Primero comenzamos con la importación de bibliotecas e inicialización, en la cual vamos a tener una variable N que será para indicar el numero de trabajadores y tareas con el cual vamos a trabajar, luego vamos a generar una matriz cuadrada o matriz bi dimensional en la cual N será el numero de filas y columnas de la misma.

Luego trabajamos las cadenas para poder visualizar la matriz de la siguiente manera (Foto de consola).

Generamos e inicializamos un arreglo que va a contener la asignación actual que se va generar a partir del backtracking, en la cual cada posición de arreglo es un trabajador y el valor int que esta contiene es la tarea asignada. Esta se inicializa en -1 porque al asignar las tareas a cada trabajador en las diferentes pasadas se debe comprobar primero que esta no haya sido asignada a otro trabajador, por lo que al inicio ninguno tendrá asignada ninguna tarea.

Aqui creamos la variable en la cual se invocara la función que trabajara con el backtraking, antes de pasar al código de esta, generamos las cadenas para después poder visualizar la asignación realizada a cada trabajador.

Para comenzar con la función de backtracking, es importante tener en cuenta que pasaremos como parámetros, la tabla bidimensional, el arreglo o vector que contendra la asignación actual de los trabajadores, la variable N, un parámetro definido por defecto PASOS que indica el trabajador al que se le esta asignando una tarea y además, sirve de variable centinela para saber si hemos alcanzado una solución total, es decir un caso base, y el parámetro mejor asignación, que como su nombre indica contendra la asignación ideal de las tareas a los trabajadores teniendo en cuenta la sumatoria de los tiempos, este parámetro trae por defecto el valor None, ya que cuando se invoque en el programa principal no se le pasa ningún valor.

Como primera parte, tratamos que sucede cuando la Mejor asignación es NONE, lo que se realiza es inicializar el vector o arreglo MEJOR y asignarle la primera tarea a cada trabajador, es decir, al trabajador 1 se le asigna la tarea 1, y asi sucesivamente con los N trabajadores y tareas, luego se invoca la misma función nuevamente, ahora con nuestro vector mejor asignación inicializado.

Una vez accedida, como nuestro vector ya no esta vacio, accederemos por el sino de nuestro condicional, en esta segunda parte de la función, analizaremos si hemos llegado al caso base o no, es decir, paso < N. Si no se llego a dicho caso, es decir que se cumple la condición, se toman cada una de las tareas y se analizan si están o no asignadas, cuando se haya una tarea que no este asignada, se le asigna a ese trabajador, indicado por la variable paso, y se vuelve a invocar la función, ahora con el paso aumentado en uno, es decir, que ahora se le asigna tarea al siguiente trabajador.

Al llegar a nuestro caso base o posible solución, es decir que todos los trabajadores tienen una tarea, iremos por nuestro sino de esta 2da parte, en la cual realizaremos dos sumatorias, una de nuestra mejor asignación, que es importante destacar que estaba con una asignación ‘predeterminada’, y una 2da sumatoria de la asignación actual realizada, luego comparamos dichas sumatorias para ver si llegamos a una mejor solución, si esto sucede, la mejor asignación, pasa a ser nuestra asignación actual.

Una vez llegado a este punto, se comienzan a desapilar los registros de activación y re asignando los trabajadores con el valor -1, en busca de una nueva posible solución, es decir se realizaran pasos hacia atrás hasta encontrar un nuevo trabajador libre para asignar una tarea, teniendo en cuenta el punto del bucle en el cual se realizo la invocación.

En conclusion, teniendo en cuenta las características de nuestra técnica, en nuestra búsqueda de la solución, no detenemos el algoritmo ante la presencia de una solución completa, sino que analizamos cada una de las soluciones posibles para encontrar no solo una solución completa sino tambien la que tenga el mejor tiempo. En segunda parte, en nuestro caso no es posible llegar a una hoja que sea solución parcial, es decir todas las hojas son soluciones completas.El resto de características se mantienen en el algoritmo, como por ejemplo la construcción de las soluciones añadiendo elementos diferentes. Por ultimo, nunca nos vemos limitados a la regiones o lotes de pruebas en las cuales vamos a poder encontrar una solucion, ya que como se menciono, no tenemos soluciones parciales hoja que nos limiten.

Trabajador 1 Trabajador 2 Trabajador 3 Trabajador 4

|  |  |  |  |
| --- | --- | --- | --- |
| 2 | 1 | 3 | -1 |









