Laboratorio - 9

Alumno: Jackson Fernando Merma Portocarrero

Escuela Profesional: Ingeniería de Sistemas

CUI: 20202143

Correo: jmermap@unsa.edu.pe

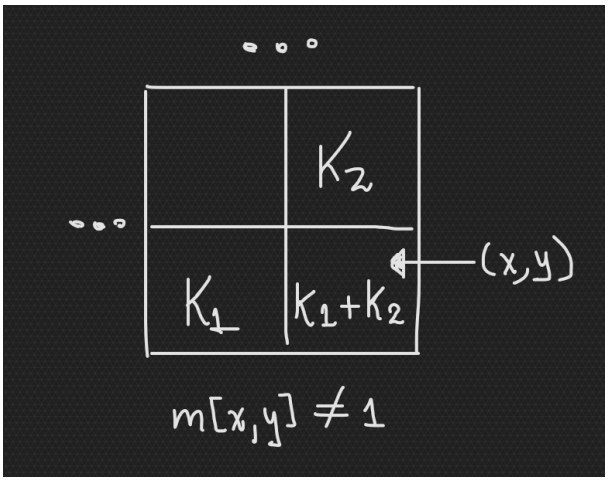
Generalidades

- Tanto las soluciones de cada problema como las evidencias del juez online se encuentran en subcarpetas de la carpeta raíz. (se indicará en cada uno de ellos).

Unique Path 2

+ **Nombre de la carpeta: unique_path**

Para su solución, simplemente se suman los dos posibles caminos de una posición determinada (x,y), estos caminos por las condiciones del problema son: arriba y la izquierda (x-1,y) y (x,y-1), esto se cumplira siempre que no se este en los obstáculos (x,y) != 1. Finalmente el resultado estará en el último dato de matriz. La siguiente imagen muestra el caso general de recorridos:



Book Shop

+ **Nombre de la carpeta: book_shop**

En este problema se maximiza el resultado como la típica implementación de “el problema de la mochila”; entonces, básicamente se llena la matriz principal tomando en cuenta la capacidad actual de la mochila y la capacidad de la opción ofrecida:

Tomando en cuenta que ‘k’ es la capacidad actual de la mochila y ‘h’ es la capacidad propuesta:

- Si k es menor que h, entonces solo se reemplaza con el ya maximizado de esa capacidad k.
- Si k es mayor que h, entonces se puede maximizar el resultado con la ganancia de la capacidad propuesta por h (pages[h]) más la capacidad propuesta por k-h (pages[k-h]).

Finalmente la mayor ganancia, estará en la última posición de la matriz.

Number of Longest Increasing Subsequence

+ **Nombre de la carpeta: longest_subsequence**

Para su solución se implementa todas las relaciones que puede tener un número con el subgrupo de número a su derecha. Entonces:

Tomando en cuenta que ‘k’ es el número a comparar y ‘m’ es la máxima cantidad de relaciones de el número en la posición ‘i’, entonces:

- Si k es menor que m, entonces la máxima relación será entre k + 1 vs el historial en relaciones posibles en i - 1.
- Si k es mayor, entonces, se consulta al historial en relaciones posibles en i - 1

Ejemplo:

```
10  9   2   5   3   7   101  18
1   1   1   1   1   1   2    2    ← 10 solo tiene relación con 101 y 18
    1   1   1   1   1   2    2
        1   2   2   2   2    2
            2   2   3   3    3    ← maximizados en una fila anterior
                2   3   3    3
                    3   4    4
                        4    4
                             4
```

Luego, se puede determinar 2 posibles soluciones siguiendo la técnica de **back tracking**:

Solución 1

```
10  9   2   5   3   7   101
18
1   1   1   1   1   1   2    2
    1   1   1   1   1   2    2
        1   2   2   2   2    2
            2   2   3   3    3
                2   3   3    3
                    3   4    4
                        4    4
                             4
```

Solución 2

```
10  9   2   5   3   7   101
18
1   1   1   1   1   1   2    2
    1   1   1   1   1   2    2
        1   2   2   2   2    2
            2   2   3   3    3
                2   3   3    3
                    3   4    4
                        4    4
                             4
```

- Estas no son todas las posibilidades, ya que el algoritmo da las soluciones más extremas; esto quiere decir que estas podrían encontrarse sobre toda la diagonal principal de la matriz.

Rectangle Cutting

+ **Nombre de la carpeta: rectangle_cutting**

En este problema se usará una variable (INF → answer=INF) para poder maximizar el resultado, se toman dos casos:

- Luego donde se corta un rectángulo horizontalmente, entonces:

  Total = min(answer, corte(k, y) + corte(x-k,y))
- Luego si se corta verticalmente:

  Total = min(answer, corte(x, k) + corte(x, y-k))
- De esta forma se deberá recorrer todos los posibles cortes verticales y horizontales. Por ejemplo:
  - Si se quiere determinar la menor cantidad de cortes en un rectángulo de 3 x 5:
    - Verticales → cortes de (1,4) y (2,3)
    - Horizontales → corte de (1,2)
    - Las demás posibilidades son repetitivas, por ello se maximiza hasta DIMENSION/2
- Finalmente el resultado estará en el último dato de la matriz principal.

Maximal Square

+ **Nombre de la carpeta: maximal_square**

Para su solución, simplemente se maximiza la dimensión [x,y], con el mínimo subcuadrado que se puede obtener de un cuadrilátero de dimensiones [x-1,y], [x,y-1] y [x-1,y-1]: