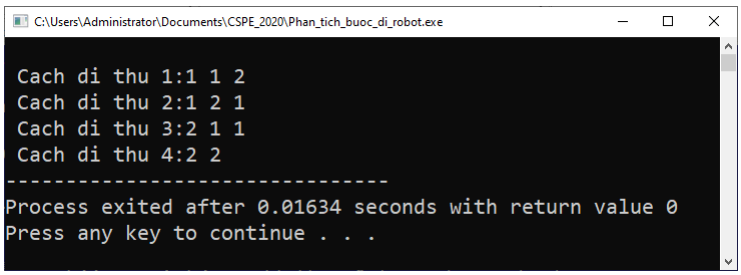
8

Phương pháp thiết kế vét cạn: Robot

```
void phantich(int ll)
  {
    if(ll==0)  {   int d=0;  for (int i=1; i<=sb; i++) if(m[i]==1) d++;   if (d<=2) in(); }
    else
      for(int i=1; i<=2; i++)  // co hai buoc co do dai la 1 va 2
         if(i<= ll)
         {
           sb++;   m[sb]=i; // them 1 buoc di co do dai i
           phantich(ll-i);  // di doan duong con lai
           sb--;          // lui lai 1 buoc de di buoc co do dai khac
         }
}
```

9

```
#include<iostream>
using namespace std;
int sb=0, dem=0, m[1000];
void in()
{
 cout<<"\n Cach di thu "<<++dem<<":";
 for(int i=1; i<=sb; i++) cout<<m[i]<<" ";
}
void phantich(int ll)    {  }
int main()
{
   int d=4;
   phantich(d);
}
```



10

Phương pháp thiết kế vét cạn: Robot ll=14

Cach di thu 1:	Cach di thu 7:	Cach di thu 13:	Cach di thu 19:	Cach di thu 25:
1 1 2 2 2 2 2 2	1 2 2 2 2 2 2 1	2 1 2 2 2 2 2 1	2 2 2 1 1 2 2 2	2 2 2 2 1 2 2 1
Cach di thu 2:	Cach di thu 8:	Cach di thu 14:	Cach di thu 20:	Cach di thu 26:
1 2 1 1 2 2 2 2	2 1 1 2 2 2 2 2	2 2 1 1 1 2 2 2	2 2 2 1 2 1 2 2	2 2 2 2 2 1 1 2
Cach di thu 3:	Cach di thu 9:	Cach di thu 15:	Cach di thu 21:	Cach di thu 27:
1 2 2 1 1 2 2 2	2 1 2 1 1 2 2 2	2 2 1 2 1 1 2 2	2 2 2 1 2 2 1 2	2 2 2 2 2 1 2 1
Cach di thu 4:	Cach di thu 10:	Cach di thu 16:	Cach di thu 22:	Cach di thu 28:
1 2 2 2 1 1 2 2	2 1 2 2 1 1 2 2	2 2 1 2 2 1 1 2	2 2 2 1 2 2 2 1	2 2 2 2 2 2 1 1
Cach di thu 5:	Cach di thu 11:	Cach di thu 17:	Cach di thu 23:	Cach di thu 29:
1 2 2 2 2 1 1 2	2 1 2 2 2 1 1 2	2 2 1 2 2 2 1 2	2 2 2 2 1 1 2 2	2 2 2 2 2 2 2
Cach di thu 6:	Cach di thu 12:	Cach di thu 18:	Cach di thu 24:	
1 2 2 2 2 2 1 2	2 1 2 2 2 2 1 2	2 2 1 2 2 2 2 1	2 2 2 2 1 2 1 2	

11

Phương pháp thiết kế vét cạn: Robot ll=34

.....

Cach di thu 150:

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 1

Cach di thu 151:

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 2

Cach di thu 152:

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 1

Cach di thu 153:

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1

Cach di thu 154:

2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2

12

Phương pháp thiết kế vét cạn: Robot ll=34

Nếu ll= 44, ll= 54 ---→ rất chậm

13

Phương pháp thiết kế vét cạn Sử dụng biến trạng thái

```
void phantich_yc(int ll, int tt) // bien tt luu so buoc 1 met da di
  {
    if(ll==0)      in();
    else
    {
      if(tt<2)   // neu chua di qua 2 buoc 1 m thi co the chon buoc 1 m hoac 2 m
      {
        for(int i=1; i<=2; i++)  // co hai buoc co do dai la 1 va 2
          if(i<= ll)
          {
            sb++;   m[sb]=i; // them 1 buoc di co do dai i
            if(i==1) phantich_yc(ll-i,tt+1);  //neu di buoc 1 m thi tang bien tt len 1
            else phantich_yc(ll-i,tt);
            sb--;          // lui lai 1 buoc de di buoc co do dai khac
          }
      }
      else   // da di xong 2 buoc 1 m thi chi co the di buoc 2m
        for(int i=2; i<=2; i++)       if(i<= ll)      {  sb++;   m[sb]=i;  phantich_yc(ll-i,tt);   sb--;      }
    }
  }
```

14

Phương pháp quay lui (Back Tracking)

Backtracking is an algorithmic-technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time (by time, here, is referred to the time elapsed till reaching any level of the search tree).

There are three types of problems in backtracking

- Decision Problem – In this, we search for a feasible solution.
- Optimization Problem – In this, we search for the best solution.
- Enumeration Problem – In this, we find all feasible solutions.

15

Phương pháp quay lui (Back Tracking)

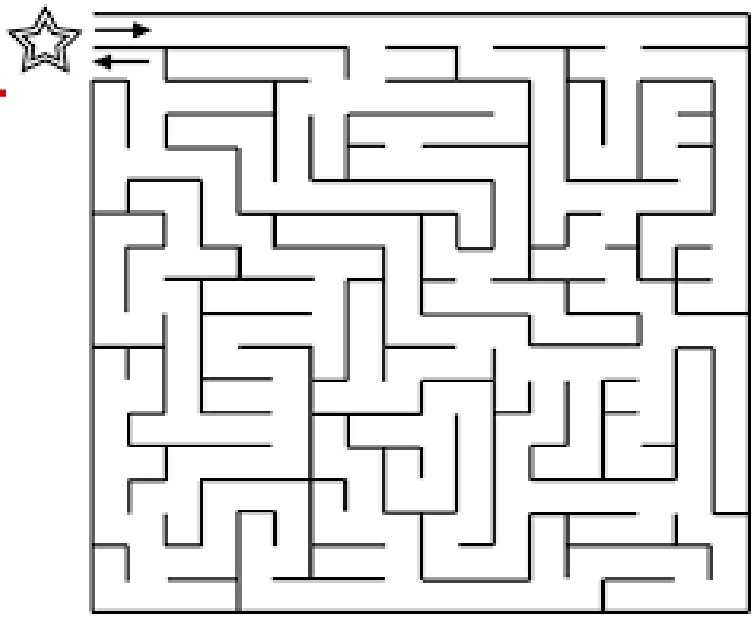
Tư tưởng của phương pháp là thử từng khả năng cho đến khi tìm thấy lời giải đúng. Đó là một quá trình tìm kiếm theo độ sâu trong một tập hợp các lời giải.

Trong quá trình tìm kiếm, nếu ta gặp một hướng lựa chọn không thỏa mãn, ta quay lui về điểm lựa chọn nơi có các hướng khác và thử hướng lựa chọn tiếp theo. Khi đã thử hết các lựa chọn xuất phát từ điểm lựa chọn đó, ta quay lại điểm lựa chọn trước đó và thử hướng lựa chọn tiếp theo tại đó. Quá trình tìm kiếm thất bại khi không còn điểm lựa chọn nào nữa.

16

Edge A

17



18

Sơ đồ chung của thuật toán quay lui

Lời giải của bài toán thường biểu diễn bằng một vectơ gồm n thành phần x = (x_1,...,x_n) phải thỏa mãn các điều kiện nào đó. Để chỉ ra lời giải x, ta phải xây dựng dần các lời giải x_i

- Tại mỗi bước i:

+ Đã xây dựng xong các thành phần x_1,.., x_{i-1}

+ Xây dựng thành phần x_i bằng cách lần lượt thử tất cả các khả năng mà x_i có thể chọn:

Nếu một khả năng j nào đó phù hợp cho x_i thì xác định x_i theo khả năng j.

Thường phải có thêm thao tác ghi nhận trạng thái mới của bài toán để hỗ trợ cho bước quay lui. Nếu i = n thì ta có được một lời giải, ngược lại thì tiến hành bước i+1 để xác định x_{i+1} .

Nếu không có một khả năng nào chấp nhận được cho x_i thì ta lùi lại bước trước (bước i-1) để xác định lại thành phần x_{i-1}.

19

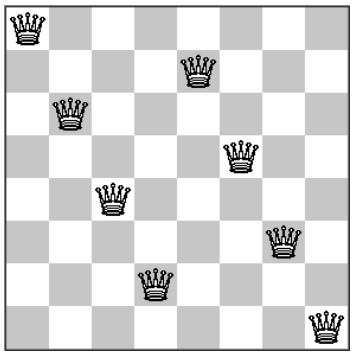
Backtracking algorithm scheme

```
ALGORITHM try(int i)
  IF (i>n) THEN RETURN (x1,...,xn)
  ELSE
     FOR each j DO
        IF (x1,...,xi-1, j) is acceptable
        {          Mark(j);
                   try(i+1);
                   UnMark(j);
        }
  RETURN;
```

20

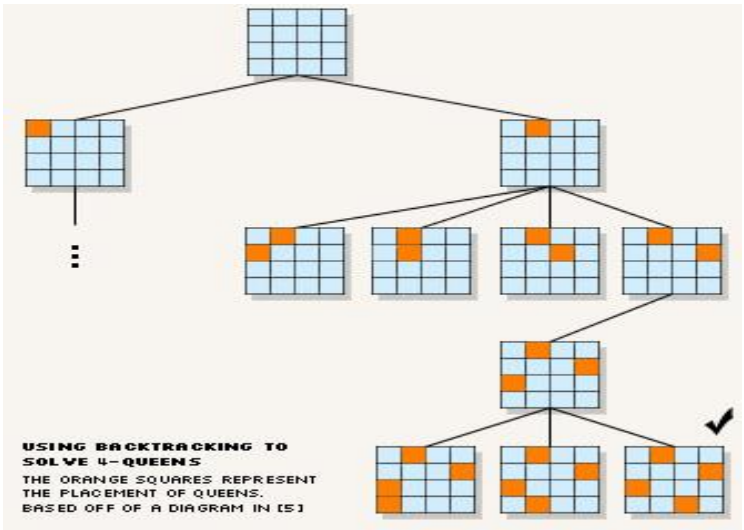
The Queens Problem (Bài toán 8 quân hậu)

- Consider a *n* by *n* chess board, and the problem of placing *n* queens on the board without the queens threatening one another.



21

Cây tìm kiếm cho bài toán 4 quân hậu

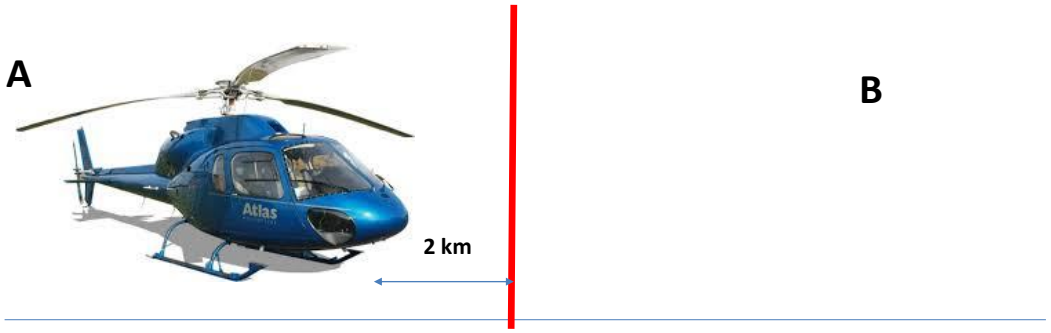


22

Phương pháp quay lui: Máy bay đi tuần

Cho n bình xăng chứa trong các bình v_1 đến v_n : { 5, 4, 2, 7, 6, 2, 3, 9}

Ban đầu trực thăng ở bên A và cách ranh giới d km. Hãy bố trí thứ tự các bình xăng để máy bay thực hiện n chuyến bay đúng theo lịch trực từ A sang B và ngược lại.



{ 5, 4, 2, 2, 3, 6, 7, 9}

23

Phương pháp quay lui: Máy bay đi tuần

Tổ chức chương trình:

+ Dùng các mảng V để lưu các bình xăng
+ Dùng mảng chọn để đánh dấu bình xăng đã được chọn hay chưa
+ Dùng mảng để lưu kết quả

Một bình xăng sẽ được chọn nếu như nó chưa được chọn trước đó và giá trị của nó phải lớn hơn d (để có thể bay chuyển từ miền này sang miền khác)

24

Phương pháp quay lui: Máy bay đi tuần

```
public class CSPE_MyP04_Quay_lui_MAY_BAY_TRUC_THANG {
   int [] kq,chon,v;
   int n, dem=0;
void khoitao()
   {
      n= 8;     v= new int [n];
      v[0]= 5; v[1]= 4; v[2]=2; v[3]= 7; v[4]= 6; v[5]= 2; v[6]=3; v[7]=9;
      chon= new int[n];        kq= new int[n];
      for(int j=0; j<n; j++) chon[j]= kq[j]=0;
   }
    void inkq()
   {
      System.out.println("\n Cach thu " + ++dem + ":");
      for(int j=0; j<n; j++) System.out.print(" " + kq[j]);
   }
```

25

Phương pháp quay lui: Máy bay đi tuần

```
public class CSPE_MyP04_Quay_lui_MAY_BAY_TRUC_THANG {
   void tim(int i, int d ) {}
   public static void main(String[] args) {
      CSPE_MyP04_Quay_lui_MAY_BAY_TRUC_THANG m= new CSPE_MyP04_Quay_lui_MAY_BAY_TRUC_THANG();
      int d=2; m.khoitao(); m.tim(1, d);
    }
}
```

26

Phương pháp quay lui: Máy bay đi tuần

```
void tim(int i, int d )
{
        if(i>n) inkq();   // nếu đã đủ hết n bình xăng thì in kq
        else
         for(int j=0; j<n; j++)    // duyệt hết các bình xăng
         {
                 if(chon[j]==0  && v[j] >d)  // nếu bình xăng chưa được chọn và có thể bay hơn khoảng cách d
                 {
                     kq[i-1]= v[j];  chon[j]=1;        // lấy bình xăng và đánh dấu đã lấy
                     tim(i+1, v[j]- d);
                      kq[i-1]= 0;  chon[j]=0;
                 }
         }
}
```

27

Các lời giải cho bài toán Máy bay đi tuần

……

Cach thu 225:
3 2 2 6 9 7 4 5
Cach thu 226:
3 2 7 9 4 2 5 6
Cach thu 227:
3 2 7 9 6 4 2 5
Cach thu 228:
3 2 6 7 5 4 2 9
Cach thu 229:
3 2 6 9 5 2 4 7
Cach thu 230:
3 2 6 9 7 4 2 5

Nếu d= 7 thì :

Cach thu 98:
9 3 2 2 4 6 5 7
Cach thu 99:
9 3 2 2 4 6 7 5
Cach thu 100:
9 3 2 2 6 7 5 4
Cach thu 101:
9 3 2 2 6 7 4 5
Cach thu 102:
9 3 2 6 7 5 4 2

28

Tìm đáp án cho bài toán Sudoku

Given a partially filled 9×9 2D array ‘grid[9][9]’, the goal is to assign digits (from 1 to 9) to the empty cells so that every row, column, and subgrid of size 3×3 contains exactly one instance of the digits from 1 to 9.

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

Ta sẽ thấy dùng máy tính có lợi như thế nào trong buổi học đến!!!

29

Bài tập về nhà (Phương pháp quay lui)

1. Cho n số nguyên khác nhau. Đếm có bao nhiêu cách chọn ra các số để có tổng bằng k.
2. Liệt kê các hoán vị của các ký tự trong chuỗi S.
3. Phân tích số n thành tổng 3 số. Đếm có bao nhiêu cách
4. Liệt kê các dãy nhị phân độ dài n, sao cho không có 2 bit 1 kề nhau.

30

Bài tập về nhà (Phương pháp quay lui)

1. Cho n số nguyên khác nhau. Đếm có bao nhiêu cách chọn ra các số để có tổng bằng k.

Ví dụ: a[0]= 1, a[1]= 2, a[2]= 4, a[3]=7, a[4]= 8

k= 10

1 2 7

2 8

k= 100: không có cách nào

31

Tài liệu đọc thêm

https://en.wikipedia.org/wiki/Backtracking

https://www.drdobbs.com/cpp/solving-combinatorial-problems-with-stl/184401194

32

Link YouTube

https://www.youtube.com/watch?v=DKCbsiDBN6c

https://www.youtube.com/watch?v=HzeK7g8cD0Y&list=PLqM7alHXFySESatj68JKWHRVhoJ1BxtLW

https://www.youtube.com/watch?v=-QcPo_DWJk4

33