Jesús Jiménez Sánchez

Algorítmica

Relación 2: Divide y vencerás

Ejercicio 1

```
int Indice(vector<int> enteros){
	int mitad;

	if(enteros.size() <= 1){
		if(enteros[0] == 0){
			return 0;
		}
		else{
			return -1;
		}
	}
	else{
		mitad = (int)(enteros.size() / 2);

		if(enteros[mitad] == mitad){
			return mitad;
		}
		else{
			if(enteros[mitad] > mitad){
				vector<int> recursivo = enteros;
				auto it = recursivo.begin();

				recursivo.erase(it, it + mitad);
				//Borramos la primera mitad del vector

				return Indice(recursivo);
			}
			else{
				vector<int> recursivo = enteros;
				auto it = recursivo.begin();

				recursivo.erase(it + mitad, recursivo.end());
				//Borramos la segunda mitad del vector

				return Indice(recursivo);
			}
		}
	}

	return 0;
}
```

Jesús Jiménez Sánchez

```
int IndiceAlReves(vector<int> enteros){
	int mitad;

	if(enteros.size() <= 1){
		if(enteros[0] == 0){
			return 0;
		}
		else{
			return -1;
		}
	}
	else{
		mitad = (int)(enteros.size() / 2);

		if(enteros[mitad] == mitad){
			return mitad;
		}
		else{
			if(enteros[mitad] < mitad){
				vector<int> recursivo = enteros;
				auto it = recursivo.begin();

				recursivo.erase(it, it + mitad);
				//Borramos la primera mitad del vector

				return Indice(recursivo);
			}
			else{
				vector<int> recursivo = enteros;
				auto it = recursivo.begin();

				recursivo.erase(it + mitad, recursivo.end());
				//Borramos la segunda mitad del vector

				return Indice(recursivo);
			}
		}
	}

	return 0;
}
```

La eficiencia en el peor caso es de $O(\log(n))$.

Jesús Jiménez Sánchez

Ejercicio 2

```
vector< pair<int, int> > Clasificacion(vector< vector<char> > matriz_resultados,
int &fil, int &col){
	vector< pair<int, int> > resultado, recursivo;
	pair<int, int> posicionGanador;
	const int COLUMNAS = (int)matriz_resultados.at(0).size() - 1;
	const int FILAS = (int)matriz_resultados.size() - 1;
	const char GANA = 'G';
	char elemento;

	if (fil != col){ // Sino son el mismo elemento
		elemento = matriz_resultados[fil][col];

		if (elemento == GANA){
			posicionGanador.first = fil;
			posicionGanador.second = col;

			resultado.push_back(posicionGanador);
		}
	}

	if(fil < FILAS){
		if(col == COLUMNAS){
			col = 0;
			fil++;
		}
		else{
			col++;
		}

		recursivo = Clasificacion(matriz_resultados, fil, col);

		resultado.insert(resultado.end(), recursivo.begin(),
recursivo.end());
	}

	return resultado;
}
```

La eficiencia en el peor de los casos es de $O(n)$.

Jesús Jiménez Sánchez

Ejercicio 3

```
vector< pair< pair<int, int>, int > > Skyline(vector< pair< pair<int, int>, int
> > edificios, int &num, int &antes){
	//La función debe llamarse siempre con un 0 en num
	if(num == 0){
		QuickSort(edificios, 0, (int)edificios.size());
	}

	vector< pair< pair<int, int>, int > > silueta, recursivo;
	pair< pair<int, int>, int > sombra;

	if(num < edificios.size()){
		if(edificios[num].second > edificios[num + 1].second){
			if(num == 0){
				sombra.first.first = edificios[num].first.first;
			}
			else{
				sombra.first.first = antes;
			}

			sombra.first.second = edificios[num].first.second;
			sombra.second = edificios[num].second;
			silueta.push_back(sombra);
		}
		else{
			if(num == 0){
				sombra.first.first = edificios[num].first.first;
			}
			else{
				sombra.first.first = antes;
			}

			sombra.first.second = edificios[num + 1].first.first;
			sombra.second = edificios[num].second;
			silueta.push_back(sombra);
		}

		num++;
		antes = sombra.first.second;

		recursivo = Skyline(edificios, num, antes);

		silueta.insert(silueta.end(), recursivo.begin(), recursivo.end());
	}

	return silueta;
}
```

La eficiencia en el peor de los casos es de $O(n)$.

Jesús Jiménez Sánchez

Ejercicio 4

```
void TornillosTuercas(int *tornillos, int *tuercas, int n1, int n2){
	if(n1 < n2){
		int piv = Pivote(tornillos, n1, n2, n1);
		int i = n1;

		while (i <= n2 && tuercas[i] != tornillos[piv]){
			i++;
		}

		swap(tuercas[n1],tuercas[i]);
		Pivote(tuercas, n1, n2, n1);

		TornillosTuercas(tornillos, tuercas, n1, piv - 1);
		TornillosTuercas(tornillos, tuercas, piv + 1, n2);
	}
}
```

La eficiencia en el peor de los casos es de $O(n \cdot \log(n))$.