خب رسیدم به این تسک نسبتا جالب و سخت برای خودم بریم قسمت به قسمت اجرا برنامه بررسی و آنالیز کنیم

۱- قسمت اول برنامه -> تعریف ماتریس و مشخص کردن تعداد ردیف و ستون و تکرار اجرا

```
int **mat,number,range;
double sum_dir=0,sum_rand=0;
cout<<"Enter a number  for row & column : ";cin>>number;
cout<<"Enter repetition number : ";cin>>range;
mat = new int*[number];
for(int i = 0; i < number; ++i)
    mat[i] = new int[number];
```

2-قسمت دوم برنامه -> حلقه تکراری که نوشتم خب مشخصه برای مرتبه تکرار هرکدوم از روش هاست
در داخل حلقه اول با تابع init ماتریس میسازیم سپس ماتریس ساخته شده و تعداد ردیف ستون پاس میدیم به توابع محاسبه کننده زمان های اجرا هر روش

```
for(int i=1;i<=range;i++){
    init(mat,number);
    sum_dir += calculator_dir(mat,number);
}
for(int i=1;i<=range;i++){
    init(mat,number);
    sum_rand += calculator_rand(mat,number);
}
```

3-قسمت سوم برنامه -> توابع محاسبه کننده

در مورد کلاس clock خودم از این سایت https://www.cplusplus.com/reference/ctime/clock/ یاد گرفتم دست پا شکسته برای انجام این تسک در حلقه while هم انجام میشه تا زمانی که درایه یک پیدا بکنه سپس در اخر زمان محاسبه شده یک بار عمل سرچ رندوم برمیگرده
در سرچ مستقیم هم همین کار انجام میشه منطقا الگوریتم خاص خودش داره !

```
double calculator_rand(int **mat,int number){

    int x,y,flag;

    clock_t t;
    t = clock();
    while(true){
    x = random(number);
    y = random(number);

    if(mat[x][y]==1){
            flag =1;
            break;
        }
    else continue;
}

    t = clock() - t;
    double time = ((double)t)/CLOCKS_PER_SEC;
    return time;
}
```

```
double calculator_dir(int **mat,int number){
    int flag=0;
    clock_t t;
    t = clock();

    for(int i=0;i<number;i++){
        for(int j=0;j<number;j++){
            if(mat[i][j]==1){
                flag=1;
                break;
            }
        }
    }
    t = clock() - t;
    double time = ((double)t)/CLOCKS_PER_SEC;

    return time;
}
```

;(

Clock program

.Returns the processor time consumed by the program

The value returned is expressed in *clock ticks*, which are units of time of a constant but .(system-specific length (with a relation of CLOCKS_PER_SEC *clock ticks* per second

The epoch used as reference by clock varies between systems, but it is related to the program execution (generally its launch). To calculate the actual processing time of a program, the value .returned by clock shall be compared to a value returned by a previous call to the same function