Análisis de complejidad temporal

Para estos dos análisis se considerará siempre el peor caso. En este contexto sería cuando los condicionales siempre se cumplen.

Método print:

public String print(int c) {

String out = "[";

for (int i = 0; i <= c; i++) {

if (edificio.containsKey(i)) {

out += " " + edificio.get(i);

if (c > 1 && i < c) {

out += ",";

}

}

}

return out + " ]";

}

|  |  |
| --- | --- |
| String out = "["; | 1 |
| for (int i = 0; i <= c; i++) { | n |
| if (edificio.containsKey(i)) { | n |
| out += " " + edificio.get(i); | n |
| if (c > 1 && i < c) { | n |
| out += ","; | n |
| return out + " ]"; | 1 |

Al sumarlos, encontramos: 5n +2, por lo que podemos considerar al algoritmo como uno de omplejidad n. Por ende está acotado superiormente por O(n^3).

Método ascensor:

public String ascensor(Persona p, int oficinas) {

if (p.getOficina() > oficinas) {

return "La oficina no se encontro en el edificio, " + p.getNombre()

+ " se quedo encerrado, llamen a los bomberos";

} else if (edificio.containsKey(p.getOficina())) {

return p.toString2();

} else if (!edificio.containsKey(p.getOficina())) {

edificio.put(p.getOficina(), p.getNombre());

return p.toString();

}

return "";

}

|  |  |
| --- | --- |
| if (p.getOficina() > oficinas) { | 1 |
| return "La oficina no se encontro en el edificio, " + p.getNombre()  + " se quedo encerrado, llamen a los bomberos"; | 1 |
| else if (edificio.containsKey(p.getOficina())) { | 1 |
| return p.toString2(); | 1 |
| else if (!edificio.containsKey(p.getOficina())) { | 1 |
| edificio.put(p.getOficina(), p.getNombre()); | 1 |
| return p.toString(); | 1 |
| return ""; | 1 |

Al sumarlos, encontramos: 8, por lo que podemos considerar al algoritmo como uno de complejidad 1. Por ende, está acotado superiormente por O(9).