Análisis de complejidad temporal y espacial del algoritmo main:

`public static void main(String[] args) throws WrongEntriesException{`	
`System.out.println("Please enter the information for star the program");`	
`int cant=sn.nextInt();`	
`sn.nextLine();`	
`for(int i=0;i<cant;i++) {`	n
`String info=sn.nextLine();`	
`String[] parts=info.split(" ");`	
`if(parts.length!=4 || Integer.parseInt(parts[1])<=0 || Integer.parseInt(parts[2])<=0 || Integer.parseInt(parts[3])<=0) {`	
`throw new WrongEntriesException();`	
`}else {`	
`String idEdifice=parts[0];`	
`int numUser=Integer.parseInt(parts[1]);`	
`int numFloors=Integer.parseInt(parts[2]);`	
`int numOfficeForFloor=Integer.parseInt(parts[3]);`	
`int contador=0;`	
`String[] user=new String[numUser];`	
`while(numUser!=0) {`	n^2
`user[contador]=sn.nextLine();`	
`//System.out.println(user[contador]);`	
`numUser--;`	
`contador++;`	
`}`	
`createEdifice( idEdifice, numUser, numFloors, numOfficeForFloor,user);`	
`//System.out.println(idEdifice+numUser+numFloors+numOfficeForFloor);`	
`}`	
`}`	

Complejidad temporal:

$f(n) = n + n^2$

$O(f(n)) = n + n^2$

$O(n + n^2)$

Hacemos uso del peor caso (n de grado superior):

$O(n^2)$

Por lo tanto, la complejidad temporal del algoritmo main es $O(n^2)$.

Complejidad espacial:

Tipo	Variable	Tamaño de 1 valor atómico	Cantidad de valores atómicos
Entrada	cant info	32 bits 16 bits	1 n
Auxiliar	i parts [4] contador	32 bits 16(4) bits 32 bits	n n n^2
Salida	idEdifice numUser numFloors numOfficeForFloor user	16 bits 32 bits 32 bits 32 bits 16 bits	n n^2 n n n

Complejidad espacial total:

$Entrada + auxiliar + salida = 1 + n + n^2 = O(n^2)$

Complejidad auxiliar:

$n + n^2 = O(n^2)$

Complejidad espacial Auxiliar + Salida:

$n + n^2 = O(n^2)$

Análisis de complejidad temporal y espacial del algoritmo star:

`public void printInfo() {`	
`System.out.println(id);`	
`System.out.println();`	
`ArrayList<T> Discarded=new ArrayList();`	
`for(int i=0;i<floors.size();i++) {`	n
`Discarded.add(floors.poll());`	n
`Floor n=(Floor) Discarded.get(i);`	n
`System.out.println(n.getId());`	
`}`	
`System.out.println();`	
`java.util.Enumeration e=office.keys();`	
`while(e.hasMoreElements()) {`	n
`System.out.println(e.nextElement());`	n
`}`	
`}`	

Complejidad temporal:

$f(n) = n$

$O(f(n)) = n$

$O(n)$

La complejidad temporal del algoritmo star es $O(n)$.