Diseño y Análisis de Algoritmos
Proyecto 2017-10
Problema B
Juan Sebastian Barragan – Stephannie Jimenez
201212774 201423727

1 Algoritmo de solución

Para solucionar este problema de manera eficiente, se encontró que usar programación dinámica seria deseable. Esto se debe, a que se quiere encontrar cual es la combinación más conveniente para que el socio logre hacer la coalición que le maximice el valor de su prima.

Para ello, se realizó el método `calcularMaximaNomina(int[] p, double maxNomina, double sumaAcciones, int indiceSumado)`. Como precondición se tiene que ninguno de los parámetros será nulo, se dará un arreglo con los porcentajes p dados por consola que corresponde a la participación que tiene cada socio en la empresa. También se tienen tres parámetros que indican la máxima Nómina ya vista, la suma de las acciones de los posibles socios con los que se desea hacer la coalición y por último, el índice actual para indicar la posición en el arreglo.

Este método, se planteó con recursividad. El caso base esta dado cuando la suma de las acciones, `maxNomina,` es mayor a 50. Cuando esto sucede, es necesario calcular la prima con los actuales socios contemplados. Para ello, se utiliza la fórmula dada en el enunciado, donde nomina= (p[p.length-1]*100)/sumaAcciones. En caso de que esta sea mayor a las que yo ya he visto, actualizo el valor de la variable maxNomina.

En cambio, cuando la suma de las acciones es menor a 50, el algoritmo entra en la parte recursiva. Es importante recalcar, que el recorrido lineal se está realizando de derecha a izquierda en el vector que contiene los porcentajes. En primer lugar, la recursión funciona de manera que se intentan todas las combinaciones desde el último socio hasta el primero. Al realizar esto, se busca el máximo local entre ellos. Seguidamente, se realiza una comparación entre el máximo que indica si es mejor quedarme con el socio actual o avanzar una posición y buscar realizar la coalición con otro.

En cuanto a la lectura de los datos, se organizó tal que en un ciclo se generará el vector que contiene los porcentajes de cada socio ingresado. Seguido a esto, se llama a la función mencionada previamente con valores iniciales de cero para la maxNomina, el valor del porcentaje del socio n en sumaAcciones, y el último índice para indiceSumado. Después de calcular la máxima nomina posible, se devuelve este valor por consola redondeada a dos cifras decimales.

2 Análisis de complejidades espacial y temporal

2.1 Complejidad espacial

En primer lugar, se está generando un vector con n porcentajes que ingresan por parámetro. Adicionalmente, se tienen tres variables numéricas que se almacenan para poder realizar todos los cálculos. Por esta razón se tiene que la complejidad espacial de la solución es $O(n)$.

2.2 Complejidad temporal

La complejidad temporal de este problema está dado por el número de veces que es necesario calcular la máxima nómina. Es decir, cuantas veces tengo que revisar mi lista de porcentajes de cada socio para poder definir cuantos elementos hay que revisar, y por ende cuanto tiempo consume. En primera instancia, se tiene que el tiempo consumido en una llamada del método que calcula la máxima nómina, tiene en su peor caso que recorrer todos los elementos del arreglo. Esto se realiza hasta que la suma acumulada es mayor o igual a 50. Por esta razón, se encuentra que en un caso donde tengo que recorrer todas las posibles combinaciones. De esta manera, se puede deducir que la complejidad temporal de la solución es $O(50*n)$.

3 Comentarios finales

Se puede verificar el tiempo de ejecución del programa a partir del uso de los casos de prueba dados. Al utilizarlos, se puede ver que el algoritmo en tiempo de ejecución es menor a medio segundo. Pese a esto, es un tiempo muy inestable. Esto se debe a que el tiempo fluctúa significativamente cada vez que se corre con un mismo caso de prueba.