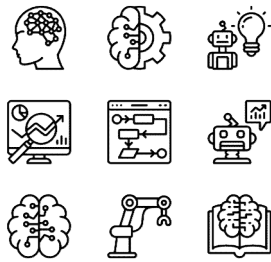


Computer Science for Practicing Engineers

Các phương pháp thiết kế thuật toán



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

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 Quy hoạch động (Dynamic Programming)
5. Phương pháp tham lam (Greedy Algorithms)
6. Phương pháp ngẫu nhiên (Randomized Algorithm)

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.

The brute-force strategy is indeed the one that is easiest to apply.

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.



Phương pháp thiết kế vét cạn

Ví dụ 1: 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*

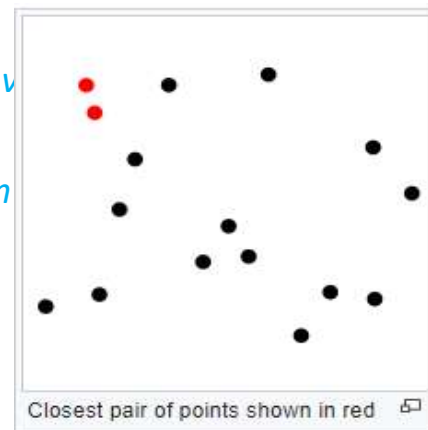


Phương pháp thiết kế vét cạn

Ví dụ 2: Tìm cặp điểm có khoảng cách nhỏ nhất trong n điểm (trên mặt phẳng Oxy)

- Liệt kê tất cả các cặp điểm *// tất cả các ứng viên*
- Tính khoảng cách từng cặp *// tính toán*
- So sánh và tìm ra kết quả *// kiểm tra để tìm lời giải*

Độ phức tạp của thuật toán là $O(n^2)$



Phương pháp thiết kế vết cạp: robot

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 l m. Hỏi có bao nhiêu cách robot đi vừa hết đoạn đường AB nhưng thực hiện không quá ($k=2$) bước đi 1 m.



Phương pháp thiết kế vết cạp: robot

Giải pháp theo pp vết cạp:

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

$l=4$ m

1 1 1 1

1 1 2

1 2 1

2 1 1

2 2



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

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
```

```
        }
```

```
    }
```

Phương pháp thiết kế vết cặn: 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
            }
}
```

```
#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);
}
```

```
C:\Users\Administrator\Documents\CSPE_2020\Phan_tich_buoc_di_robot.exe
Cach di thu 1:1 1 2
Cach di thu 2:1 2 1
Cach di thu 3:2 1 1
Cach di thu 4:2 2
-----
Process exited after 0.01634 seconds with return value 0
Press any key to continue . . .
```

Phương pháp thiết kế vết cạp: Robot II=14

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

Phương pháp thiết kế vết cạp: Robot II=34

.....

Cách di thu 150:

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

Cách di thu 151:

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

Cách di thu 152:

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

Cách di thu 153:

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

Cách di thu 154:

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

Phương pháp thiết kế vết cạp: Robot $l=34$

Nếu $l=44$, $l=54$ ---→ rất chậm!!!

Các bạn xem lại chương trình và cho biết hạn chế của thuật toán là gì???

Phương pháp thiết kế vết cạp: Robot $l=34$

Nếu $l=44$, $l=54$ ---→ rất chậm!!!

Cải tiến thuật toán

- Ta sử dụng một biến trạng thái để lưu các bước có độ dài 1 đã đi.
- Trường hợp nếu như chưa đi quá 2 (trường hợp tổng quát là k) bước có độ dài 1, ta có thể chọn bước có độ dài 1 để đi ở lần sau.
- Trường hợp đã đi 2 bước độ dài 1 thì lần sau ta chỉ chọn được độ dài 2.

Phương pháp thiết kế vết cạn sử dụng biến trạng thái

```
void phantich_yc(int ll, int tt) {
    if(ll==0)    in();
    else {
        if(tt<2) {
            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); 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--; }
    }
}
```

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).

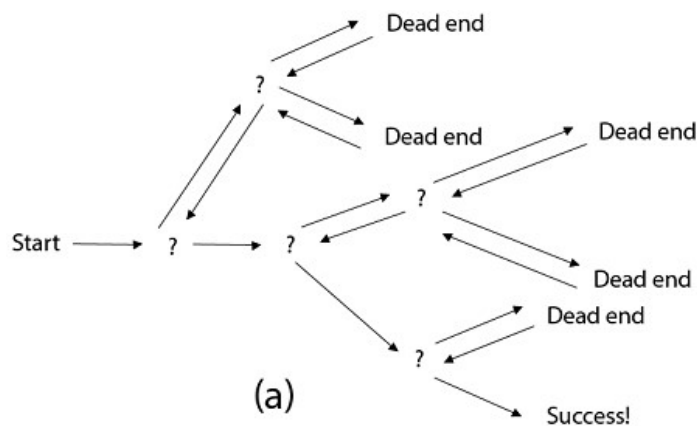
Phương pháp quay lui (Back Tracking)

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.

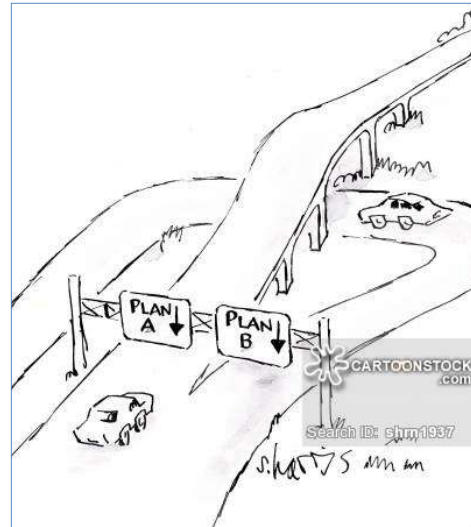
Phương pháp quay lui (Back Tracking)

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.



Phương pháp quay lui (Back Tracking)

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.

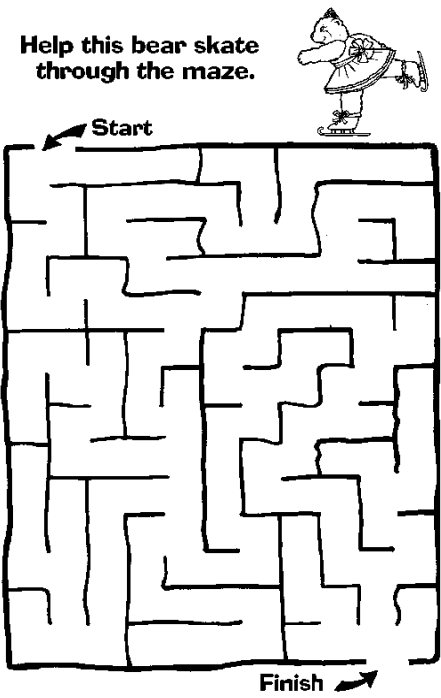


Help this bear skate through the maze.

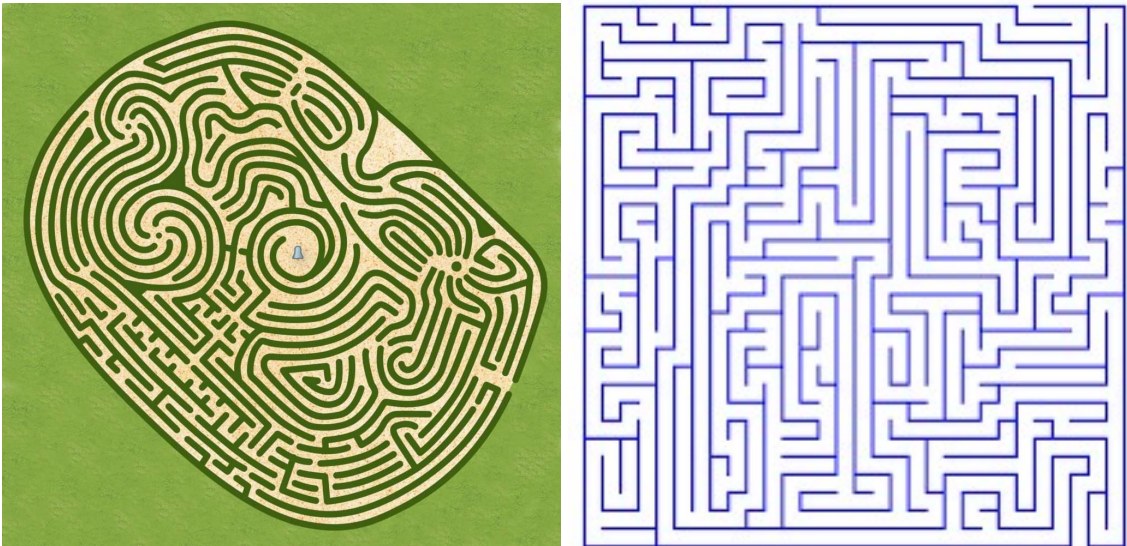
Start

Finish

Bạn thử tìm đường???
Nhận xét gì về cách tìm đường của mình???



Tìm đường trong mê cung



Sơ đồ chung của thuật toán quay lui-1

Lời giải của bài toán thường được biểu diễn bằng một vector gồm n thành phần $x = (x_1, \dots, x_n)$ 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, \dots, 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.

Sơ đồ chung của thuật toán quay lui-2

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} .

Backtracking algorithm scheme

ALGORITHM try(int i)

```

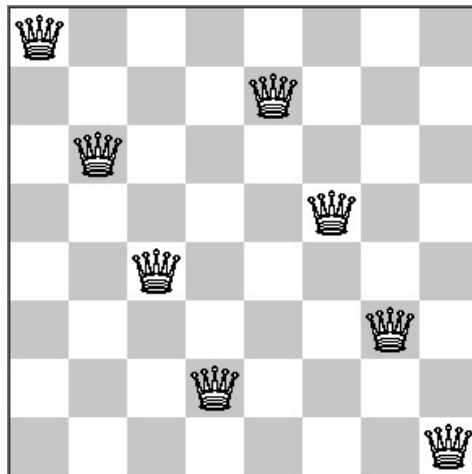
IF (i>n) THEN RETURN (x1,...,xn) // đã tìm xong lời giải thì in kết quả
ELSE
  FOR each j DO // thử lần lượt các khả năng j
    IF (x1,...,xi-1, j) is acceptable // nếu khả năng j chấp nhận đc cho giá trị thứ i
    {
      Mark(j); // chọn j cho x[i] và đánh dấu đã chọn (để không chọn lại)
      try(i+1); // tiếp tục thực hiện tìm cho x[i+1]
      UnMark(j); // gỡ lựa chọn khả năng j cho x[i] (để chọn khả năng khác)
    }
  }
RETURN;

```

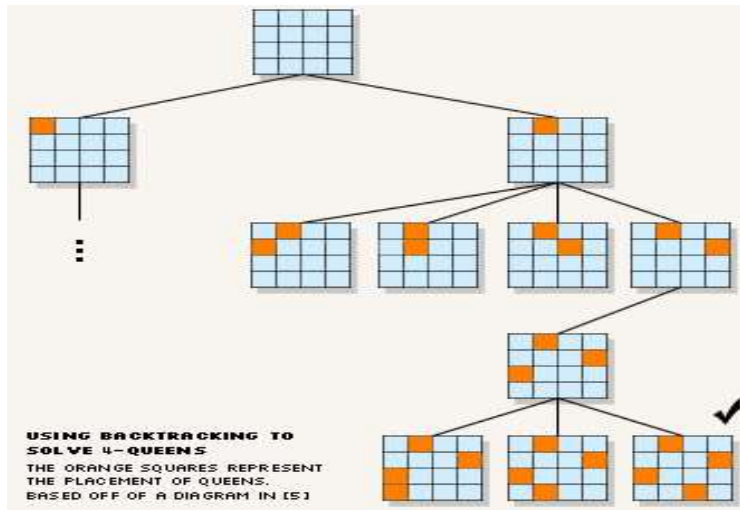
The Queens Problem (Bài toán n quân hậu)

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

Bạn phải quay lui lại bao nhiêu bước cho hình bàn cờ bên???



Cây tìm kiếm cho bài toán 4 quân hậu



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.



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 **kq** 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)

Phương pháp quay lui: Máy bay đi tuần

```
#include<bits/stdc++.h>
using namespace std;
int *kq,*chon, *v; // các mảng
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;
}
```

v_1 đến v_n : { 5, 4, 2, 7, 6, 2, 3, 9}

Phương pháp quay lui: Máy bay đi tuần

```
void inkq(){
    cout<<"\n Cach thu " << ++dem <<" :";
    for(int j=0; j<n; j++) cout<<" " <<kq[j];
}
void tim(int i, int d ) {
int main() {
    int d=2; khoitao(); tim(1, d);
}
```

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;
                }
        }
}
```

Các lời giải cho bài toán Máy bay đi tuần

.....

Cách thu 225:

3 2 2 6 9 7 4 5

Cách thu 226:

3 2 7 9 4 2 5 6

Cách thu 227:

3 2 7 9 6 4 2 5

Cách thu 228:

3 2 6 7 5 4 2 9

Cách thu 229:

3 2 6 9 5 2 4 7

Cách thu 230:

3 2 6 9 7 4 2 5

Nếu d= 7 thì :

Cách thu 98:

9 3 2 2 4 6 5 7

Cách thu 99:

9 3 2 2 4 6 7 5

Cách thu 100:

9 3 2 2 6 7 5 4

Cách thu 101:

9 3 2 2 6 7 4 5

Cách thu 102:

9 3 2 6 7 5 4 2

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		

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!!!

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 khác nhau. Đếm có bao nhiêu cách.
 - $N < 100000$
 - Thời gian chạy < 2 giây.

Tên file: *dem.cpp*; dữ liệu vào *dem.inp* chứa số nguyên n
 Dữ liệu ra: file *dem.out* chứa số bộ 3 số (khác nhau)
4. Liệt kê các dãy nhị phân độ dài n , sao cho không có 2 bit 1 kề nhau.

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

Tài liệu đọc thêm

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

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

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

