Prueba Práctica Competitiva
"Caminos minimales"

Supongamos que tenemos una cuadrícula rectangular con *m* filas, numeradas de *0* a *(m - 1),* y *n* columnas, numeradas de *0* a *(n - 1),* sobre la que podemos desplazarnos de una casilla a otra siguiendo un camino compuesto sólo por tramos horizontales y verticales. Esto es, desde la casilla *(i, j)* solo podemos pasar directamente a las casillas *(i, j + 1), (i, j - 1), (i + 1, j)* e *(i - 1, j)* (siempre y cuando estén dentro de la cuadrícula, claro). Cada casilla que se pise en el recorrido supone un coste que viene indicado por los elementos del vector *v : int array array*; esto es, el coste de pisar la casilla *(i, j)* es *v.(i).(j)*. Todos los costes son mayores que *0*. (Puede asumirse que la suma de los costes de todas las casillas de la cuadrícula no superará el valor *max_int)*

Defina una función
valid_path : int -> int -> int * int -> int * int -> (int * int) list -> bool,
de modo que ***valid_path m n ini fin path*** indique si path es un camino válido en la cuadrícula ***m x n*** para ir de la casilla ***ini*** a la ***fin***. Para ello, tanto la casilla *ini* como la *fin* han de estar dentro de la cuadrícula; la lista, que no puede contener casillas repetidas, debe tener como primer elemento la casilla ***ini*** y como último la casilla ***fin***, y de cada casilla de la lista debe poderse pasar directamente a la siguiente.

Defina una función
path_weight : int array array -> (int * int) list -> int,
de modo que ***path_weight v path*** sea el coste de pisar todas las casillas de la lista ***path*** en la cuadrícula cuyos costes están indicados en el vector ***v***.

El objetivo final es definir una función
min_weight_path : int array array -> int * int -> int * int -> (int * int) list,
de modo que ***min_weight_path v ini fin***, sea una lista de casillas que represente un camino válido de la casilla ***ini*** a la casilla ***fin,*** en la cuadrícula cuyos costes están indicados por el vector ***v***, tal que su **longitud** (número de casillas pisadas) sea **mínima** dentro de los que tienen **coste mínimo** (llamaremos "caminos minimales" a tales caminos).

Por ejemplo, si consideramos la siguiente cuadrícula

6	*2*	*1*	*1*	*6*
6	*5*	*1*	*2*	*1*
1	*2*	*3*	*1*	*1*
1	*8*	*6*	*5*	*1*

los costes de las casillas estarían indicados por el vector

```
[|[|6; 2; 1; 1; 6|]; [|6; 5; 1; 2; 1|]; [|1; 2; 3; 1; 1|]; [|1; 8; 6; 5; 1|]|]
```

El camino minimal desde la casilla *(2,3)* a la *(3,2)* sería

[(2, 3); (2, 2); (3, 2)] que tiene coste *10*

y un camino minimal desde la casilla *(0,0)* a la *(3,4)* podría ser

[(0, 0); (0, 1); (0, 2); (1, 2); (1, 3); (2, 3); (2, 4); (3, 4)]

que tiene coste *15.*

```
# let v = [|[|6; 2; 1; 1; 6|]; [|6; 5; 1; 2; 1|]; [|1; 2; 3; 1; 1|];
    [|1; 8; 6; 5; 1|]|];;
val v : int array array =
  [|[|6; 2; 1; 1; 6|]; [|6; 5; 1; 2; 1|]; [|1; 2; 3; 1; 1|];
    [|1; 8; 6; 5; 1|]|]

# min_weight_path v (2, 3) (2, 3);;
- : (int * int) list = [(2, 3)]

# min_weight_path v (2, 3) (3, 3);;
- : (int * int) list = [(2, 3); (3, 3)]

# min_weight_path v (2, 3) (3, 2);;
- : (int * int) list = [(2, 3); (2, 2); (3, 2)]

# let p = min_weight_path v (0, 0) (3, 4);;
val p : (int * int) list =
  [(0, 0); (0, 1); (0, 2); (1, 2); (1, 3); (2, 3); (2, 4); (3, 4)]

# valid_path 4 5 (0, 0) (3, 4) p;;
- : bool = true

# path_weight v p;;
- : int = 15

# let v = [|[| 1;  1;  1;  1; 10|]; [| 1;  4;  6;  8; 10|]; [| 1;  1;  1;  1;  1|];
            [|10;  8;  6;  4;  1|]; [| 1;  1;  1;  1;  1|]|];;
val v : int array array =
  [|[|1; 1; 1; 1; 10|]; [|1; 4; 6; 8; 10|]; [|1; 1; 1; 1; 1|];
    [|10; 8; 6; 4; 1|]; [|1; 1; 1; 1; 1|]|]

# let p = min_weight_path v (0, 4) (4,0);;
val p : (int * int) list =
  [(0, 4); (0, 3); (0, 2); (0, 1); (0, 0); (1, 0); (2, 0); (2, 1); (2, 2);
   (2, 3); (2, 4); (3, 4); (4, 4); (4, 3); (4, 2); (4, 1); (4, 0)]

# path_weight v p;;
- : int = 26
```

```
# let v = [|[|1; 1; 1; 10|]; [|1; 3; 5; 7|]; [|1; 1; 1; 10|]|];;
val v : int array array =
  [|[|1; 1; 1; 10|]; [|1; 3; 5; 7|]; [|1; 1; 1; 10|]|]

# let p = min_weight_path v (0,3) (2,0);;
val p : (int * int) list = [(0, 3); (0, 2); (0, 1); (0, 0); (1, 0); (2, 0)]

# path_weight v p;;
- : int = 15
```

La función ***min_weight_path*** debería provocar la excepción ***Invalid_argument "min_weight_path"*** si el vector de costes está vacío o contiene valores no estrictamente positivos, y en el caso de que las casillas inicial o final no correspondan a coordenadas válidas para ese vector.

Todas las funciones a implementar en este ejercicio deben mostrar un "comportamiento funcional"; en particular, las funciones ***path_weight*** y ***min_weight_path*** no deben alterar en modo alguno el valor del vector que reciben como argumento ni el de ninguna otra variable externa.

Las tres definiciones deben incluirse en un archivo ***"min_weight.ml"*** que **debe compilar sin errores con la orden**

`ocamlc -c min_weight.mli min_weight.ml`

Instrucciones adicionales

Esta prueba tiene carácter **individual** y **competitivo**; sólo serán consideradas para su evaluación las **30 primeras implementaciones correctas** recibidas. Sólo se deberá entregar el archivo ***"min_weight.ml"*** (la entrega debe confirmarse y sólo se puede realizar una vez). El código deberá venir convenientemente **comentado** para su fácil comprensión. Sólo se admitirán propuestas **originales**. En caso de detectarse algún **plagio**, los involucrados serán calificados con *0* puntos tanto en el apartado de *Pruebas Prácticas* como en el de *Prácticas de Laboratorio*. Es vital, por tanto, mantener la discreción sobre el desarrollo que se esté realizando, al menos hasta que concluya el plazo de presentación.

Se valorará, después de la **corrección** de la implementación, la **sencillez** y **claridad** del código y su **eficiencia**[1]. Las 30 primeras implementaciones correctas podrán ser valoradas con hasta **1.6 puntos** dentro del apartado *Pruebas Prácticas* (que admite hasta 2 puntos máximo).

La entrega se realizará a través de la tarea habilitada específicamente para este efecto en el Moodle de la asignatura, a partir de las 20:00 horas del día 28 de noviembre y siempre antes de las 20:00 del día 11 de diciembre. La tarea de entrega se cerrará tan pronto se detecte la recepción 30 entregas correctas (o, si no, a las 20:00 del día 11 de diciembre).

---

[1] Como referencia piense que, con una implementación estándar, la función *min_weight_path* podría tardar del orden de una decena de segundos en encontrar la solución en una cuadrícula 1000 x 1000, ejecutando bytecode en una máquina *estándar* actual.