

## Práctica 3: Algoritmos Greedy

Alejandro Rodríguez López  
alexrodriguezlop@correo.ugr.es

29 de abril de 2018

## Presentación del problema

**Asignación de tareas.** Un electricista necesita hacer  $n$  reparaciones urgentes, y sabe de antemano el tiempo que le va a llevar cada una de ellas. Como en su empresa le pagan dependiendo de la satisfacción del cliente y esta es inversamente proporcional al tiempo que tardan en atenderles, necesita decidir el orden en el que atenderá los avisos para minimizar el tiempo medio de atención de los clientes (desde el inicio hasta que su reparación es efectuada).



Técnicas para dar solución al problema:

- Solución “Algoritmos Greedy o Voraces”

# Planteamiento de la solución: GREEDY

Planteo la solución mas óptima asignando las tareas de forma creciente en función a su tiempo de ejecución.

## Esquema Greedy:

- **Candidatos:** Consideramos el conjunto de candidatos como las tareas a ejecutar.
- **Función Selección:** Elegiremos siempre la tarea cuya realización conlleve un menor tiempo.
- **Función de factibilidad:** Valoramos que la tarea no haya sido seleccionada previamente.
- **Función de solución:** Deben haber sido asignadas todas las tareas.
- **Función objetivo:** Asignar todas las tareas de la forma mas óptima posible entendiendo ésta como la ordenación que produce el tiempo de espera mínimo

## Planteamiento de la solución: GREEDY

---

"Seudocódigo"

```
1 optimizacionTareas(int [n] conjuntoTareas, int numTareas){
2
3     while(Queden candidatos y no se haya encontrado la solución){
4
5         //Selecciono una tarea
6         tarea = tareaMenortiempo(conjuntoTareas);
7
8         //Si la tarea es factible, la incluyo en la solución
9         solucion.add(tarea);
10
11        //La elimino del vector conjuntoTareas
12        conjuntoTareas.erase(tarea);
13    }
14    return solucion;
15 }
```

---

Figura 1: Seudo código de la solución planteada para resolver el problema con un solo electricista.

## Demostración

Si valoramos todas las permutaciones posibles obtenemos los resultados mostrados en la tabla 1.

Cuadro 1: Permutaciones posibles

T1 (5)	T2 (10)	T3 (3)	$\sum_t^{t_n}$
T1	T2	T3	$5+(5+10)=20$
T1	T3	T2	$5+(5+3)=13$
T2	T1	T3	$10+(10+5)=25$
T2	T3	T1	$10+(10+3)=23$
T3	T1	T2	$3+(3+5)=11$
T3	T2	T1	$3+(3+10)=16$

El algoritmo Greedy propuesto nos da una solución que coincide con la óptima por lo que el criterio de selección propuesto siempre conduce a una solución óptima.

# VARIOS ELECTRICISTAS: Presentación del problema

**Asignación de tareas.** En esta variante del problema hay varios electricistas, lo cual nos obliga a reestructurar el algoritmo.



Técnicas para dar solución al problema:

- Solución “Algoritmos Greedy o Voraces”

## Planteamiento de la solución: GREEDY

Planteo la solución asignando la tarea en orden de tiempo de ejecución de mayor a menos a aquel electricista cuya última tarea asignada tenga el tiempo de ejecución mas pequeño y ademas tenga el menor número de tareas asignadas.

De esta forma se obtendrá la solución mas óptima al problema planteado.

### **Esquema Greedy:**

- Consideraremos el conjunto de candidatos como las tareas a ejecutar, aunque también se podrían seleccionar los electricistas.
- Función Selección: Elegiremos siempre la tarea cuya realización conlleve un mayor tiempo para poder hacer el reparto de forma equitativa.
- Función de factibilidad: Valoramos que la tarea no haya sido seleccionada previamente.
- Función de solución: Deben haber sido asignadas todas las tareas de forma equitativa entre los electricistas.
- Función objetivo: Asignar todas las tareas de la forma mas óptima posible entendiendo ésta como la ordenación que produce el tiempo de espera mínimo

# Planteamiento de la solución: GREEDY

---

"Seudocódigo"

```
1 optimizacionTareas(int [n] conjuntoTareas, int numElectricistas){
2
3     //Mientras queden candidatos y no se haya encontrado la solucion
4     while(!conjuntoTareas.empty()){
5
6         //Selecciono una tarea
7         tarea = tareaMayorTiempo(conjuntoTareas);
8
9         //Buscamos a que electricista asignar la tarea.
10        while(queden electricistas sin comprobar){
11            //Si la tarea es factible, la incluyo en la solucion
12            //Comprueba entre aquellos en los que no se a llevado
13            //a cabo la ultima insercion
14            if((TareasElectricista == menosTareasAsignadas &&
15                UltimaTareaElectricista == tareaMasPequeñaAsignada){
16                insercion = lectura;
17            }
18            ++ lectura;
19        }
20        //Asignamos la tarea a la cola del electricista seleccionado.
21        (*insercion).push(*tarea);
22
23
24        //La elimino del vector conjuntoTareas
25        conjuntoTareas.erase(tarea);
26    }
27
28    return solucion;
29 }
30 }
```

---

## Demostración

Asignaremos la tarea en orden de tiempo de ejecución de mayor a menos a aquel electricista cuya última tarea asignada tenga el tiempo de ejecución mas pequeño y ademas tenga el menor número de tareas asignadas.

De esta forma es reparto será equitativo y se encontrará la solución mas óptima.

Cuadro 2.3: Reparto de tareas 1

	T1 (5)	T2 (10)	T3 (3)	T4 (20)	T5 (14)	T6 (18)	T7 (32)	T8 (10)	Sum
E1			x						35
E2	x			x			x		25
E3		x				x			28
E4					x			x	24

En la tabla podemos observar como repartiendo las tareas con el método anteriormente descrito el reparto se hace de manera equitativa.

# Demostración

Cuadro 2.4: Reparto de tareas 2

	T1 (5)	T2 (10)	T3 (3)	T4 (20)	T5 (14)	T6 (18)	T7 (32)	T8 (10)	Sum
E1			x		x				17
E2	x					x			23
E3		x		x					30
E4							x	x	42

En la tabla podemos observar como repartiendo las tareas de forma ascendente y rotativa el reparto no llega a ser equitativo.

## CONCLUSIONES:



*¿PREGUNTAS?*

Gracias por su atención! . . .