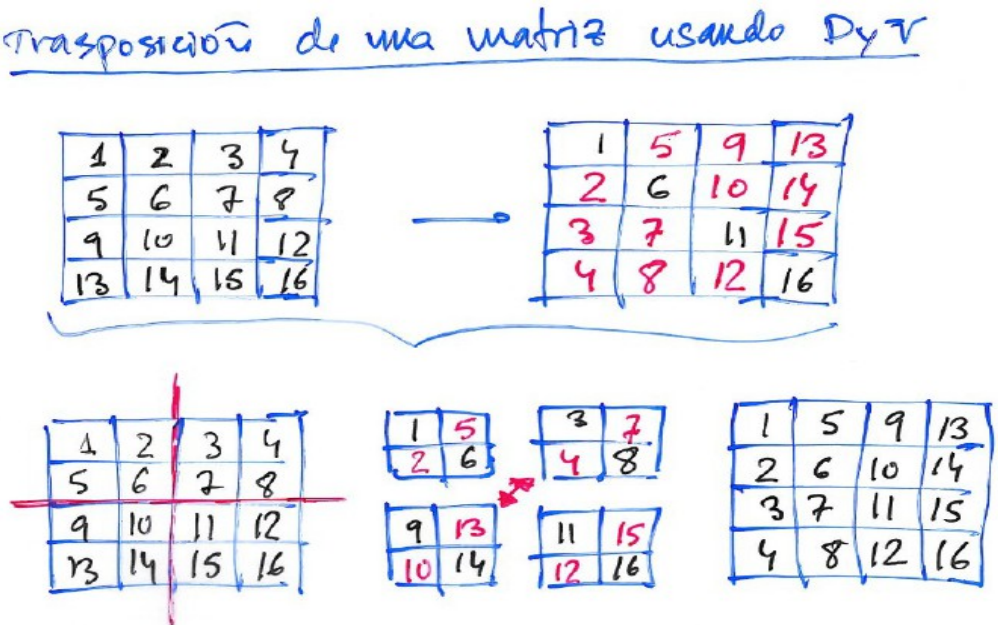
RETOS ALGORÍTMICA

Javier Gómez Luzón
2º C

Reto de eficiencia. Decir si son verdaderas o falsas.

$3n^2 \in O(n^2)$	$3n^2 \in \Omega(n^2)$	$3n^2 \in \Theta(n^2)$	$2^{n+1} \in O(2^m)$	$O(n) \in O(n^2)$
Verdadero	Verdadero	Verdadero	Verdadero si n<m	Verdadero
$n^2 \in O(n^3)$	$n^2 \in \Omega(n^3)$	$n^2 \in \Theta(n^3)$	$(2+1)^n \in O(2^n)$	$(n+1)! \in O(n!)$
Verdadero	Falso	Falso	Falso	Verdadero
$n^3 \in O(n^2)$	$n^3 \in \Omega(n^2)$	$n^3 \in \Theta(n^2)$	$(2+1)^n \in \Omega(2^n)$	$n^2 \in O(n!)$
Falso	Verdadero	Falso	Verdadero	Verdadero

Reto divide y vencerás.



Iremos llamando recursivamente a la función dividiendo la matriz en cuadrantes e intercambiando valores.

```
void intercambiar(vector<vector<int> > m, int iniF, int iniC, int finF, int finC){
	int f=(iniF+finF)/2;
	int c=(iniC+finC)/2;

	for(int i=f;i<=finF;i++){
		for(j=iniC;j<=c;j++){
			swap(m[i][j], m[i-f][c+j]);
		}
	}
}


int trasposicionMatriz(vector<vector<int> >m, int iniF, int iniC, int finF, int finC){
	if( (iniF==finF) && (iniC==finC)) return 0;
	else{
		int f=(iniF+finF)/2;
		int c=(iniC-finC)/2;

		int a=trasposicionMatriz(m,iniF,iniC,f,c);
		int b=trasposicionmatriz(m,iniF,c+1,f,finC);
		int c=trasposicionmatriz(m,f+1,iniC,finF,c);
		int d=trasposicionmatriz(m,f+1,c+1,finF,finC);

		intercambiar(m,iniF,iniC,finF,finC);
		return (a+b+c+d);
	}
}
```

No encontraba otra forma de hacer que la función se detuviese.

Reto divide y vencerás. Dado un vector que contiene secuencias de números enteros, diseñar un algoritmo D y V que determine la longitud de la secuencia más larga de una determinada clave k

6 6 4 4 4 4 1 6 6 6 6 3 3 3

Iremos dividiendo hasta llegar al caso base (un sub vector de tamaño 1). Si es igual a K devolveremos 1, de lo contrario. Iremos juntando todo lo obtenido. Y devolveremos la secuencia de K.

```
int secuenciaMasLarga(vector<int> & v, int k, int ini, int fin){
	if( (fin-ini)==0){
		if(vector[ini]==k) return 1;
		else return 0;
	}
	else{
		int mitad=(ini+fin)/2;
		int first=secuenciaMasLarga(v,ini,mitad);
		int second=secuenciaMasLarga(v,mitad+1,fin);
		if(v[mitad]==k && v[mitad+1]==k) return (first+second);
		else return max(first, second);
	}
}
```

Reto del fontanero.

Tiempos de espera:

Tº espera 2= Tº espera 1

Tº espera 3=Tº espera 2+ Tº espera 1

Tº espera 4= Tº espera 3+ Tº espera 2

….

Hay que minimizar el tiempo de espera.

Tenemos 4 tareas:

Tarea 1= 5 horas.

Tarea 2=3 horas

Tarea 3= 8 horas

Tarea 4= 2 horas

Posibles combinaciones:

1,2,3,4
1,2,4,3
1,3,2,4
4,2,1,3
….

Para escoger el orden lo haremos actuando como un algoritmo voraz. Elegiremos cada vez la tarea que tarde menos tiempo de entre las tareas libres que queden por hacer.

Primer la tarea 4, después la 2, después la 1 y por último la 3. Es decir:

2+(2+3)+(5+5)+(10+8)=35 horas de espera en total.

Tiempo de espera medio= 35/4=8,75 Horas. Este es el tiempo de espera medio menor.