1 Problema 1

1.1 Análisis de Complejidad

Para el analisis de este algoritmo, lo podemos dividir en varias subpartes. La primera parte, seria la lectura de cada una de las pruebas. Esta lectura, es de orden $O(n)$, que que presenta un ciclo de tamaño N La segunda parte es la creacion de cada uno de los arcos posibles Esta, creacion, es de orden $O(n^2)$, ya que recorre la lista en orden cuadratico para conseguir generar todos los arcos posibles Se usa un algorimo de ordenamiento que viene con el lenguaje. Este algoritmo su complejidad es de $O(nlogn)$ La tercera parte y mas compleja es una busqueda de las componentes conexas. Esta trabaja con una lista y su recorrido de orden $O(n)$ y adicionalmente, unas consultas. Y una busqueda en otra lista. Cada busqueda es de orden $O(componentes_conexas)$. Es decir que la busqueda es sobre un grafo mucho mas pequeño que N, por consiguiente, podriamos considerar que esta tercera parte es de orden $O(n)$

1.2 Psucódigo

Algoritmo 1 Calcular *total*, que es el total de actividades que puede realizar el empleado

Entrada: C Cantidad de tareas, A Lista de tuplas de actividades
Salida: Entero que indica la cantidad maxima de actividades que puede realizar

```
1: entero ultimo = −1
2: devolver total
```

1.3 Explicación de la solución dada

2 Problema 2

2.1 Análisis de Complejidad

Este Algoritmo se parece mucho al primero, este tiene un while de orden $O(n)$, pero es depreciado porque este N representa la cantidad de pruebas y no deberia ser un numero muy grande. Despues, internamente exiten 3 ciclos, cada uno de orden $O(n)$ Cada uno de estos ciclos, son ciclos de lectura y de acomodacion de datos. Su complejidad no es muy grande para números pequeños, pero se podría complicar para números muy grandes o para pruebas con números muchos mas grandes. El siguiente ciclo, es un for externo que llama a una funcion Unir internamente El ciclo es de orden $O(n)$, luego la

funcion de interna, tiene una llamada a otra funcion auxiliar que es recursiva. Esta funcion recursiva tambien tiene un comportamiendo de orden $O(n)$ ya que se sabe que nunca sera mayor que N Podriamos decir entonces, que este ciclo, y por consiguiente la funcion interna tienen orden $O(n^2)$, ya que hay un ciclo interno y otro externo que estan ciclando y uno depende el otro. Por ultimo, quedan dos ciclos, cada uno de orden $O(n)$, son ciclos de recorrido de los vectores. Tomando en consideracion todos estos ordenes internos a nuestro algoritmo, podemos decir que nuestro algoritmo se comporta en el peor caso como un algoritmo de orden $O(n^2)$

2.2 Psucódigo

Algoritmo 2 Calcular *total*, que es el total de actividades que puede realizar el empleado

Entrada: C Cantidad de tareas, A Lista de tuplas de actividades
Salida: Entero que indica la cantidad maxima de actividades que puede realizar

```
1: entero ultimo = −1
2: devolver total
```

2.3 Explicación de la solución dada

3 Tercer problema

3.1 Análisis de Complejidad

Primera parte es una lectura de la cantidad de pruebas que tiene el caso. Esta primera parte es $O(n)$, ya que es un ciclo de tamaño N. Luego, tenemos para cada empleado un ciclo por el numero de actividades que este realizara. Este ciclo es de tamaño n, por consiguiente, el proceso es $O(n)$ Se usa un algorimo de ordenamiento que viene con el lenguaje. Este algoritmo su complejidad es de $O(nlogn)$ Por ultimo, se realiza un ciclo de tamaño N, este ciclo tiene una complejidad de $O(n)$ Como tenemos el primer ciclo por afuera, y todos los demas internos, podriamos pensar que entonces la complejidad del problema es de $O(n^2 logN)$. Pero, al ser el primer ciclo un ciclo de iteraciones de pruebas, donde el numero no sera muy grande, podriamos pensar entonces que la complejidad de todo el algoritmo es la complejidad mas grande. Esta seria $O(nlogN)$

3.2 Psucódigo

3.3 Explicación de la solución dada

erbgtebhryinhtrinbtrnitnkhbnrynht bjutrhbihntnn

Algoritmo 3 Calcular *total*, que es el total de actividades que puede realizar el empleado

Entrada: C Cantidad de tareas, A Lista de tuplas de actividades
Salida: Entero que indica la cantidad maxima de actividades que puede realizar

```
1: entero ultimo = −1
2: entero total = 0
3: ordenar(A)
4: para cada elemento de A hacer
5:    si ultimo > elemento.primero entonces
6:       total = total + 1
7:    fin si
8:    ultimo = elemento.segundo
9: fin para
10: devolver total
```