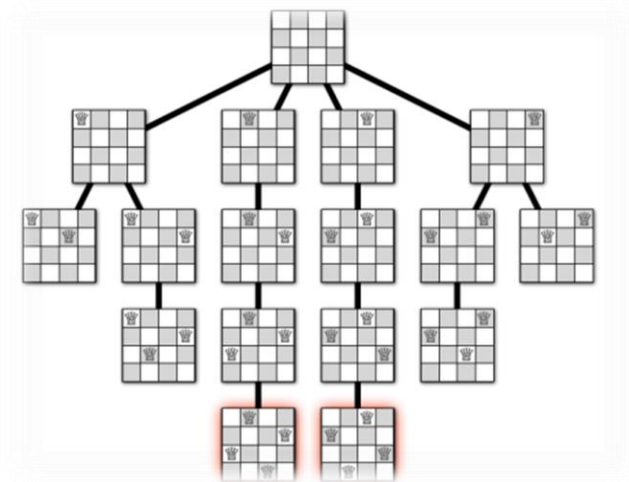


ĐẠI HỌC ĐÀ NẴNG
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN GIẢI THUẬT LẬP TRÌNH



Đề tài 7:

Tìm hiểu phương pháp **quay lui** + viết chương trình minh họa một số giải thuật **sắp xếp** trên mảng có kích thước n phần tử nguyên.

GVHD: *Trương Ngọc Châu*

SVTH: *Nguyễn Quốc Tấn*

Nhóm: *15.11D*

Lớp: *15T2*

Đà Nẵng, tháng 12/2017

This image shows a full page of primary-ruled paper. It features a solid vertical line on the left side, creating a margin. The rest of the page is filled with horizontal dotted lines, providing a guide for handwriting practice. There are no other markings or text on the page.

LỜI MỞ ĐẦU

Ngày nay, Tin học ngày càng phát triển nhanh chóng và được ứng dụng rộng rãi trong mọi lĩnh vực của đời sống xã hội, việc học và nắm bắt công nghệ mới đặc biệt là công nghệ thông tin ngày càng trở nên bức thiết. Đối với sinh viên trong ngành càng phải tích cực học tập, nắm vững mọi kiến thức về công nghệ thông tin, trong đó **giải thuật & lập trình** được xem là 1 trong những cơ sở, nền tảng không thể thiếu cho một lập trình viên.

Giải thuật & lập trình giúp cho sinh viên hiểu được tầm quan trọng của giải thuật và cách tổ chức cấu trúc dữ liệu, cũng như hình thành được phương pháp thuật toán rõ ràng để giải quyết những bài toán cụ thể.

Sau một thời gian nghiên cứu ngôn ngữ lập trình C (*C/C++ Programming Language*), môn Phân tích & thiết kế giải thuật, Cấu trúc dữ liệu, để nắm bắt những kiến thức đã học một cách tốt hơn, em đã thực hiện đề tài: **“Tìm hiểu phương pháp quay lui + viết chương trình minh họa một số giải thuật sắp xếp trên mảng có kích thước n phần tử nguyên.”**

Trong quá trình thực hiện đề tài, em không tránh khỏi gặp nhiều thiếu sót, nhưng đã nhận được sự chỉ dẫn, ý kiến đóng góp của quý thầy cô để hoàn thiện đề tài của mình. Đồng thời em xin gửi lời cảm ơn chân thành đến thầy **Trương Ngọc Châu** đã giúp em hoàn thành đề tài này.

Sinh viên thực hiện

Nguyễn Quốc Tấn

MỤC LỤC

LỜI MỞ ĐẦU.....	2
MỤC LỤC.....	3
1. GIỚI THIỆU ĐỀ TÀI.....	4
2. CƠ SỞ LÝ THUYẾT.....	4
2.1. Ý tưởng.....	4
2.2. Cơ sở lý thuyết.....	4
3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN.....	4
3.1. Phát biểu bài toán.....	4
3.2. Cấu trúc dữ liệu.....	4
3.3. Thuật toán.....	5
4. CHƯƠNG TRÌNH VÀ KẾT QUẢ.....	39
4.1. Tổ chức chương trình.....	39
4.2. Kết quả.....	39
4.2.1. Giao diện chính của chương trình	39
4.2.2. Kết quả thực thi của chương trình.....	40

1. GIỚI THIỆU ĐỀ TÀI

Đề tài 7: Tìm hiểu phương pháp **quay lui** +viết chương trình minh hoạ một số giải thuật **sắp xếp** trên mảng có kích thước n phần tử nguyên.

Mô tả: Tìm hiểu phương pháp và ứng dụng vào các bài toán: n-hậu hòa bình, mã đi tuần, liệt kê các cấu hình tổ hợp, một số thuật toán sắp xếp, tìm kiếm.

Dữ liệu vào: Tập chứa dữ liệu nhập của bài toán tương ứng.

Phương pháp gợi ý: Phương pháp Backtracking, đệ quy,...

Dữ liệu ra: Tập chứa dữ liệu xuất của bài toán tương ứng.

2. CƠ SỞ LÝ THUYẾT

2.1. Ý tưởng

Quay lui là một kỹ thuật thiết kế giải thuật dựa trên đệ quy. Ý tưởng của quay lui là tìm lời giải từng bước, mỗi bước chọn một trong số các lựa chọn khả dĩ và đệ quy.

2.2. Cơ sở lý thuyết

Về bản chất, 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. Quy trình đó thường được cài đặt bằng một hàm đệ quy mà trong đó mỗi thể hiện của hàm lấy thêm một biến và lần lượt gán tất cả các giá trị có thể cho biến đó, với mỗi lần gán trị lại gọi chuỗi đệ quy tiếp theo để thử các biến tiếp theo. Chiến lược quay lui tương tự với tìm kiếm theo độ sâu nhưng sử dụng ít không gian bộ nhớ hơn, nó chỉ lưu giữ trạng thái của một lời giải hiện tại và cập nhật nó.

3. TỔ CHỨC CẤU TRÚC DỮ LIỆU VÀ THUẬT TOÁN

3.1. Phát biểu bài toán

Input:

- Tập chứa dữ liệu nhập của bài toán tương ứng.
- Nhập từ bàn phím

Output:

- Xuất ra màn hình
- Tập chứa dữ liệu ra của bài toán tương ứng

3.2. Cấu trúc dữ liệu

Giải thuật quay lui có thể được cài đặt bằng đệ quy hoặc vòng lặp.

Tìm kiếm và sắp xếp được cài đặt bằng vòng lặp.

3.3. Thuật toán

A. Thuật toán quay lui

1. Xếp n con hậu

Tìm tất cả các khả năng xếp n con hậu trên một bàn cờ có $n \times n$ ô sao cho các con hậu không tấn công nhau

Độ phức tạp nhỏ nhất $O(n!)$

Hàm `trienVong()` xác định điều kiện tiếp tục được trình bày như sau:

```
int trienVong(int k)
{
    int i;
    for (i=1; i<k; i++)
        if (s[k]==s[i] || abs(i-k)==abs(s[i]-s[k])) return 0;
    return 1;
}
```

Điều kiện tiếp tục là mỗi hàng, mỗi cột và mỗi đường chéo chỉ chứa 1 con hậu.

Hàm `xepHau()` in vị trí con Hậu ra màn hình

```
int xepHau(int k)
{
    int i;
    if (k==n+1)
    {
        d1++;
        printf("\n%d: ",d1);
        for (i=1;i<=n;i++) printf("%d ", s[i]);
        printf("\n");
    }
    else for (i=1;i<=n;i++)
    {
        s[k]=i;
        if (trienVong(k)) xepHau(k+1);
    }
}
```

Hàm `xepHau1()` in vị trí con Hậu ra file output.txt

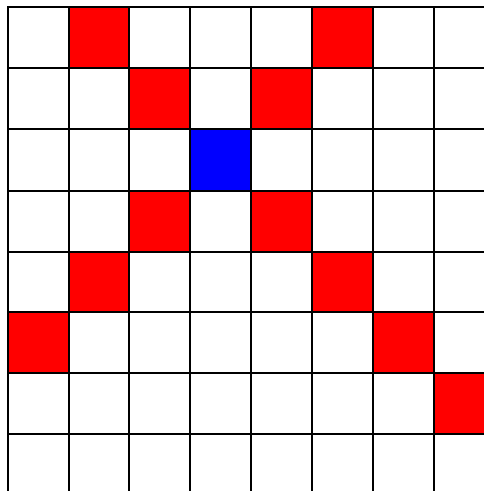
```
int xepHau1(int k)
```

```

{
    int i;
    if (k==n+1)
    {
        d11++;
        fprintf(f,"\\n%d: ",d11);
        for (i=1;i<=n;i++) fprintf(f,"%d ", s[i]);
        fprintf(f,"\\n");
    }
    else for (i=1;i<=n;i++)
    {
        s[k]=i;
        if (trienVong(k)) xepHau1(k+1);
    }
}

```

Bài toán 8 hậu hòa bình (Niklaus Wirth)



Các khai báo:

- Bàn cờ: `int S[8]`

ý nghĩa: $S[i] = j \leftrightarrow$ hàng i đặt hậu tại ô (i, j) .

- Các mảng cần đánh dấu:

`int a[8], b[15], c[15];`

$a[j] = \text{TRUE} \leftrightarrow$ cột j còn trống.

$c[i-j+7] = \text{TRUE} \leftrightarrow$ đường chéo song song chéo chính qua ô (i, j) còn trống.

$c[i+j] = \text{TRUE} \leftrightarrow$ đường chéo song song chéo phụ qua ô (i, j) còn trống.

Trước khi gọi `Try(0)` cần lấp đầy a, b, c giá trị 1 (TRUE).

Chương trình được cài đặt như sau.

```
#include <stdio.h>

int S[8], a[8], b[15], c[15], sol=0;

void Try(int i)
{
    int j;
    for (j=0; j<8; j++)
        if (a[j]&&b[i-j+7]&&c[i+j])
        {
            S[i]=j;
            a[j]=0;
            b[i-j+7]=0;
            c[i+j]=0;
            if (i==7)
            {
                printf("\n\n%3d: ",++sol);
                for (int i=0; i<8; i++)
                    printf("%d ", 1+S[i]);
            }
            else Try(i+1);
            a[j]=1;
            b[i-j+7]=1;
            c[i+j]=1;
        }
}

int main()
{
    int i;
    for (i=0; i<8; i++) a[i]=1;
    for (i=0; i<15; i++) b[i]=1;
    for (i=0; i<15; i++) c[i]=1;
    Try(0);
}
```

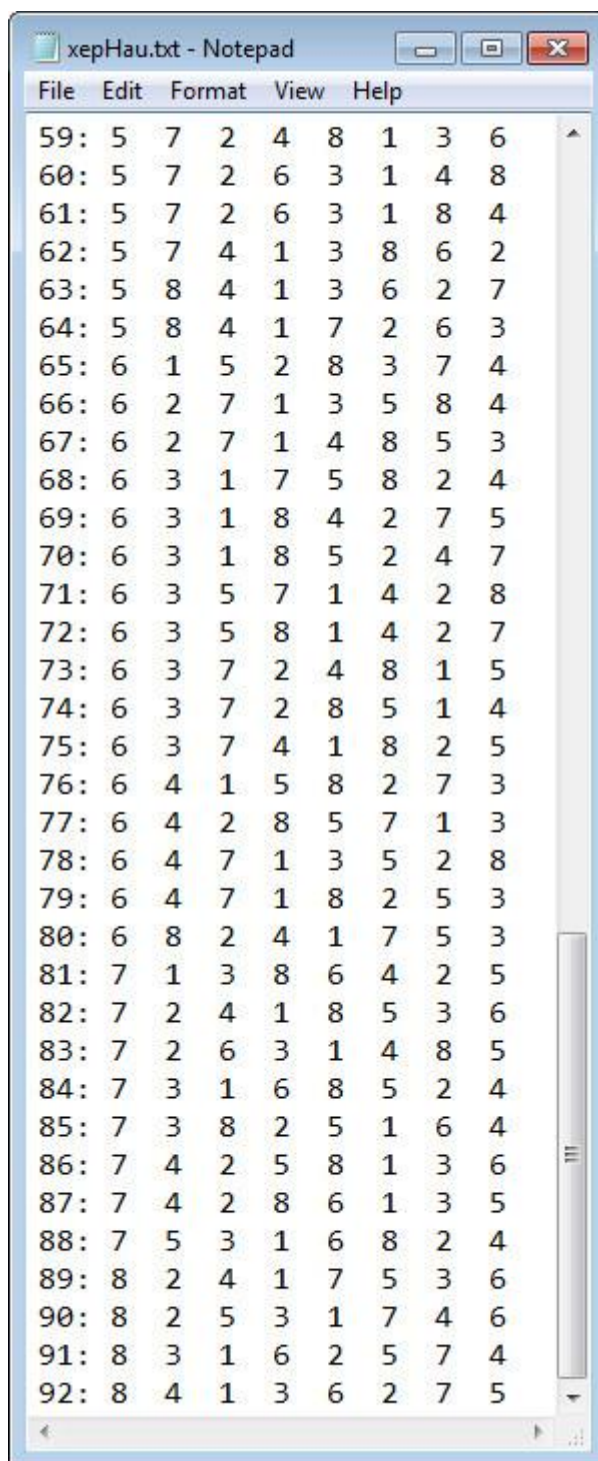

Kết quả chạy trên Dev C++ khi $n = 6$ như hình có 4 cách:

```

Chon Bai Toan: 11
11.Bai toan Xep Hau nhap tu ban phim bat ki la :
Nhap n >= 4: 6
1: 2  4  6  1  3  5
2: 3  6  2  5  1  4
3: 4  1  5  2  6  3
4: 5  3  1  6  4  2
  
```

Tệp tin xepHau.txt với $n = 8$ có 92 cách kể cả các vị trí đối xứng như hình :

Thực ra bài toán chỉ có 12 lời giải



2. Hoán vị

Hàm trienVong() xác định điều kiện tiếp tục được trình bày như sau:

```
int trienVong1(int k)
{
    int i;
    for (i=1; i<k; i++)
        if (s[k]==s[i]) return 0;
    return 1;
}
```

Hàm in hoán vị ra màn hình

```
int hoanVi(int k)
{
    int i;
    if (k==n+1)
    {
        d2++;
        printf("\n%d: ",d2);
        for (i=1;i<=n;i++) printf("%d ", s[i]);
        printf("\n");
    }
    else for(i=1;i<=n;i++)
    {
        s[k]=i;
        if (trienVong1(k)) hoanVi(k+1);
    }
}
```

Hàm in hoán vị ra file output.txt

```
int hoanVi1(int k)
{
    int i;
    if (k==n+1)
    {
        d22++;
    }
```

```

        fprintf(f, "\n%d: ", d22);
        for (i=1; i<=n; i++)
            fprintf(f, "%d ", s[i]);
        fprintf(f, "\n");
    }
    else for (i=1; i<=n; i++)
    {
        s[k]=i;
        if (trienVong1(k)) hoanVi1(k+1);
    }
}

```

Trong các hoán vị thì nhỏ nhất theo thứ tự tăng và lớn nhất theo thứ tự giảm

Ví dụ: $n=5$. Nhỏ nhất 1 2 3 4 5 và lớn nhất 5 4 3 2 1

Giả sử hiện thời $S = 32541$ thì phần tử tăng được là $s_i = 2$, vì 541 lớn nhất. Tăng s_i bằng cách đổi giá trị cho s_j nhỏ nhất lớn hơn ở bên phải là $s_j=4$. Sau khi hoán đổi có $S=34521$ thì 521 vẫn lớn nhất, để trở thành nhỏ nhất thì đảo thứ tự của chúng. Vậy hoán vị kế tiếp $S=32541$ là $S=34125$.

```

for (i=1; i<=n; i++) S[i]=i; print();
while (c<n!){
    i=n-1; while (S[i]>S[i+1]) i--;
    j=n; while (S[j]<S[i]) j--;
    tam=S[i]; S[i]=S[j]; S[j]=tam;
    j=i+1; k=n;
    while (j<k){ tam=S[j]; S[j]=S[k]; S[k]=tam; j++; k--;}
    print();
}

```

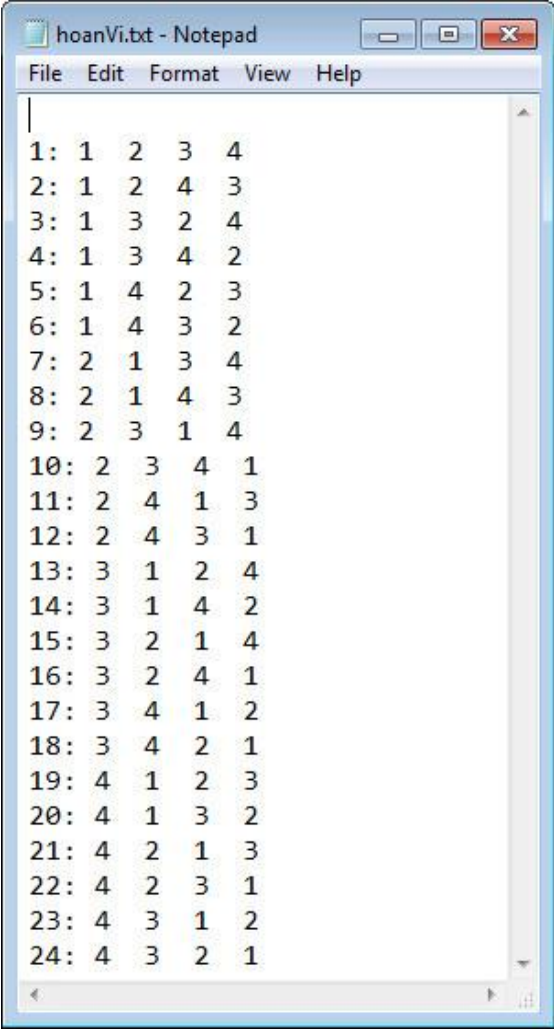
Với $n = 4$ ta có kết quả

12. Hoán vị của 1 số nhập từ bàn phím bất kỳ là :

Nhập n: 4

```
1: 1 2 3 4
2: 1 2 4 3
3: 1 3 2 4
4: 1 3 4 2
5: 1 4 2 3
6: 1 4 3 2
7: 2 1 3 4
8: 2 1 4 3
9: 2 3 1 4
10: 2 3 4 1
11: 2 4 1 3
12: 2 4 3 1
13: 3 1 2 4
14: 3 1 4 2
15: 3 2 1 4
16: 3 2 4 1
17: 3 4 1 2
18: 3 4 2 1
19: 4 1 2 3
20: 4 1 3 2
21: 4 2 1 3
22: 4 2 3 1
23: 4 3 1 2
24: 4 3 2 1
```

Tệp tin chứa kết quả chương trình là :



```
hoanVi.txt - Notepad
File Edit Format View Help

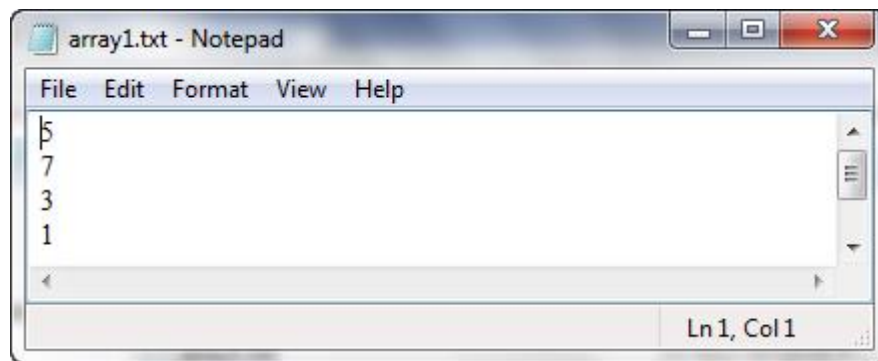
1: 1 2 3 4
2: 1 2 4 3
3: 1 3 2 4
4: 1 3 4 2
5: 1 4 2 3
6: 1 4 3 2
7: 2 1 3 4
8: 2 1 4 3
9: 2 3 1 4
10: 2 3 4 1
11: 2 4 1 3
12: 2 4 3 1
13: 3 1 2 4
14: 3 1 4 2
15: 3 2 1 4
16: 3 2 4 1
17: 3 4 1 2
18: 3 4 2 1
19: 4 1 2 3
20: 4 1 3 2
21: 4 2 1 3
22: 4 2 3 1
23: 4 3 1 2
24: 4 3 2 1
```

3. Tập con

```
void tapCon(int k)
{
    int i;
    if(k==n+1){
        d3++;
        printf("\n%d: ",d3);
        printf("{ ");
        for (i=1;i<=n;i++)
            if(s[i]==1)
                printf("%d ", a[i]);
        printf("}\n");
    }
    else{
        s[k]=0;tapCon(k+1);
        s[k]=1;tapCon(k+1);
    }
}
```

Độ phức tạp $O(2^n)$

Tệp tin chứa tập hợp cần nhập vào là



Kết quả bài toán với tập hợp gồm 4 phần tử 5,7,1,3 là

Chọn Bài Toán: 16

16. Tap con của mảng mô tu file array1.txt là :

1: { }

2: { 1 }

3: { 3 }

4: { 3 1 }

5: { 7 }

6: { 7 1 }

7: { 7 3 }

8: { 7 3 1 }

9: { 5 }

10: { 5 1 }

11: { 5 3 }

12: { 5 3 1 }

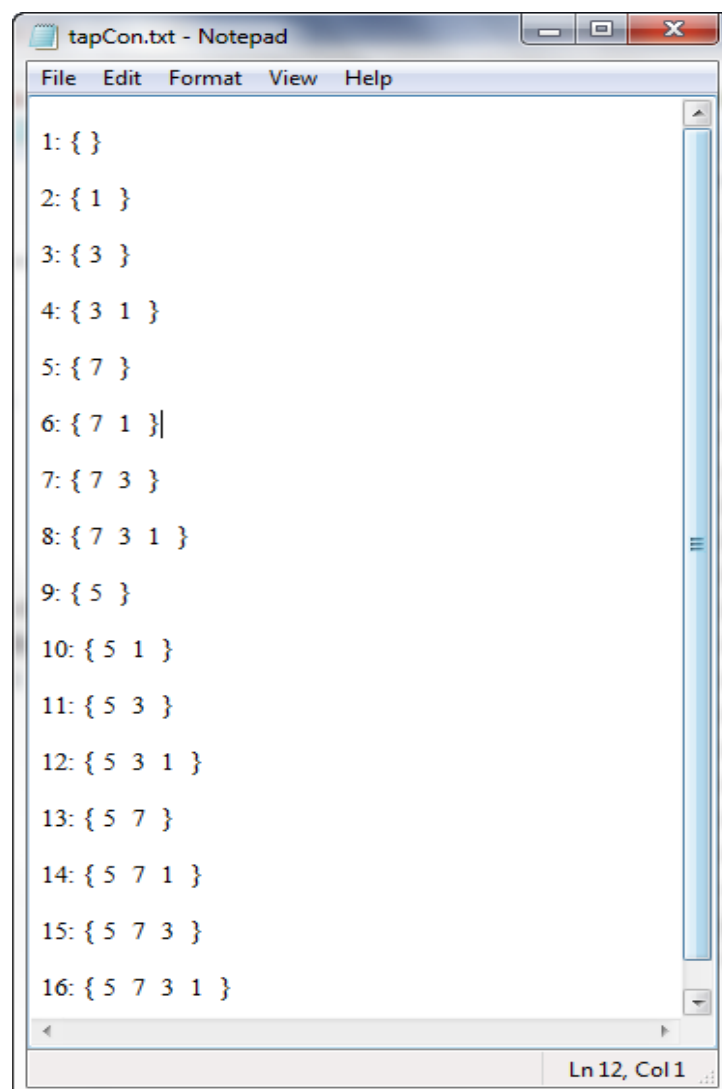
13: { 5 7 }

14: { 5 7 1 }

15: { 5 7 3 }

16: { 5 7 3 1 }

Tập tin chứa kết quả là



```
File Edit Format View Help
1: { }
2: { 1 }
3: { 3 }
4: { 3 1 }
5: { 7 }
6: { 7 1 }
7: { 7 3 }
8: { 7 3 1 }
9: { 5 }
10: { 5 1 }
11: { 5 3 }
12: { 5 3 1 }
13: { 5 7 }
14: { 5 7 1 }
15: { 5 7 3 }
16: { 5 7 3 1 }
Ln 12, Col 1
```

4. Tổ hợp

Nguyên lý chung

Để liệt kê tất cả các cấu hình $S=s_1s_2...s_k$ thuật toán giả sử đã có cấu hình con $s_1s_2...s_{i-1}$. Ở bước thử tìm giá trị cho s_i , $Try(i)$, duyệt qua mọi giá trị j đề cử được cho s_i và thực hiện 4 bước sau:

- $s_i=j$;
- <Thay đổi trạng thái>;
- Nếu đủ cấu hình ($i=k$) thì Print(S) ngược lại, gọi đệ quy để thử cho s_{i+1} , $Try(i+1)$;
- <Trả lại trạng thái cũ>;

Chương trình chính gọi $Try(1)$.

Lưu ý: Một số cấu hình không có hai bước b) và d)

Tổ hợp là bộ không thứ tự. Liệt kê theo thứ tự tăng ($s_i < s_{i+1}$). Giá trị j đề cử được cho s_i : $1+s_i \leq j \leq n-k+i$.

Chương trình chính gọi $Try(1)$ nên mảng S có $S[0]=0$.

```
void Try(int i)
{
    int j;
    for (j=1+S[i-1]; j<=n-k+i; j++)
    {
        S[i]=j;
        if (i==k)
        {
            printf("\n%3d:",++c);
            for (int i=1; i<=k; i++) printf("%d ", S[i]);
        }
        else Try(i+1);
    }
}
```

Phương pháp lặp

Liệt kê tổ hợp theo thứ tự tăng ($s_i < s_{i+1}$). Giá trị lớn nhất cho s_i là $n-k+i$.

```
for (i=1; i<=k; i++) S[i]=i; print();
while (c<C(n,k)) {
    i=k; while (S[i]==n-k+i) i--;
    S[i]++;
    for (j=i+1; j<=k; j++) S[j]=S[j-1]+1;
    print();
}
```

13. To hop cua 2 so nhap tu ban phim $n \geq k$ la :
 Nhap $n \geq k$: 7 3

```

1:1 2 3
2:1 2 4
3:1 2 5
4:1 2 6
5:1 2 7
6:1 3 4
7:1 3 5
8:1 3 6
9:1 3 7
10:1 4 5
11:1 4 6
12:1 4 7
13:1 5 6
14:1 5 7
15:1 6 7
16:2 3 4
17:2 3 5
18:2 3 6
19:2 3 7
20:2 4 5
21:2 4 6
22:2 4 7
23:2 5 6
24:2 5 7
25:2 6 7
26:3 4 5
27:3 4 6
28:3 4 7
29:3 5 6
30:3 5 7
31:3 6 7
32:4 5 6
33:4 5 7
34:4 6 7
35:5 6 7

```

5. Chinh hợp k lặp

Cần mảng a đánh dấu các giá trị j đã dùng, với ý nghĩa $a[j] = \text{TRUE} \leftrightarrow j$ chưa dùng.

```

void Try1(int i)
{
    int j;
    for (j=1; j<=n; j++) if (a[j])
    {
        s[i]=j;
        a[j]=0;
        if (i==k)
        {
            printf("\n%3d:", ++c1);
            for (int i=1; i<=k; i++) printf("%d ", s[i]);
        }
        else Try1(i+1);
    }
    a[j]=1;
}

```


}

Phương pháp lặp

Đưa thuật toán phát sinh hoán vị vào mỗi bước sinh tổ hợp có được thuật toán sinh chỉnh hợp không lặp.

Chương trình được cài đặt như sau.

```
#include <stdio.h>
#include <conio.h>

#define MAX 20
int S[MAX], A[MAX], c=0, cc=0, dem=0, n, k; //bo qua S[0].

void print();
int Com(int, int), fact(int k);

void main(){
    int i, j, ii, jj, kk, tam;

    printf("nhap n, k:"); scanf("%d%d%c", &n, &k);

    for (i=1; i<=k; i++) S[i]=i; c++;
    cc=0;
    for (ii=1; ii<=k; ii++) A[ii]=ii;
    cc++; print();
    while (cc<fact(k)){
        ii=k-1; while (A[ii]>A[ii+1]) ii--;
        jj=k; while (A[jj]<A[ii]) jj--;
        tam=A[ii]; A[ii]=A[jj]; A[jj]=tam;
        jj=ii+1; kk=k;
        while (jj<kk){ tam=A[jj]; A[jj]=A[kk]; A[kk]=tam; jj++; kk--
; }
        cc++; print();
    }
    while (c<Com(n,k)){
        i=k; while (S[i]==n-k+i) i--;
        S[i]++;
        for (j=i+1; j<=k; j++) S[j]=S[j-1]+1; c++;
        // Phát sinh k! hoán vị của tổ hợp hiện tại
        cc=0;
        for (ii=1; ii<=k; ii++) A[ii]=ii;
        cc++; print();
        while (cc<fact(k)){
            ii=k-1; while (A[ii]>A[ii+1]) ii--;
            jj=k; while (A[jj]<A[ii]) jj--;
            tam=A[ii]; A[ii]=A[jj]; A[jj]=tam;
            jj=ii+1; kk=k;
            while (jj<kk){ tam=A[jj]; A[jj]=A[kk]; A[kk]=tam; jj++; kk--
; }
            cc++; print();
        }
    }
    getch();
}
```

```

}

void print()
{
    int i;
    printf("\n%3d:", ++dem);
    for (i=1; i<=k; i++) printf("%d ", S[A[i]]);
}

int Com(int n, int k)
{
    if (k==0 || k==n) return 1; else return Com(n-1, k-1) + Com(n-1, k);
}

int fact(int k)
{
    if (k==0) return 1; else return k*fact(k-1);
}

```

Cũng có thể, không dùng bài toán đếm, kết hợp vừa kiểm tra dừng vừa tìm s_i bên phải nhất có thể tăng được. Chẳng hạn,

- với chỉnh hợp không lặp: $i=k$; while ($i>0$ && $S[i]==n-k+i$) $i--$;
if ($i==0$) dừng; else $S[i]$ tăng được...
- với hoán vị: $i=n-1$; while ($i>0$ && $S[i]>S[i+1]$) $i--$;
if ($i==0$) dừng; else $S[i]$ tăng được...

Chọn Bài Toán: 14

14. Chỉnh hợp của 2 số nhập từ bàn phím $n \geq k$ là :
Nhập $n \geq k$: 4 2

```

1:1 2
2:1 3
3:1 4
4:2 1
5:2 3
6:2 4
7:3 1
8:3 2
9:3 4
10:4 1
11:4 2
12:4 3

```

6. Chỉnh hợp lặp

```

void Try2(int i)
{
    int j;
    for (j=1; j<=n; j++)
    {
        s[i]=j;
        if (i==k)

```

```

    {
        printf("\n%3d:", ++c2);
        for (int i=1; i<=k; i++) printf("%d ", s[i]);
    }
    else Try2(i+1);
}
}

```

Phương pháp lặp

```

for (i=1; i<=k; i++) S[i]=1; print();
while (c<nk){
    i=k; while (S[i]==n) i--;
    S[i]++;
    for (j=i+1; j<=k; j++) S[j]=1;
    print();
}

```

Chon Bai Toan: 15

15.Chinh hop lap cua 2 so nhap tu ban phim bat ki la :
Nhap n, k bat ki : 6 2

```

1:1 1
2:1 2
3:1 3
4:1 4
5:1 5
6:1 6
7:2 1
8:2 2
9:2 3
10:2 4
11:2 5
12:2 6
13:3 1
14:3 2
15:3 3
16:3 4
17:3 5
18:3 6
19:4 1
20:4 2
21:4 3
22:4 4
23:4 5
24:4 6
25:5 1
26:5 2
27:5 3
28:5 4
29:5 5
30:5 6
31:6 1
32:6 2
33:6 3
34:6 4
35:6 5
36:6 6

```

7. Mã đi tuần

Có rất nhiều lời giải cho bài toán này, chính xác là 26.534.728.821.064 lời giải trong đó quân mã có thể kết thúc tại chính ô mà nó khởi đầu.

Với $n = 5$, các tọa độ xuất phát sau không có lời giải : (2,3), (3,2), (1,2), (2,1),

Hàm kiểm tra vị trí con Mã có đúng hay k

```
int ok(int u, int k)
{
    if (u >= 1 && u <= k) return 1;
    else return 0;
}
```

Hàm nhập dữ liệu vị trí con Mã

```
void nhapdl()
{
    printf("\nNhap so hang M cua ban co : ");
    scanf("%d",&m1);
    printf("Nhap so cot N cua ban co : ");
    scanf("%d",&n1);
    do
    {
        printf("Nhap toa do hang ban dau X cua con ma (1<=X<=M) : ");
        scanf("%d",&x1);
    }
    while (!ok(x1, m1));
    do
    {
        printf("Nhap toa do cot ban dau Y cua con ma (1<=Y<=N) : ");
        scanf("%d",&y1);
    }
    while (!ok(y1, n1));
}
```

Khởi tạo vị trí =1, biến đếm sc=0

```
void khoitao()
{
    sc = 0;
    daqua[x1][y1] = 1;
}
```

Hàm in vị trí con Mã đi hết bàn cờ

```
void xuat(int c)
```

```
{  
    printf("-----\n");  
    printf("Cach thu %d de con ma di het ban co : \n",c);  
    for (int i = 1; i <= m1; i++)  
    {  
        for (int j = 1; j <= n1; j++)  
            printf("%d  ",daqua[i][j]);  
        printf("\n\n");  
    }  
}
```

Hàm in vị trí con Mã đi hết bàn cờ ra file MaDiTuan.txt

```
void xuat1(int c)  
{  
    fprintf(f,"-----\n");  
    fprintf(f,"Cach thu %d de con ma di het ban co : \n",c);  
    for (int i = 1; i <= m1; i++)  
    {  
        for (int j = 1; j <= n1; j++)  
            fprintf(f,"%d  ",daqua[i][j]);  
        fprintf(f,"\n\n");  
    }  
}
```

Hàm kiểm tra vị trí con Mã

```
void mxt(int u, int v, int c)  
{  
    int uu, vv;  
    if (c == m1 * n1)  
    {  
        sc++;  
        xuat(sc);  
        xuat1(sc);  
    }  
    else  
    {
```

```
for (int i = 0; i <= 7; i++)  
{  
    uu = u + h[i][0];  
    vv = v + h[i][1];  
    if (ok(uu, m1) && ok(vv, n1))  
        if (!daqua[uu][vv])  
        {  
            daquea[uu][vv] = c + 1;  
            mxt(uu, vv, c + 1);  
            daquea[uu][vv] = 0;  
        }  
    }  
}
```

Cach thu 302 de con ma di het ban co :

1 10 19 14 23

18 5 22 9 20

11 2 13 24 15

6 17 4 21 8

3 12 7 16 25

Cach thu 303 de con ma di het ban co :

1 10 15 18 23

16 5 22 9 14

11 2 17 24 19

6 21 4 13 8

3 12 7 20 25

Cach thu 304 de con ma di het ban co :

1 10 15 20 23

16 5 22 9 14

11 2 19 24 21

6 17 4 13 8

3 12 7 18 25

Vay co tong cong 304 cach de con ma di het ban co!(^_^)

```

MaDiTuan1.txt - Notepad
File Edit Format View Help
Cách thu 52237 de con ma đi hết bàn cờ :
1  32  51  46  3  48  19  22

52  45  2  49  20  23  4  17

33  50  31  54  47  18  21  10

44  53  56  61  24  11  16  5

57  34  43  30  55  26  9  12

40  37  60  25  62  15  6  27

35  58  39  42  29  8  13  64

38  41  36  59  14  63  28  7

-----
Cách thu 52238 de con ma đi hết bàn cờ :
1  32  55  46  3  48  19  22

54  45  2  49  20  23  4  17

33  50  31  56  47  18  21  10

44  53  58  51  24  11  16  5

59  34  43  30  57  26  9  12

40  37  52  25  62  15  6  27

35  60  39  42  29  8  13  64

38  41  36  61  14  63  28  7 |

Ln 940283, C

```

8. Sudoku

int Xuat(): Trong thuật toán ta sử dụng mảng một chiều gồm 81 phần tử (từ $a[0]$ đến $a[80]$). Hàm xuất giúp chúng ta in ra một ma trận vuông 9×9 thay vì một mảng 81 phần tử nằm trên một hàng ngang.

Int xuấtfile(): Ghi kết quả sau khi chạy chương trình vào file “output.txt”

int Try(): Điền số vào các ô trống trong bảng Sudoku, những ô nào đã có giá trị (khác 0) thì bỏ qua.

int isOK(): Kiểm tra các hàng, các cột và các vùng.

Với các biến $k, t, t1, t2; tmpX, tmpY$ ta tiến hành kiểm tra: (mỗi số chỉ xuất hiện 1 lần, từ 1 đến 9)

- Kiểm tra hàng thứ i đã đúng chưa?
 $t2 = i/9;$

1. for (k=(t2*9); k<(t2*9+9); k++)
2. if (a[k] == x)
 - return 0;
- Kiểm tra cột thứ j đã đúng chưa?
 - t1= i%9;
 - for (k=0; k<9; k++)
 - if (a[k*9+t1] == x)
 - return 0;
- Kiểm tra trong từng vùng đã đúng chưa?
 - tmpX = t2%3; tmpY = t1%3;
 - for (k=t2-tmpX; k<=t2-tmpX+2; k++)
 - for (t=t1-tmpY; t<=t1-tmpY+2; t++)
 - if (a[k*9+t] == x)
 - return 0;
- Cách để xác định vị trí của 1 ô a[i] (với a[i] = a[0..80]) thuộc khối thứ m là bao nhiêu trong ô Sudoku, thì ta có:
 - h = i/9; k = i%9;
 - h1 = h/3; k1 = k/3;
 - m = 3*h1 + k1;
 - Ví dụ: i = 15 thì ta có:
 - h = i/9 = 1; k = i%9 = 6;
 - h1 = h/3 = 0; k1 = k/3 = 2;
 - m = 3*h1 + k1 = 3*0 + 2 = 2;
 Vậy ô a[2] sẽ thuộc khối thứ 2;
 - Ví dụ: i = 78;
 - h = i/9 = 8; k = i%9 = 6;
 - h1 = h/3 = 2; k1 = k/3 = 2;
 - m = 3*h1 + k1 = 3*2 + 2 = 8;
 Vậy ô a[78] sẽ thuộc khối thứ 8; (là khối cuối cùng).

int findLastK(): Hàm dùng để kiểm tra lại trong bảng Sudoku có còn ô trống hay không. Sử dụng vòng lặp **for (i=81-1; i>=0;i--)**, nếu gặp ô trống thì quay lại xử lý tiếp.

int Nhap() //Lay du lieu tu file input.txt

```
{
    int i,cv1, tmp;
    FILE *fp = NULL;
    printf("Chon De bai 1->6 : ");
    scanf("%d",&cv1);
    switch(cv1)
    {
        case 1:
            fp = fopen("input1.txt", "r");
            break;
```

```
case 2:
    fp = fopen("input2.txt", "r");
    break;
case 3:
    fp = fopen("input3.txt", "r");
    break;
case 4:
    fp = fopen("input4.txt", "r");
    break;
case 5:
    fp = fopen("input5.txt", "r");
    break;
case 6:
    fp = fopen("input6.txt", "r");
    break;
}
fscanf(fp, "%d", &n);
for (i=0; i<n*n*n*n; i++)
{
    fscanf(fp, "%d", &tmp);
    a1[i]= tmp;
}
fclose(fp);
return 0;
}

int Xuat() //Xuat ra man hinh
{
    int i;
    printf("-----\n");
    for (i=0; i<n*n*n*n; i++)
    {
        if ( i%(n*n) ==0) printf("\n");
        printf(" %d", a1[i]);
    }
}
```

```
    }

    printf("\n-----");
    return 0;
}

int xuấtfile() //Xuất file output.txt
{
    int i;
    FILE *fp;
    fp=fopen("KetQuaSudoku.txt","w");
    for (i=0; i<n*n*n*n; i++)
    {
        if ( i == n*n*n*n) printf("\n-----");
        if(i%(n*n)==0)
        {
            printf("\n");
            fprintf(fp,"\n");
        }
        printf(" %d", a1[i]);
        fprintf(fp,"%d ", a1[i]);

    }
    fclose(fp);
}

int isOK(int i, int x) //kiem tra hang, cot, vung
{
    int k, t, t1, t2;
    int tmpX, tmpY;
    //kiem tra hang thu i da co cai nao trung chua
    t1= i%(n*n);
    t2= i/(n*n);
    for (k=(t2*n*n); k<(t2*n*n+n*n); k++)
        if (a1[k] == x)
            return 0;

    //kiem tra cot thu j da co cai nao trung chua
```

```
for (k=0; k<n*n; k++)
    if (a1[k*n*n+t1] == x)
        return 0;

//kiem tra trong o nxn
tmpX = t2%n; tmpY = t1%n;
for (k=t2-tmpX; k<=t2-tmpX+n-1; k++)
    for (t=t1-tmpY; t<=t1-tmpY+n-1; t++)
        if (a1[k*n*n+t] == x)
            return 0;
return 1;
}

int Try0(int i) //Thu dien so vao o trong
{
    int x1;
    while (a1[i]!=0)
        i++;
    for (x1=1; x1<=n*n; x1++)
    {
        if (isOK(i, x1))
        {
            a1[i] = x1;
            if (i==lastK){
                dem++;
                printf("\n-----\n");
                printf("Cach %d la \n",dem);
                xuatfile();
                printf("\n");
            }
            else
                Try0(i+1);
            a1[i] = 0;
        }
    }
}
```

```

    }

    return 0;
}

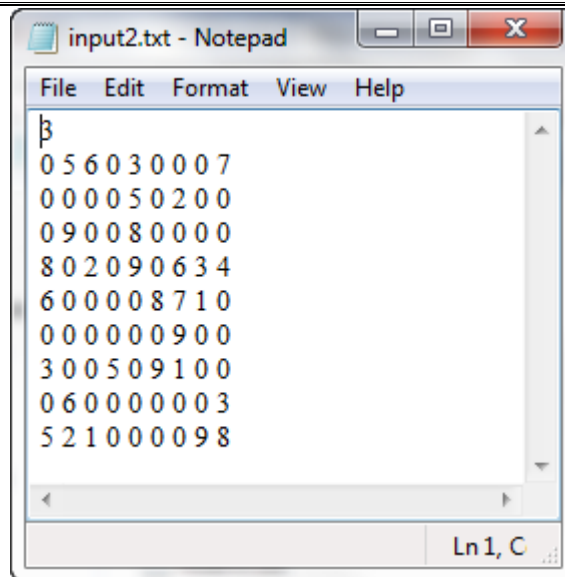
int findLastK() //Tim xem con o trong hay khong
{
    int i;
    for (i=n*n*n*n-1; i>=0;i--)
        if (a1[i]==0)
            return i;
}

```

Với hàm **int Try(int i)** ta có:

- Hàm **if(i==lastK)** sẽ thực hiện lệnh in ra, nên có độ phức tạp là $O(1)$;
- Hàm **if(isOK(i,x))**: ta có hàm **isOK** có 3 vòng lặp for, để kiểm tra hàng, cột và vùng nên có 3 độ phức tạp là **$O(fn)$** , **$O(gn)$** và **$O(hn)$** .
 - Ở vòng lặp kiểm tra hàng: **for (k=(t2*n*n); k<(t2*n*n+n*n); k++)** thì k chạy từ $(t2*n*n)$ đến $(t2*n*n+n*n)$ thực hiện $n*n$ lần, nên có độ phức tạp:
 $O(fn) = O(n^2)$;
 - Ở vòng lặp kiểm tra cột: **for (k=0; k<n*n; k++)** thì k chạy từ 0 đến $n*n$, thực hiện $n*n$ lần, nên có độ phức tạp:
 $O(gn) = O(n^2)$;
 - Ở vòng lặp kiểm tra vùng:
for (k=t2-tmpX; k<=t2-tmpX+n-1; k++)
for (t=t1-tmpY; t<=t1-tmpY+n-1; t++)
 thì k chạy từ $(t2-tmpX)$ đến $(t2-tmpX+n-1)$ và t chạy từ $(t1-tmpY)$ đến $(t1-tmpY+n-1)$ trong đó k và t thực hiện $n-1$ lần, nên có độ phức tạp:
 $O(hn) = O((n-1)*(n-1)) = O(n^2-2*n+1) = O(n^2)$;
- Vậy trong hàm **isOK** ta áp dụng quy tắc cộng cho 3 hàm thì có độ phức tạp là:
 $O(fn) + O(gn) + O(hn) = \max(O(fn), O(gn), O(hn)) = O(n^2)$;
- Vòng lặp **for (x=1; x<=n*n; x++)** có x chạy từ 1 đến $n*n$ nên có độ phức tạp là $O(n^2)$;

Vậy, hàm **int Try()** có độ phức tạp là: $O(1) * O(n^2) * O(n^2) = O(n^4)$;



Cách 7 là

```

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

```

Cách 8 là

```

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

```

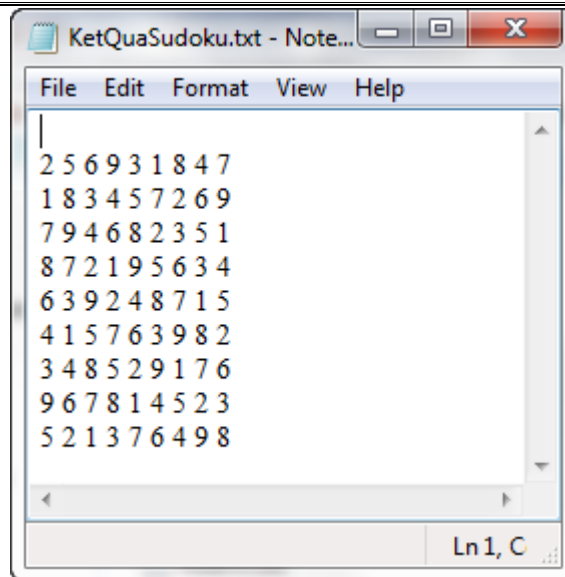
Cách 9 là

```

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

```

Vậy tổng cộng có 9 cách giải Sudoku này



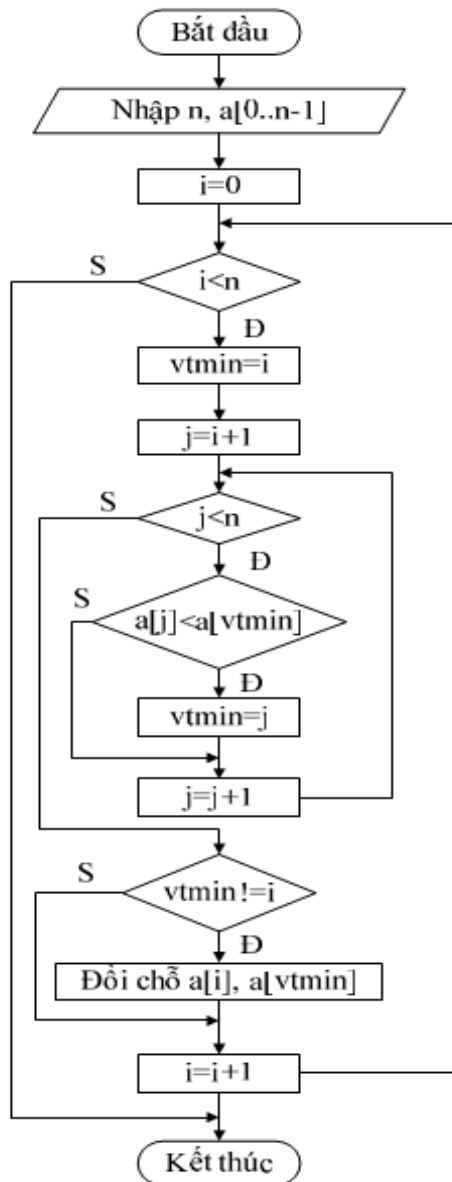
B. Thuật toán sắp xếp

1. Sắp xếp chọn

Mô tả thuật toán:

Tìm phần tử có khóa lớn nhất (nhỏ nhất), đặt nó vào đúng vị trí và sau đó sắp xếp phần còn lại của mảng.

Sơ đồ thuật toán:



Cài đặt bằng C của thuật toán:

```
void SelectSort(int *a,int n )
```

```
{
```

```
    int min;
```

//min là chỉ số phần tử nhỏ nhất

```
    for (i = 0; i < n - 1; i++)
```

```
    {
```

```
        int min = i;
```

```
        for (j = i + 1; j < n; j++)
```

```
            if (*(a+min)>*(a+j))
```

```
                min = j;
```

//tìm min trong phần tử còn lại

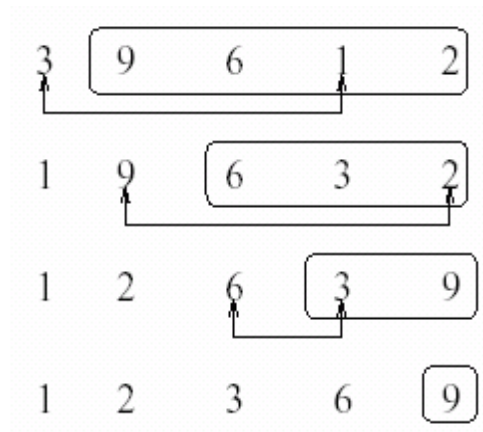
```
            swap(*(a+min),*(a+i));
```

//đổi chỗ nếu tìm thấy

```
        }
```

```
    }
```


Ví dụ:



Với mỗi giá trị của i thuật toán thực hiện $(n - i - 1)$ phép so sánh và vì i chạy từ 0 cho tới $(n-2)$, thuật toán sẽ cần $(n-1) + (n-2) + \dots + 1 = n(n-1)/2$ tức là $O(n^2)$ phép so sánh. Trong mọi trường hợp số lần so sánh của thuật toán là không đổi. Mỗi lần chạy của vòng lặp đối với biến i , có thể có nhiều nhất một lần đổi chỗ hai phần tử nên số lần đổi chỗ nhiều nhất của thuật toán là n . Như vậy trong trường hợp tốt nhất, thuật toán cần 0 lần đổi chỗ, trung bình cần $n/2$ lần đổi chỗ và tồi nhất cần n lần đổi chỗ.

2. Sắp xếp nổi bọt

Mô tả thuật toán:

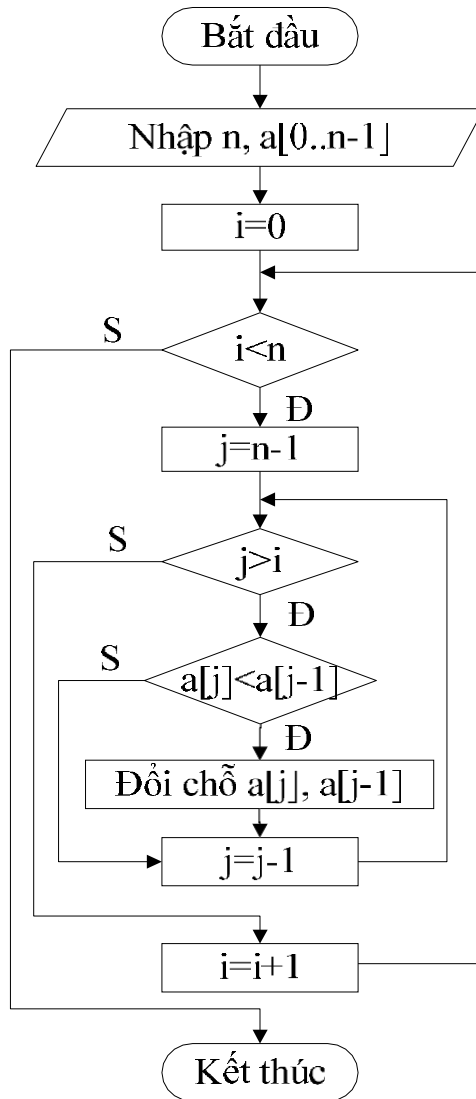
Thuật toán sắp xếp nổi bọt dựa trên việc so sánh và đổi chỗ hai phần tử ở kề nhau:

- i. Duyệt qua danh sách các bản ghi cần sắp theo thứ tự, đổi chỗ hai phần tử ở kề nhau nếu chúng không theo thứ tự.
- ii. Lặp lại điều này cho tới khi không có hai bản ghi nào sai thứ tự.

Không khó để thấy rằng n pha thực hiện là đủ cho việc thực hiện xong thuật toán.

Thuật toán này cũng tương tự như thuật toán sắp xếp chọn ngoại trừ việc có thêm nhiều thao tác đổi chỗ hai phần tử.

Sơ đồ thuật toán:

**Cài đặt thuật toán:**

```
void BubbleSort(int *a, int n)
```

```
{
    for (i=0; i<n; i++)                //sắp từ cuối lên đầu
    {
        for (j=n-1; j>i; j--)
            if(*(a+j)>*(a+j-1))
                swap(*(a+j),*(a+j-1));
    }
}
```

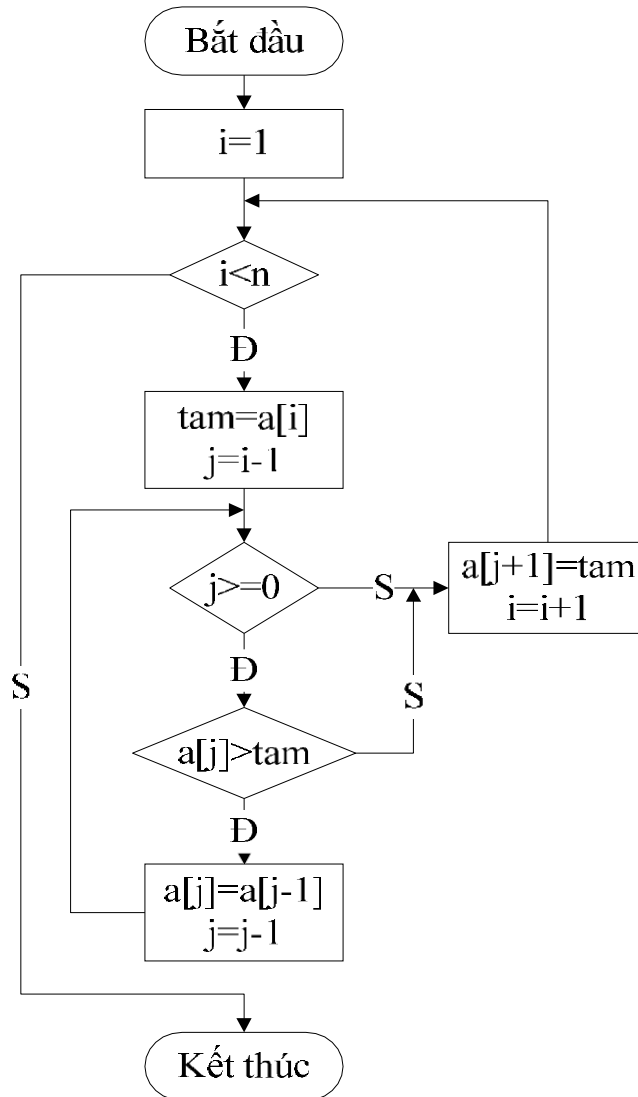
Thuật toán có độ phức tạp là $O(N*(N-1)/2) = O(N^2)$, bằng số lần so sánh và số lần đổi chỗ nhiều nhất của thuật toán (trong trường hợp tồi nhất). Thuật toán sắp xếp nổi bọt là thuật toán chậm nhất trong số các thuật toán sắp xếp cơ bản, nó còn chậm hơn thuật toán sắp xếp đổi chỗ trực tiếp mặc dù có số lần so sánh bằng nhau, nhưng do đổi chỗ hai phần tử kề nhau nên số lần đổi chỗ nhiều hơn.

3. Sắp xếp chèn

Mô tả thuật toán:

Thuật toán dựa vào thao tác chính là chèn mỗi khóa vào một dãy con đã được sắp xếp của dãy cần sắp. Phương pháp này thường được sử dụng trong việc sắp xếp các cây bài trong quá trình chơi bài.

Sơ đồ giải thuật của thuật toán như sau:



Có thể mô tả thuật toán bằng lời như sau: ban đầu ta coi như mảng $a[0..i-1]$ (gồm i phần tử, trong trường hợp đầu tiên $i = 1$) là đã được sắp, tại bước thứ i của thuật toán, ta sẽ tiến hành chèn $a[i]$ vào mảng $a[0..i-1]$ sao cho sau khi chèn, các phần tử vẫn tuân theo thứ tự tăng dần. Bước tiếp theo sẽ chèn $a[i+1]$ vào mảng $a[0..i]$ một cách tương tự. Thuật toán cứ thế tiến hành cho tới khi hết mảng (chèn $a[n-1]$ vào mảng $a[0..n-2]$). Để tiến hành chèn $a[i]$ vào mảng $a[0..i-1]$, ta dùng một biến tạm lưu $a[i]$, sau đó dùng một biến chỉ số $j = i-1$, dò từ vị trí j cho tới đầu mảng, nếu $a[j] > tam$ thì sẽ copy $a[j]$ vào $a[j+1]$, có nghĩa là lùi mảng lại một vị trí để chèn tam vào mảng. Vòng lặp sẽ kết thúc nếu $a[j] < tam$ hoặc $j = -1$, khi đó ta gán $a[j+1] = tam$.

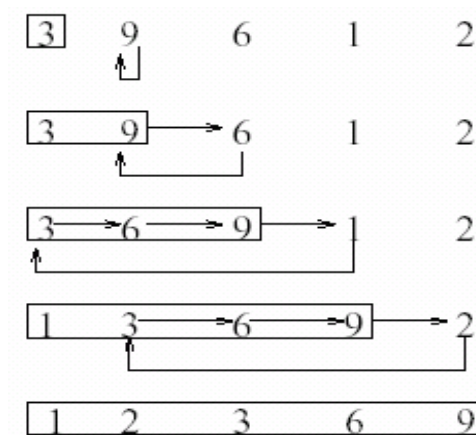
Cài đặt bằng C của thuật toán:

```
void InsertSort (int *a, int n)
{
```

```

int t;
for(i=1;i<n;i++)
{
    j=i-1;
    t=*(a+i);
    while(t<*(a+j)&& j>=0)
    {
        *(a+j+1)=*(a+j);
        j--;
    }
    *(a+j+1)=t;
}
}

```

Ví dụ:

Thuật toán sắp xếp chèn là một thuật toán sắp xếp ổn định (stable) và là thuật toán nhanh nhất trong số các thuật toán sắp xếp cơ bản.

Với mỗi i chúng ta cần thực hiện so sánh khóa hiện tại ($a[i]$) với nhiều nhất là i khóa và vì i chạy từ 1 tới $n-1$ nên chúng ta phải thực hiện nhiều nhất: $1 + 2 + \dots + n-1 = n(n-1)/2$ tức là $O(n^2)$ phép so sánh tương tự như thuật toán sắp xếp chọn. Tuy nhiên vòng lặp while không phải lúc nào cũng được thực hiện và nếu thực hiện thì cũng không nhất định là lặp i lần nên trên thực tế thuật toán sắp xếp chèn nhanh hơn so với thuật toán sắp xếp chọn. Trong trường hợp tốt nhất, thuật toán chỉ cần sử dụng đúng n lần so sánh và 0 lần đổi chỗ. Trên thực tế một mảng bất kỳ gồm nhiều mảng con đã được sắp nên thuật toán chèn hoạt động khá hiệu quả. Thuật toán sắp xếp chèn là thuật toán nhanh nhất trong các thuật toán sắp xếp cơ bản (đều có độ phức tạp $O(n^2)$).

- Trong trường hợp tốt nhất thuật toán sử dụng $n-1$ phép so sánh và 0 lần hoán vị.
- Trung bình thuật toán sử dụng $n^2/4$ phép so sánh và $n^2/4$ lần hoán vị.
- Trong trường hợp xấu nhất thuật toán sử dụng $n^2/2$ phép so sánh và $n^2/2$ lần hoán vị.
- Thuật toán thích hợp đối với mảng đã được sắp xếp một phần hoặc mảng có kích thước nhỏ.

4. Sắp xếp đổi chỗ trực tiếp

Tương tự như thuật toán sắp xếp bằng chọn và rất dễ cài đặt (thường bị nhầm với thuật toán sắp xếp chèn) là thuật toán sắp xếp bằng đổi chỗ trực tiếp (một số tài liệu còn gọi là thuật toán Interchange sort hay Straight Selection Sort).

Mô tả: Bắt đầu xét từ phần tử đầu tiên $a[i]$ với $i = 0$, ta xét tất cả các phần tử đứng sau $a[i]$, gọi là $a[j]$ với j chạy từ $i+1$ tới $n-1$ (vị trí cuối cùng). Với mỗi cặp $a[i]$, $a[j]$ đó, để ý là $a[j]$ là phần tử đứng sau $a[i]$, nếu $a[j] < a[i]$, tức là xảy ra sai khác về vị trí thì ta sẽ đổi chỗ $a[i]$, $a[j]$.

Ví dụ minh họa: Giả sử mảng ban đầu là $\text{int } a[] = \{2, 6, 1, 19, 3, 12\}$.

Các bước của thuật toán sẽ được thực hiện như sau:

$i=0, j=2$: 1, 6, 2, 19, 3, 12

$i=1, j=2$: 1, 2, 6, 19, 3, 12

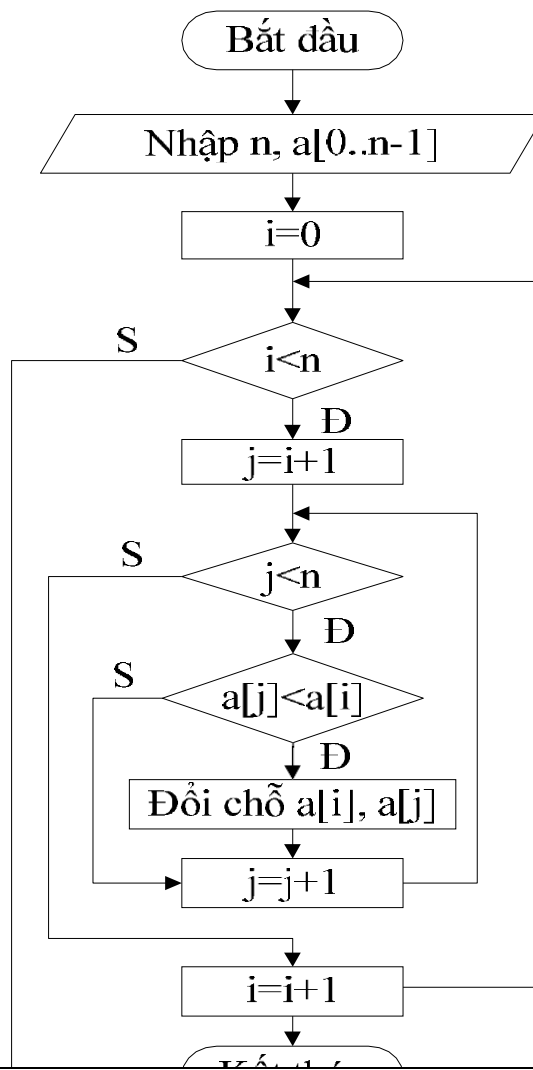
$i=2, j=4$: 1, 2, 3, 19, 6, 12

$i=3, j=4$: 1, 2, 3, 6, 19, 12

$i=4, j=5$: 1, 2, 3, 6, 12, 19

Kết quả cuối cùng: 1, 2, 3, 6, 12, 19.

Sơ đồ thuật toán:



Cài đặt của thuật toán:

```
void InterchangeSort(int *a,int n)
{
    for(i=0;i<n-1;i++)
        for(j=i+1;j<n;j++)
            if(*(a+j)<*(a+i))
                swap(*(a+i),*(a+j));
}
```

5. ShellSort**Ý tưởng thuật toán**

Là giải thuật cải tiến từ Insertion sort. Ý tưởng chính của thuật toán là phân chia dãy ban đầu thành những dãy con mà mỗi phần tử của dãy cách nhau 1 vị trí là h. Insertion sort áp dụng sau đó trên mỗi dãy con sẽ làm cho các phần tử được đưa về vị trí đúng tương đối (trong dãy con) 1 cách nhanh chóng.

Sau đó tiếp tục giảm khoảng cách h để tạo thành các dãy con mới (Tạo điều kiện để so sánh một phần tử với nhiều phần tử khác trước đó không ở cùng dãy con với nó) và lại tiếp tục sắp xếp.

Thuật toán dừng khi $h = 1$, lúc này bảo đảm tất cả các phần tử trong dãy ban đầu sẽ được so sánh với nhau để xác định trật tự cuối cùng.

```
void ShellSort(int *a,int n, int *h, int k)
{
    int step,i,j, x,len;
    for (step = 0 ; step <k; step++)
    {
        len = *(h+step);
        for (i = len; i<n; i++)
        {
            x = a[i];
            j = i-len;           //a[j] dùng ke truoac a[i] trong cung day con
            while ((x<*(a+j)&&(j>=0))) // sap xep day con chua x = pp chen truc tiep
            {
                *(a+j+len) = *(a+j);
                j = j - len;
            }
            *(a+j+len) = x;
        }
    }
}
```

}

}

6. Sắp xếp Nhanh

Quick sort là thuật toán sắp xếp được C. A. R. Hoare đưa ra năm 1962.

Quick sort là một thuật toán sắp xếp dạng chia để trị với các bước thực hiện như sau:

Selection: chọn một phần tử gọi là phần tử quay (pivot)

Partition (phân hoạch): đặt tất cả các phần tử của mảng nhỏ hơn phần tử quay sang bên trái phần tử quay và tất cả các phần tử lớn hơn phần tử quay sang bên phải phần tử quay. Phần tử quay trở thành phần tử có vị trí đúng trong mảng.

Đệ qui: gọi tới chính thủ tục sắp xếp nhanh đối với hai nửa mảng nằm 2 bên phần tử quay

```
int Partition(int *a,int l,int r)
```

```
{
```

```
    int p=*(a+l);
```

```
    int i=l+1;
```

```
    int j=r;
```

```
    while(1)
```

```
    {
```

```
        while(*(a+i)<=p&& i<r)
```

```
            ++i;
```

```
        while(*(a+j)>=p&& j>l)
```

```
            --j;
```

```
        if(i>=j)
```

```
        {
```

```
            swap(*(a+j),*(a+l));
```

```
            return j;
```

```
        }
```

```
        else swap(*(a+i),*(a+j));
```

```
    }
```

```
}
```

Sắp xếp nhanh

```
void QuickSort(int *a,int l,int r)
```

```
{
```

```
    if(r>l)
```

```
    {
```

```
        int p=Partition(a,l,r);
```

```

        QuickSort(a,l,p-1);

        QuickSort(a,p+1,r);

    }

}

```

Quick sort có độ phức tạp là $O(n \cdot \log(n))$, và trong hầu hết các trường hợp Quick sort là thuật toán sắp xếp nhanh nhất, ngoại trừ trường hợp tồi nhất, khi đó Quick sort còn chậm hơn so với Bubble sort.

4. CHƯƠNG TRÌNH VÀ KẾT QUẢ

4.1. Tổ chức chương trình

Viết trên Dev C++

Ngôn ngữ cài đặt C

4.2. Kết quả

4.2.1. Giao diện chính của chương trình

```

Nguyễn Quốc Tân 15T2 - nqt.it.bk@gmail.com

DO AN GIAI THUAT LAP TRINH C/C++ 2017
NGUYEN QUOC TAN 15T2 MSSV: 102150131

Mang mo tu file array.txt la:
50 25 49 24 48 23 47 22 46 21 45 20 44 19 43 18 42 17 41 16 40 15 39 14 38 13 37 12 36 11 35 10 34 9 33 8 32 7 5 31 3 30 1 29 2 28 4 27 6 26 16 26 29 53 67 100 83

1.SelectSort
2.InsertSort
3.BubbleSort
4.InterchangeSort
5.ShellSort
6.QuickSort
7.Bai toan Chuyen Dia Thap Ha Noi
8.Bai toan xep hau
9.Hoan vi
10.To hop
11.Chinh hop
12.Chinh hop lap
13.Tap con cua 1 taphop mo tu file
14.Bai toan Ma Di Tuan
15.Sudoku
0.Nhan 0 de thoat

Chon Bai Toan: 1

1.Mang da sap xep Chon Truc Tiep SelectSort la :
100 83 67 53 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Mang mo tu file output.txt la:
100 83 67 53 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Chon Bai Toan: 2

2.Mang da sap xep Chen InsertSort la :
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26 26 27 28 29 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 53 67 83 100

Mang mo tu file output.txt la:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26 26 27 28 29 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 53 67 83 100

Chon Bai Toan: 3

3.Mang da sap xep Noi Bot BubbleSort la :

```


4.2.2. Kết quả thực thi của chương trình

```

Nguyen Quoc Tan 15T2 - nqt.it.bk@gmail.com
100 83 67 53 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Chon Bai Toan: 4

4.Mang da sap xep Doi Cho Truc tiep InterchangeSort la :
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26 26 27 28 29 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 53 67 83 100

Mang mo tu file output.txt la:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26 26 27 28 29 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 53 67 83 100

Chon Bai Toan: 5

5.Mang da sap xep ShellSort la :
100 83 67 53 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Mang mo tu file output.txt la:
100 83 67 53 50 49 48 47 46 45 44 43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Chon Bai Toan: 6

6.Mang da sap xep Nhanh la :
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26 26 27 28 29 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 53 67 83 100

Mang mo tu file output.txt la:
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 16 17 18 19 20 21 22 23 24 25 26 26 27 28 29 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 53 67 83 100

Chon Bai Toan: 7

7.Bai toan ThapHaNoi nhap tu ban phim bat ki la :

Cho biet so dia can chuyen: 3

Cac buoc chuyen nhu sau:
1 : Chuyen dia 1 tu A sang C
2 : Chuyen dia 2 tu A sang B
3 : Chuyen dia 1 tu C sang B
4 : Chuyen dia 3 tu A sang C
5 : Chuyen dia 1 tu B sang A
6 : Chuyen dia 2 tu B sang C
7 : Chuyen dia 1 tu A sang C

Chon Bai Toan:

```

```

Nguyen Quoc Tan 15T2 - nqt.it.bk@gmail.com

Chon Bai Toan: 11

11.Bai toan Xep Hau nhap tu ban phim bat ki la :
Nhap n >= 4: 6

1: 2 4 6 1 3 5
2: 3 6 2 5 1 4
3: 4 1 5 2 6 3
4: 5 3 1 6 4 2

Chon Bai Toan: 12

12.Hoan vi cua 1 so nhap tu ban phim bat ki la :
Nhap n: 3

1: 1 2 3
2: 1 3 2
3: 2 1 3
4: 2 3 1
5: 3 1 2
6: 3 2 1

Chon Bai Toan: 13

13.To hop cua 2 so nhap tu ban phim n>=k la :
Nhap n>=k: 4 2

1:1 2
2:1 3
3:1 4
4:2 3
5:2 4
6:3 4

Chon Bai Toan:

```

Chon Bai Toan: 14

14.Chinh hop cua 2 so nhap tu ban phim $n \geq k$ la :

Nhap $n \geq k$: 4 2

1:1 2

2:1 3

3:1 4

4:2 1

5:2 3

6:2 4

7:3 1

8:3 2

9:3 4

10:4 1

11:4 2

12:4 3

Chon Bai Toan: 15

15.Chinh hop lap cua 2 so nhap tu ban phim bat ki la :

Nhap n, k bat ki : 4 2

1:1 1

2:1 2

3:1 3

4:1 4

5:2 1

6:2 2

7:2 3

8:2 4

9:3 1

10:3 2

11:3 3

12:3 4

13:4 1

14:4 2

15:4 3

16:4 4

Chon Bai Toan:

16. Tập con của mảng mô tu file array1.txt là :

1: { }

2: { 1 }

3: { 3 }

4: { 3 1 }

5: { 7 }

6: { 7 1 }

7: { 7 3 }

8: { 7 3 1 }

9: { 5 }

10: { 5 1 }

11: { 5 3 }

12: { 5 3 1 }

13: { 5 7 }

14: { 5 7 1 }

15: { 5 7 3 }

16: { 5 7 3 1 }

Chon Bai Toan:

Chon Bai Toan: 18

Chon De bai 1->6 : 3

De bai:

```

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

```

Giai

Cach 1 la

```

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

```

Vay tong cong co 1 cach giai Sudoku nay

TÀI LIỆU THAM KHẢO

1. Giáo trình Toán rời rạc của thầy Phan Thanh Tao
2. Giáo trình Phân tích và Thiết kế Giải Thuật Nguyễn Thanh Bình, Phạm Minh Tuấn, Đặng Thiên Bình
3. Google