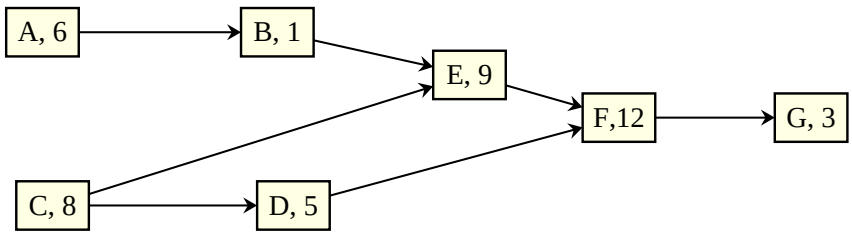
Planificación X97393_es

Un proyecto consiste en una serie de *tareas* o *actividades* a realizar y de restricciones o dependencias en el orden en que las tareas se pueden llevar a cabo. En su forma más elemental, una *dependencia* o *prerrequisito* simplemente establece que una determinada tarea A no puede comenzar a realizarse hasta que otra determinada tarea B no se haya completado. Se asumirá que existen todos los recursos necesarios para poder realizar todas las tareas simultáneamente si las dependencias lo permiten.
Veamos un ejemplo simple en el que tenemos un proyecto de manufacturación en el que hay 7 tareas. Para cada tarea tenemos un identificador, una descripción, una duración (p.e. en días) y las dependencias. Todo ello se recoge en la Tabla 1.

Identificador	Descripcion	Duracion	Prerrequisitos
A	Dise˜no preliminar	6	-
B	Evaluacion del dise˜no	1	A
C	Negociacion del contrato	8	-
D	Preparacion de la planta de fabricacion	5	C
E	Dise˜no final	9	B,C
F	Fabricacion del producto	12	D,E
G	Distribucion del producto	3	F

Table 1: Ejemplo de proyecto

Gráficamente podemos representar un proyecto mediante un diagrama en el que cada tarea se representa mediante un nodo con el correspondiente identificador y su duración, y hay una flecha entre dos tareas si existe una relación de dependencia entre ellas. La Figura 1 muestra el diagrama correspondiente al ejemplo anterior.



Para nuestro ejemplo anterior la entrada (ver la descripción del formato de entrada más abajo) podría ser:

```
7
A 6 B @
B 1 E @
C 8 D E @
D 5 F @
E 9 F @
F 12 G @
G 3 @
```

El algoritmo de planificación tiene que determinar el tiempo mínimo necesario para completar el proyecto. En el ejemplo, las tareas A y C pueden comenzar a realizarse en paralelo el día 0. El día 6 (a contar desde el inicio del proyecto) puede empezar la tarea B, mientras que C todavía no se ha completado. El día 7 podemos tener acabada B, pero E todavía no puede

empezar puesto que C no ha terminado. El día 8 pueden empezar D y E. Si empezamos D el día 8 habremos terminado el día 13, pero de todos modos F no puede empezar hasta el día 17, para cuando haya terminado E. Doce días después, el día 29, podemos comenzar G y el proyecto estará acabado el día 32. Es imposible completarlo en menos tiempo, a no ser que redujesemos las duraciones de algunas tareas.

El ejemplo anterior también nos permite ilustrar otro concepto importante: el de *camino crítico* (*critical path*). Por ejemplo, si la tarea A hubiese comenzado el día 1 en vez del día 0 la duración total del proyecto seguiría siendo de 32 días, puesto que la tarea E puede comenzar a realizarse el día 8. Más importante es el margen que tenemos para el comienzo de la tarea D; puede posponerse su comienzo hasta el día 12 sin que se vea afectada la duración total del proyecto. Por contra cualquier retraso en el comienzo de las tareas C (día 0), E (día 8), F (día 17) o G (día 29) supondrá un retraso en la finalización del proyecto. Se dice que C-E-F-G constituye un camino crítico en el proyecto. Pueden existir varios caminos críticos en un proyecto (en el ejemplo sólo hay uno) y su identificación es uno de los objetivos básicos de cualquier algoritmo de planificación.

A fin de poder formalizar adecuadamente los conceptos de duración total del proyecto, caminos críticos, etc. lo que se hace es introducir dos tareas, digamos `START` y `END`, de modo que ambas tienen duración 0. Todas las tareas del proyecto que no dependen de ninguna otra tarea se hace que dependan de `START`, y `END` dependerá de todas las tareas de las cuales no depende ninguna otra. En el ejemplo que hemos venido utilizando habría dependencias entre `START` y las tareas A y C; y la tarea `END` dependería de G. Puesto que las duraciones de estas tareas ficticias es 0, no se distorsionan las características que nos interesan en la planificación.

Dada una tarea i, el algoritmo de planificación debe hallar los siguientes valores:

- *Earliest Start time*, $ES(v)$: el instante de tiempo más temprano en el que se puede iniciar la tarea v;
- *Earliest Finish time*, $EF(v)$: el instante de tiempo más temprano en el que se puede finalizar la tarea v;
- *Latest Start time*, $LS(v)$: el instante de tiempo más tardío en el que se puede iniciar la tarea v sin retrasar el proyecto;
- *Latest Finish time*, $LF(v)$: el instante de tiempo más tardío en el que se puede finalizar la tarea v sin retrasar el proyecto.

Las relaciones entre ES y EF y entre LS y LF son relativamente sencillas de calcular y bastará con obtener una de cada; además resulta obvio que la duración total de un proyecto será igual a $ES(\texttt{END}) = EF(\texttt{END}) = LS(\texttt{END}) = LF(\texttt{END})$. Una vez calculados estos parámetros también es muy fácil e intuitivo identificar las tareas que pertenecen a algún camino crítico. Un ejemplo completo (el formato de la salida se describe más abajo):

```
Entrada          |   Salida
                 |
9                |   START 0     0     0     0     *
A 2 C D F @      |   A     0     2     0     2     *
B 5 E @          |   C     2     6     2     6     *
C 4 F @          |   D     2     5     7     10
D 3 G H @        |   F     6     12    6     12    *
E 5 G H @        |   B     0     5     0     5     *
F 6 I @          |   E     5     10    5     10    *
```

```
G 2 I @        |   G     10    12    10    12    *
H 4 @          |   H     10    14    13    17
I 5 @          |   I     12    17    12    17    *
               |   END   17    17    17    17    *
```

Entrada

La entrada del programa que tenéis que implementar consistirá en una serie de líneas. En primer lugar vendrá el número de tareas n. A continuación viene una línea por cada tarea que contendrá el identificador (un string), su duración (un entero positivo menor o igual a 1000) y la lista (separada por blancos) de las tareas de las que es prerrequisito; la lista termina con el string `@`. Los siguientes strings son reservados y no se utilizarán para identificar tareas: `START`, `END`, `@`.

Salida

La salida del programa consistirá en el string `Proyecto contiene ciclos` si el grafo no es acíclico; si el grafo es acíclico imprimirá una tabla donde para cada tarea, incluídas `START` y `END`, se escribe una línea con el identificador, el ES, el EF, el LS y el LF de la tarea. Cada columna tendrá una anchura de 6 caracteres. Además se escribirá un asterisco si la tarea forma parte de un camino crítico. La tarea `START` aparecerá en primer lugar. La tarea `END` en último lugar. Todas las restantes tareas deben listarse en el mismo orden que aparecen **mencionados** en la entrada.

Observación

Vuestro programa deberá estar escrito en C++. Para la evaluación de la práctica se usarán juegos de pruebas públicos y privados, pero el *Jutge* sólo tendrá en cuenta los juegos de pruebas públicos para determinar si el programa pasa o no, y los resultados de los juegos de pruebas privados sólo se usarán para perfilar la nota de la práctica.
En el fichero entregado, el algoritmo deberá estar debidamente comentado. En particular debéis justificar su corrección y su eficiencia en función de n (número de vértices/tareas) y m (número de arcos/dependencias).
La fecha y hora límite de entrega se publicará en un aviso del Racó.

Ejemplo de entrada 1

```
7
A 6 B @
B 1 E @
C 8 D E @
D 5 F @
E 9 F @
F 12 G @
G 3 @
```

Ejemplo de salida 1

```
START 0     0     0     0     *
A     0     6     1     7
B     6     7     7     8
E     8     17    8     17    *
C     0     8     0     8     *
D     8     13    12    17
F     17    29    17    29    *
G     29    32    29    32    *
END   32    32    32    32    *
```

Ejemplo de entrada 2

```
7
A 6 B  @
B 1 E C @
C 8 D A @
D 5 F @
E 9 F @
F 12 G @
G 3 @
```

Ejemplo de salida 2

```
Proyecto contiene ciclos
```

Ejemplo de entrada 3

```
9
A 2 C D F @
B 5 E @
C 4 F @
D 3 G H @
E 5 G H @
F 6 I @
G 2 I @
H 4 @
I 5 @
```

Ejemplo de salida 3

```
START 0     0     0     0     *
A     0     2     0     2     *
C     2     6     2     6     *
D     2     5     7     10
F     6     12    6     12    *
B     0     5     0     5     *
E     5     10    5     10    *
G     10    12    10    12    *
H     10    14    13    17
I     12    17    12    17    *
END   17    17    17    17    *
```

Información del problema

Autor : Conrado Martinez
Generación : 2013-03-06 00:22:13