**Estructuras de Datos y Algoritmos**

**Taller**

**Licenciatura en Ciencias de la Computación UNSJ**

**Tema**: Técnicas de Diseño de Algoritmos

**Técnica de Diseño**: Backtracking

**Ejemplo abordado**: Asignación de tareas

**Autores**:

- Benemérito Ignacio Registro: 21036

- Peña Juan Registro: 21151

**Año académico**: 2022

Índice

[Sobre la técnica de diseño 3](#1%19Sobre_la_t%C3%A9cnica_de_dise%C3%B1o%19C)

[Sobre la problemática tratada 3](#1%19Sobre_la_problem%C3%A1tica_tratada%19C)

[Código fuente del algoritmo 3](#1%19C%C3%B3digo_fuente_del_algoritmo%19C)

[Seguimiento del algoritmo con un ejemplo concreto 5](#1%19Seguimiento_del_algoritmo_con_un_ejemplo_concreto%19C)

[Conclusión 7](#1%19Conclusi%C3%B3n%19C)

[Bibliografía 7](#1%19Bibliograf%C3%ADa%19C)

**obre la técnica de diseño**

La vuelta atrás o backtracking se basa en la construcción de soluciones parciales hasta llegar a una solución total, donde se considera que el recorrido realizado fue exitoso. En este caso particular (asignación de tareas), dicho algoritmo no se detiene aún luego de alcanzar una solución total, ya que no solo se busca una solución total, sino la mejor solución total de todas.

La asignación de tareas es un problema que consiste en asignar de la mejor manera posible un conjunto de tareas a un conjunto de trabajadores o procesadores para minimizar el tiempo total que dichas tareas insumen, es decir, lograr la asignación con menor tiempo total posible.

CODIGO DE LA SOLUCION A LA PROBLEMÁTICA PLANTEADA

import random

import numpy as np

def backtracking(tabla:np.ndarray, asignacion\_actual:np.ndarray, N:int, paso=0, mejor\_asignacion = None):

    if mejor\_asignacion is None:

        mejor = np.empty(N, dtype=int)

        for i in range(N):

            mejor[i] = i

        backtracking(tabla=tabla, asignacion\_actual=asignacion\_actual, N=N, paso=0, mejor\_asignacion=mejor)

        return mejor

    else:

        if paso < N:

            for i in range(N):

                if i not in asignacion\_actual:

                    asignacion\_actual[paso] = i

                    backtracking(tabla=tabla, asignacion\_actual=asignacion\_actual, N=N, paso=paso+1, mejor\_asignacion=mejor\_asignacion)

                    asignacion\_actual[paso] = -1

        else:

            tiempo\_actual = 0

            mejor\_tiempo = 0

            for i in range(N):

                tiempo\_actual += tabla[i][asignacion\_actual[i]]

                mejor\_tiempo += tabla[i][mejor\_asignacion[i]]

            if tiempo\_actual < mejor\_tiempo:

                for i in range(N):

                    mejor\_asignacion[i] = asignacion\_actual[i]

if \_\_name\_\_ == "\_\_main\_\_":

    N = 3

    random.seed(20)

    tabla\_trabajadores\_tareas = np.empty((N, N), dtype=int)

    for i in range(N):

        for j in range(N):

            tabla\_trabajadores\_tareas[i][j] = random.randrange(1, 21)

    print("Tabla de trabajadores y tareas")

    cadena = "{0:<13}".format("Trabajador")

    for i in range(N):

        cadena += "{0:<10}".format("Tarea {0}".format(i+1))

    for i in range(N):

        cadena += "\n{0:<13}".format(i+1)

        for j in range(N):

            cadena += "{0:<10}".format(tabla\_trabajadores\_tareas[i][j])

    print(cadena)

    asignacion\_actual = np.empty(N, dtype=int)

    asignacion\_actual.fill(-1)

    print("\n")

    mejor\_asignacion = backtracking(tabla=tabla\_trabajadores\_tareas, asignacion\_actual=asignacion\_actual, N=N)

    print("La mejor asignacion posible es: ")

    for i in range(N):

        print("Trabajador {0} con tarea {1}".format(i+1, mejor\_asignacion[i]+1))

    tiempo = 0

    for i in range(N):

        tiempo += tabla\_trabajadores\_tareas[i][mejor\_asignacion[i]]

    print("El tiempo para esta asignacion es {0}".format(tiempo))

Para este ejemplo se considerará la siguiente tabla que representa los tiempos que tardaría cada trabajador en realizar cada tarea

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tarea 1 | Tarea 2 | Tarea 3 |
| Trabajador 1 | 5 | 9 | 4 |
| Trabajador 2 | 11 | 19 | 6 |
| Trabajador 3 | 1 | 14 | 14 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| asignacion\_actual | | | N | paso | mejor\_asignacion | | | tiempo\_actual | mejor\_tiempo |
| -1 | -1 | -1 | 3 | 0 | None | | |  |  |
|  |  |  |  |  | 0 | 1 | 2 |  |  |
|  |  |  |  | 0 |  |  |  |  |  |
| 0 | -1 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  |  |
| 0 | 1 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 2 |  |  |  |  |  |
| 0 | 1 | 2 |  |  |  |  |  |  |  |
|  |  |  |  | 3 |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 38 | 38 |
|  |  |  |  | 2 |  |  |  |  |  |
| 0 | 1 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  |  |
| 0 | 2 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 2 |  |  |  |  |  |
| 0 | 2 | 1 |  |  |  |  |  |  |  |
|  |  |  |  | 3 |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 25 | 38 |
|  |  |  |  |  | 0 | 2 | 1 |  |  |
|  |  |  |  | 2 |  |  |  |  |  |
| 0 | 2 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  |  |
| 0 | -1 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 0 |  |  |  |  |  |
| 1 | -1 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  |  |
| 1 | 0 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 2 |  |  |  |  |  |
| 1 | 0 | 2 |  |  |  |  |  |  |  |
|  |  |  |  | 3 |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 34 | 25 |
|  |  |  |  | 2 |  |  |  |  |  |
| 1 | 0 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  |  |
| 1 | 2 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 2 |  |  |  |  |  |
| 1 | 2 | 0 |  |  |  |  |  |  |  |
|  |  |  |  | 3 |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 16 | 25 |
|  |  |  |  |  | 1 | 2 | 0 |  |  |
|  |  |  |  | 2 |  |  |  |  |  |
| 1 | 2 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  |  |
| 1 | -1 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 0 |  |  |  |  |  |
| 2 | -1 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  |  |
| 2 | 0 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 2 |  |  |  |  |  |
| 2 | 0 | 1 |  |  |  |  |  |  |  |
|  |  |  |  | 3 |  |  |  |  |  |
|  |  |  |  |  |  |  |  | 29 | 16 |
|  |  |  |  | 2 |  |  |  |  |  |
| 2 | 0 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  |  |
| 2 | 1 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 2 |  |  |  |  |  |
| 2 | 1 | 0 |  |  |  |  |  |  |  |
|  |  |  |  | 3 |  |  |  | 24 | 16 |
|  |  |  |  | 2 |  |  |  |  |  |
| 2 | 1 | -1 |  |  |  |  |  |  |  |
|  |  |  |  | 1 |  |  |  |  |  |
| 2 | -1 | -1 |  |  |  |  |  |  |  |
| -1 | -1 | -1 |  | 0 |  |  |  |  |  |

Como se puede ver, el arreglo retornado contiene la asignación 1, 2, 0, es decir, trabajador 1 con tarea 2, trabajador 2 con tarea 3 y trabajador 3 con tarea 1, el cual tiene un tiempo total asociado de 16, el menor de todas las posibles asignaciones.

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 también la que tenga el mejor tiempo. En segundo lugar, 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 solución, ya que como se mencionó, no tenemos soluciones parciales hoja que nos limiten.

Mark Allen Weiss, “Estructuras de datos y algoritmos”, Addison-Wesley, págs 418-432.