LAB

Algoritmos y Programación

Semana 14: Programación con Restricciones

Restricción global Cumulative

```
predicate cumulative(array [int] of var int: s,
                     array [int] of var int: d,
                     array [int] of var int: r,
                     var int: b)
```

Requires that a set of tasks given by start times `s` , durations `d` , and resource requirements `r` , never require more than a global resource bound `b` at any one time.

https://www.minizinc.org/doc-2.6.4/en/predicates.html#cumulative

VPL-1: Planificación de proyecto

Una empresa tiene que desarrollar 4 aplicaciones.

Para ello dispone de 5 programadores, y suponemos que cada programador se dedica a una sola aplicación en cada momento.

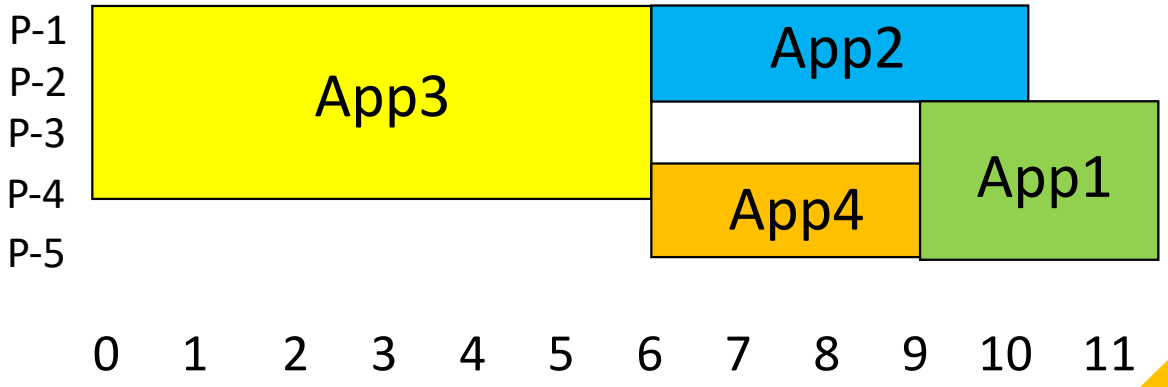
Cada aplicación tarda un tiempo determinado en realizarse y requiere una cantidad prefijada de programadores.

Tenemos que minimizar el tiempo que se requiere para acabar las 4 aplicaciones.

```
include "cumulative.mzn";

int:n=4; % total programas

int:k=5; % total de programadores

int:max_tiempo = 100; % limite superior de tiempo

array [1..n] of int: tiempo = [2,4,6,3];
array [1..n] of int: prog =   [3,2,4,2];

%%% variables de decisión
array [1..n] of var 0..max_tiempo: comienzo;
```

Total: 11
9 6 0 6

==========



Restricción global Circuit

Documentación de minizinc:

circuit

circuit[array[int] of var int: x)
Constraints the elements of x to define a circuit where x[i] = j mean that j is the successor of i.

Por ejemplo, si nos devuelve: [3, 7, 2, …]

representa este recorrido: 1→ 3 → 2 → 7 → …

VPL-2: TSP con trayecto parcial mínimo

- Dado un problema de TSP (circuito hamiltoniano), a diferencia del problema clásico donde el objetivo es minimizar la distancia total del recorrido, queremos en este caso que el objetivo sea **minimizar** la distancia mayor entre cada par de ciudades, es decir, queremos minimizar la distancia del máximo trayecto parcial.

Parámetros

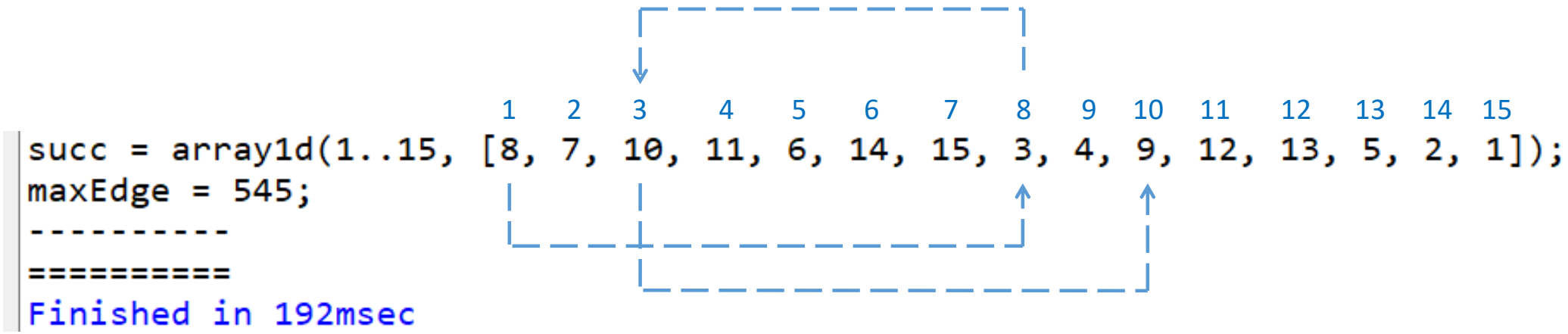
```
1 include "globals.mzn";
2 int: numCities;       %número de ciudades
3 set of int: City = 1..numCities;
4 int: maxAllowedEdge;     % máxima distancia permitida de un trayecto del recorrido
5
6 % distancia entre ciudades
7 % -1 significa que no hay conexión directa
8 array[City,City] of int: distance;
```

Ejemplo

-1 => no hay camino entre las ciudades

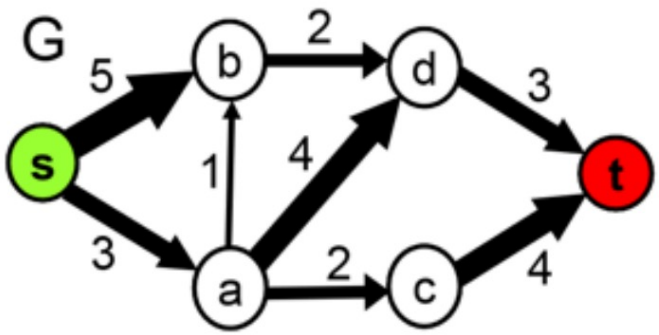
```
maxAllowedEdge = 600;
numCities = 15;
distance = [|0,-1,250,-1,-1,473,-1,172,-1,372,360,414,-1,-1,243
|-1,0,-1,-1,99,161,284,-1,-1,-1,446,431,478,262,457
|250,-1,0,-1,-1,-1,573,408,697,159,281,332,-1,-1,219
|-1,-1,-1,0,-1,-1,296,-1,400,481,392,338,172,449,-1
|-1,99,-1,-1,0,247,196,-1,-1,-1,410,384,380,181,446
|473,161,-1,-1,247,0,391,519,-1,-1,448,453,-1,422,416
|-1,284,573,296,196,391,0,-1,670,480,293,247,224,233,375
|172,-1,408,-1,-1,519,-1,0,-1,542,524,576,-1,-1,407
|-1,-1,697,400,-1,-1,670,-1,0,545,605,577,567,-1,-1
|372,-1,159,481,-1,-1,480,542,545,0,201,234,-1,-1,216
|360,446,281,392,410,448,293,524,605,201,0,56,445,-1,118
|414,431,332,338,384,453,247,576,577,234,56,0,390,478,171
|-1,478,-1,172,380,-1,224,-1,567,-1,445,390,0,295,-1
|-1,262,-1,449,181,422,233,-1,-1,-1,-1,478,295,0,-1
|243,457,219,-1,446,416,375,407,-1,216,118,171,-1,-1,0|];
```

Resultado esperado para el ejemplo



VPL-3

- Dado un grafo dirigido donde existen dos nodos especiales, uno llamado **fuente (s),** y otro llamado **sumidero (t)**.
  A cada arista se le asocia cierta **capacidad** positiva.



- Se plantean como restricciones:
- a) que para cada nodo del grafo, excepto el nodo fuente y el nodo sumidero, la suma de flujos entrantes a un nodo debe ser igual a la suma de flujos que salen de él.
- b) el flujo tanto de entrada como de salida no puede superar la capacidad de la arista correspondiente.

VPL-3

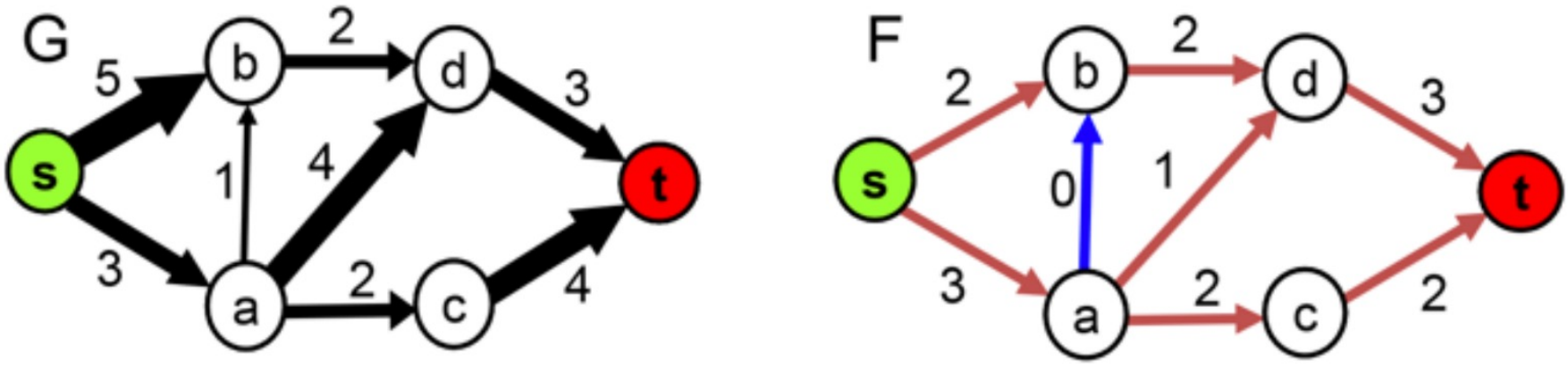
Características principales

- El flujo va a ser siempre positivo y con unidades enteras.
- El flujo que entra en un vértice es igual al que sale.
- El flujo que atraviesa una arista nunca será mayor que la capacidad; sólo puede ser menor o igual que ella.

- Objetivo: Maximizar la cantidad de flujo que llega al nodo sumidero, t.

VPL-3

- **Solución. G:** grafo del problema. **F:** grafo resultante.



En este caso el flujo máximo es 5 (lo que llega al nodo sumidero, t).

VPL 3

maxflow.mzn

```
1  int: num_nodes;                          % número de nodos del grafo
2  int: num_edges;                          % número de aristas del grafo
3
4  1..num_nodes:source;                     % nodo fuente
5  1..num_nodes:sink;                       % nodo sumidero
6
7  array[1..num_edges, 1..2] of int: edges; % aristas del grafo (origen->destino)
8  array[1..num_edges] of int: capacity;    % capacidad de cada arista
9
10 var int: max_flow;                       % variable de decision para el valor
11                                          % del flujo máximo
12 array[1..num_edges] of var int: flow;    % variable de decisión para el flujo
13                                          % final de cada arista
14
15 output
16 [
17   "max flow =" ++ show(max_flow)
18 ];
19
20 % Escribir el código a partir de aquí----------------------------------------
```