Documentación del primer punto

Descripción.

El algoritmo usa salto de múltiples tamaños que permiten encontrar con espacios superiores a uno.

1. Creo una función que permita hacer saltos de diferentes tamaños.
2. Repito los saltos hasta encontrar el tamaño mayor posible que lo determino como la máxima cantidad de terrenos.
3. Para determinar el mayor beneficio realizo almaceno el beneficio en una variable y el terreno que lo otorga y cada vez que termino de recorrer una posible combinación compara para determinar si el beneficio es mayor al que ya tengo al encontrar un mayor resultado remplazo la posibilidad del terreno.

**Seudocódigo.**

**ganancia = 0;**

**Valor = ValorI;**

**ArrayList<Zona> temporal = new ArrayList<>();**

**for (int i = Valor; i < TerreniSize; i += salto) {**

**temporal.add(Terrenos.get(i));**

**ganancia += Terrenos.get(i).getBeneficio();**

**}**

**if (ganancia > verificar) {**

**firstOption = (ArrayList<Zona>) temporal.clone();**

**verificar = ganancia;**

**temporal.clear();**

**ganancia = 0;**

**if (salto < TerreniSize) {**

**calculo(salto + 1, Valor);**

**}**

**} else {**

**// Si no se cumple la condicion reinica la longitud del arreglo**

**temporal.clear();**

**ganancia = 0;**

**if (salto < TerreniSize) {**

**calculo(salto + 1, Valor);**

**}**

**}**

**4 Pruebas**

|  |  |  |
| --- | --- | --- |
| Numero de Terrenos | Archivo | Tiempo de ejecución |
|  | Test\_1 |  |
|  | Test\_2 |  |
|  | Test\_3 |  |
|  | Test\_4 |  |
|  | Test\_5 |  |
|  |  |  |