Informe práctica 9:

Qué representan las componentes del vector s:

En este caso se esta resolviendo un problema de optimización en el que se busca asignar a los trabajadores las tareas de la forma más eficiente posible, para ello se usa un algoritmo de backtracking que prueba todas las combinaciones posibles (asigna todas las tareas a todos los trabajadores) y va guardando cada combinación en el vector s, es decir el vector s funciona como una variable auxiliar por la que van pasando todas las posibles soluciones al problema, como en este caso el problema consiste en asignar tareas a trabajadores las componentes de s mostraran que tarea se asigna a cada trabajador.

Qué función realiza la asignación de una tarea a una persona:

La función encargada de asignar tareas a los trabajadores es la función “generar” que asigna de forma secuencial las tareas a los trabajadores independientemente de que esas asignaciones estén permitidas o no, ya que en caso de asignarle una tarea no permitida a un trabajador la función criterio se dará cuenta por lo que esa combinación de tareas no podrá ser una solución al problema.

Qué comprueba la función MasHermanos:

En este caso particular comprueba si queda alguna tarea que aún no se haya probado a asignar a un trabajador x, en términos más generales indica si para un nivel x queda alguna otra opción por probar.

Cuál es el objetivo de la función Retroceder:

Volver atrás en el proceso de construcción de la solución actual para poder probar con otra posible solución en caso de que exista.

Diego Baleirón Pais

Conclusiones teóricas y prácticas de la complejidad de la tarea:

Desde un punto de vista teórico del problema se podría decir que el problema tiene complejidad factorial ya que para que una solución sea válida los trabajadores tienen que desempeñar tareas distintas (lo que implica que el árbol de soluciones es factorial), sin embargo la complejidad de este algoritmo esta en un punto medio entre exponencial y factorial, ya que a pesar de no buscar entre todas las combinaciones posibles de tareas el algoritmo si prueba con muchas combinaciones que en teoría por la naturaleza del problema no son válidas. (Por como está implementado el problema es imposible que asigne a más de dos trabajadores la misma tarea, sin embargo es muy normal que asigne a dos trabajadores la misma tarea)

Además la complejidad de las dos versiones es la misma ya que el número de nodos generados es el mismo, sin embargo el tiempo de ejecución será sutilmente menor en la segunda versión ya que la función criterio pasa de tener complejidad lineal a constante. (Para que haya una diferencia visible en el tiempo de ejecución la entrada del algoritmo tiene que ser bastante más grande que la del ejercicio, con estas entradas la diferencia es imperceptible).

Pasos de las funciones:

Ejemplo	N.º de nodos válidos	N.º de veces que se ejecuta la función generar()	N.º de veces que se ejecuta la función solucion()	N.º de veces que se ejecuta la función criterio()	N.º de veces que se ejecuta la función mas-Hermanos()	N.º de veces que se ejecuta la función retroceder()
1 sin optimizar	15	30	58	88	40	10
1 optimizado	15	30	58	88	40	10
2 sin optimizar	1956	7422	14832	22254	8659	1237
2 optimizado	1956	7422	14832	22254	8659	1237

Gracias a la tabla se puede ver claramente lo expuesto anteriormente, por una parte la complejidad del algoritmo es la misma en las dos variaciones del algoritmo ya el algoritmo tiene que examinar los mismos nodos y por otro lado se ve el que número de nodos válidos es coherente con los apuntes de teoría, los cuales dicen que el número de nodos de un árbol permutacional se puede obtener con el siguiente sumatorio → $n + n(n-1) + n(n-1)(n-2) + \ldots + n!$