Rock concert planning

Your new-found optimization skills may not help you become a rock star, but they may help you `plan` for rock concerts. This will surely increase your appeal to members of the opposite sex even more than becoming a rock star. Below is a set of tasks required to be performed before a rock concert. Also included (in GAMS format) is a set of precedence relationships between tasks and the number of days required to perform each task.

The promoters of a rock concert in Madison must perform the tasks shown before the concert can be held:

In [7]:
```
# Load the gams extension
%load_ext gams.magic
m = gams.exchange_container
```

```
The gams.magic extension is already loaded. To reload it, use:
  %reload_ext gams.magic
```

In [8]:
```
%%gams
set activity /
        A "Find Site",
        B "Find Engineers",
        C "Hire Opening Act",
        D "Set Radio and TV Ads",
        E "Set Up Ticket Agents",
        F "Prepare Electronics",
        G "Print Advertising",
        H "Set up Transportation",
        I "Rehearsals",
        J "Last-Minute Details"
/;

alias (activity,ai,aj);

set pred(ai,aj) "ai preceeds aj" /
        A. (B,C,E)
        B . F
        C . (D,G,H)
        (F,H) . I
        I . J
/;

parameter duration(activity) "in days" /
        A       3,      B       2
        C       6,      D       2
        E       3,      F       3
        G       5,      H       1
        I       1.5,    J       2
/;
```

```
--- Warning: The GAMS version (44.3.0) differs from the API version (45.1.0).
--- Warning: The GAMS version (44.3.0) differs from the API version (45.1.0).
```

If any increase in the duration of an activity increases the time needed to complete the project, then the activity is called a critical activity. We will find the critical activities in this project in two different ways.

Method 1

Set up a linear program to find the project duration (that is the minimum number of days needed to prepare for the concert). Determine the critical tasks by inspecting the multipliers of this problem! Make sure you store and print out the critical tasks.

In [9]:
```
%%gams
variables projdur;
positive variable t(activity) "time activity starts";

equations incidence(ai,aj), endtime(activity);

incidence(ai,aj)$pred(ai,aj)..
    t(aj) =g= t(ai) + duration(ai);

endtime(activity)..
    projdur =g= t(activity) + duration(activity);

model cpm /incidence,endTime/;

solve cpm using lp minimizing projdur;

set critical(activity) "critical activities";

critical(activity) = yes$(smax(aj$pred(aj,activity),incidence.m(aj,activity)) ge 1 o

display critical;
```

```
--- Warning: The GAMS version (44.3.0) differs from the API version (45.1.0).
--- Warning: The GAMS version (44.3.0) differs from the API version (45.1.0).
```

Out[9]:

	Solver Status	Model Status	Objective	#equ	#var	Model Type	Solver	Solver Time
0	Normal (1)	Optimal Global (1)	14.0	20	11	LP	CPLEX	0

In [10]:
```
print('critical arcs: ', *m.data['critical'].toList())
```

```
critical arcs:  A C G
```

Method 2

Now use two other linear programs to find the early event times (the earliest possible starting time for an activity) and the late event times (the latest possible time to start the activity without delaying project completion time). Then construct the critical activities by seeing which early event times are the same as the corresponding late event times. Make sure your print out includes a listing of the early and late event times, and the critical activities.

In [11]:
```
%%gams
parameter
    eetime(activity) "early event time",
    letime(activity) "late event time";

projdur.fx = projdur.l;

variables obj;
equations timeopt;

timeopt..
```

```
    obj =e= sum(activity,t(activity));

model eventtimes /timeopt,incidence,endtime/;

solve eventtimes using lp maximizing obj;

letime(activity) = t.l(activity);

solve eventtimes using lp minimizing obj;

eetime(activity) = t.l(activity);

critical(activity) = yes$(eetime(activity) ge letime(activity));

display eetime,letime,critical;
```

```
--- Warning: The GAMS version (44.3.0) differs from the API version (45.1.0).
--- Warning: The GAMS version (44.3.0) differs from the API version (45.1.0).
```

Out[11]:

	Solver Status	Model Status	Objective	#equ	#var	Model Type	Solver	Solver Time
0	Normal (1)	Optimal Global (1)	80.0	21	12	LP	CPLEX	0
1	Normal (1)	Optimal Global (1)	62.5	21	12	LP	CPLEX	0

In [12]:
```
print('critical arcs: ', *m.data['critical'].toList())
print('early event times: ', *m.data['eetime'].toList(), sep='\n')
print('late event times: ', *m.data['letime'].toList(), sep='\n')
```

```
critical arcs:  A C G
early event times:
('B', 3.0)
('C', 3.0)
('D', 9.0)
('E', 3.0)
('F', 5.0)
('G', 9.0)
('H', 9.0)
('I', 10.0)
('J', 11.5)
late event times:
('B', 5.5)
('C', 3.0)
('D', 12.0)
('E', 11.0)
('F', 7.5)
('G', 9.0)
('H', 9.5)
('I', 10.5)
('J', 12.0)
```

Which of these two methods is more reliable?

Type answer here!

In [13]:
```
%gams_cleanup --closedown
```

Method 1 identifies the critical tasks by optimizing the project duration and examining the corresponding dual variables. This approach is mathematically precise and gives a direct measure of the project's shortest duration. However, for very large problems, considering all constraints directly might lead to computational inefficiencies.

Method 2 identifies the critical tasks by computing the earliest start times and the latest start times for each task. This approach is generally more intuitive, especially for those unfamiliar with linear programming. This method is also scalable for large problems as it breaks down the problem into two smaller ones.

If one is looking for a quick and intuitive way to identify critical tasks and less concerned about the exact shortest duration of the project, method 2 might be more suitable.If you want to optimize the project's duration and gather related dual information, then method 1 might be the better choice.

```
In [ ]:
```