PERT/CPM Project

First Semester 2015/2016

The **Project Evaluation and Review Technique (PERT)** is a tool used in general project management and was designed to analyse the involved tasks in completing a given project, especially the time needed to complete each task, and to identify the minimum time needed to complete the total project. It is commonly used in addition to the **Critical Path Method (CPM)**, which is an algorithm for scheduling a set of project activities.

The goal of this work is to specify and model an application for business management of project activities, as well as to provide a particular set of typical features of these applications.

Consider that the projects involve two types of activity - **fixed and variable cost activities**, dependent on the resources and time involved in its implementation. Thus, the information necessary to associate with each activity comprises the key activity identification, type of activity, description, duration, time unit (hour, day, week, month ...) cost of the activity (for fixed cost activities), and cost/time and total time, (for variable cost activities) and activities that immediately precede it.

Taking advantage of the facilities offered by the Object Oriented paradigm, the project should be done incrementally starting with the simplest classes until reaching the final system. Given that it is important to test the different classes as they are developed, the following steps should be followed in order.

First Part

1. Develop the required classes that allow you to manage the different activities allocated to a project.

2. Develop a method to insert the activities of a project from a text file in a data structure of your choice. The information in the file is structured as follows:

```
ActivKey,ActivType,descr,duration,duration-unit,tot-cost
ActivKey,ActivType,descr,duration,duration-unit,cost/time,total-time,actkeyprev1
ActivKey,ActivType,descr,duration,duration-unit,tot-cost,actkeyprev1,actkeyprev2
ActivKey,ActivType,descr,duration,duration-unit,tot-cost
…
ActivKey,ActivType,descr,duration,duration-unit,cost/time,totalime,actkeyprev1
ActivKey,ActivType,descr,duration,duration-unit,tot-cost,actkeyprev1,actkeyprev2,actkeyprc3
```

There are no spaces.

3. Present the temporal complexity (Big-Oh notation) of the main methods and data structures developed.

Part 1 Data Delivery: 10 Oct. 2015

PERT/CPM Project

Second Part

One of the applications of graphs is the management of the activities of a project through the PERT/CPM technique.

A PERT graph:

– has one source node, where the project begins

– has one finish node, where the project ends

– does not have cycles

– the vertices represent the activities and the branches the precedence between the activities - **activity-on-node** (AON) graph

The starting and finishing times of each activity, if no delays occur anywhere in the project, are called the earliest start time and the earliest finish time of the activity. These times are represented by the symbols:

– **Earliest Start (ES) time**
– **Earliest Finish (EF) time**

Where

EF = ES + (estimated) duration of the activity

The ES time of the source node (node without predecessors) has the value 0, and for all other nodes j with one or more predecessors, the ES_j is the maximum value of its k predecessors:

$$ES_j = \max (EF_K)$$

The ES and EF values are calculated making a **forward pass** through the graph.

The latest start time for an activity is the latest possible time that it can start without delaying the completion of the project, assuming no subsequent delays in the project. The latest finish time has the corresponding definition with respect to finishing the activity.

– **Latest Start (LS) time**
– **Latest Finish (LF) time**

Where

LS = LF - (estimated) duration of the activity

The LF of the final node (node where the project ends) is equal to its EF (LF = EF) and LS = EF (duration of the activity=0) . For all other nodes, if the event already has one or more predecessors, its LF_j will be the minimum value:

$$LF_j = \min (LS_K)$$

Where all immediate predecessors k of node j (in a **reverse traverse**) must be considered.

The **slack** of an activity is the maximum delay that activity can start with respect to the previous one, without causing delays to the project. The slack for an activity is the difference between its latest finish time and its earliest finish time. In symbols,

$$Slack_i = LF_i – EF_i$$

The nodes where LF = EF are **critical nodes**, because they do not have any slack. Each activity with zero slack is on the **critical path** through the project network such that any delay along this path will delay the completion of the project.

For small project networks, finding all the paths and determining the longest path(s) is a convenient way to identify the critical path(s). However, this is not an efficient procedure for larger projects. PERT/CPM uses a considerably more efficient procedure instead, which also provides much more information than is available from finding all the paths.

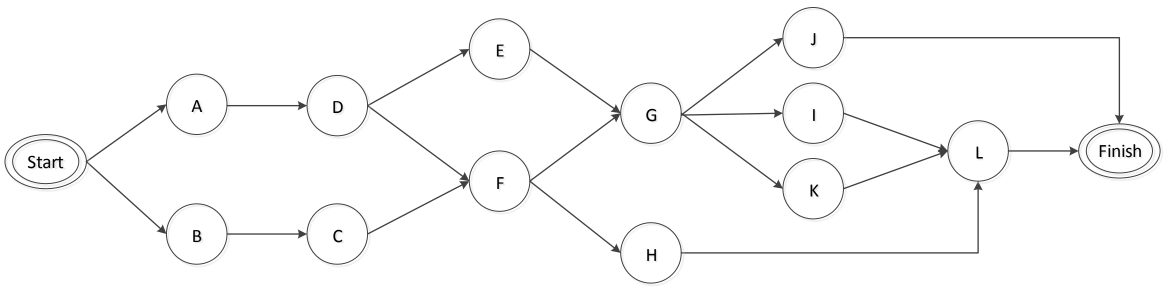
As an example consider a project to develop a software application with the following activities:

Activity	Description	Cost	Duration (weeks)	Immediate Predecessors
A	High level analysis	30 €/112h	1	_
B	Order Hardware platform	2500 €	4	_
C	Installation and commissioning of hardware	1250 €	2	B
D	Detailed analysis of core modules	30 €/162h	3	A
E	Detailed analysis of supporting modules	30 €/108h	2	D
F	Programming of core modules	20 €/108h	4	C,D
G	Programming of supporting modules	20 €/54h	3	E,F
H	Quality assurance of core modules	30 €/54h	2	F
I	Quality assurance of supporting modules	30 €/27h	1	G
J	Application Manual	550 €	1	G
K	User Manual	750 €	1	G
L	Core and supporting module training	1500 €	2	H,I,K

PERT/CPM Project

First Semester 2015/2016

Graphical Representation of the Project with an AON graph



1. Create a PertCpm class that, in addition to the methods necessary for the proper functioning of the class, should include methods to:

i. Build one PERT/CPM graph from the objects previously loaded in the first part of this project, taking into account the necessary validations

ii. Return the activities by its order of completion

iii. Return all paths between the Start and Finish nodes, with its respective length

iv. For each project activity provide the following information:

Activity	Cost	Duration	Predecessors	ES	LS	EF	LF	Slack
A	VCA 3360 €	1	_	0	2	1	3	2
B	FCA 2500 €	4	_	0	0	4	4	0
C	FCA 1250 €	2	B	4	4	6	6	0
D	VCA 4860 €	3	A	1	3	4	6	2
E	VCA 3240 €	2	D	4	8	6	10	4
F	VCA 2160 €	4	C,D	6	6	10	10	0
G	VCA 1080 €	3	E,F	10	10	13	13	0
H	VCA 1620 €	2	F	10	12	12	14	2
I	VCA 810 €	1	G	13	13	14	14	0
J	FCA 550 €	1	G	13	15	14	16	2
K	FCA 750 €	1	G	13	13	14	14	0
L	FCA 1500 €	2	H,I,K	14	14	16	16	0

v. Return the **critical path(s)**, that is, the path that contains the critical activities.

Part 2 Data Delivery: 14 Nov. 2015

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

isep Instituto Superior de Engenharia do Porto

PERT/CPM Project

First Semester 2015/2016

Third Part

In this last part of the project you will develop an application to manage several projects and its activities. For that you will use a binary search tree to store the reference of the project, project type, its completion time and delay time. Another tree will be used to store the activities of the projects (reference of the project, code activity, type activity, duration time, delay time).

1. Develop a method to load all the information of several projects and its activities from a text file to the binary search trees.
2. Print the projects and its activities by order of project delay time.
3. For two projects with delay, return the late activities with the same type.

Part 3 Data Delivery: 5 Dec. 2015

Rules

- The work must be done in **groups of two students**.
- The project report must be prepared according to the report template available in Moodle and shall contain the elements ordered as enunciated above: class diagram, sequence diagram of the specific methods of classes, including the analysis of complexity time.
- The project will be evaluated in accordance with the features implemented, modularity, organization, clarity and code efficiency.
- Each part of the work must be submitted to the Moodle, in the area of the subject until **24:00 on the stipulated day**. The mark of the work will be penalized **10% per day late** and no projects will be accepted after **two days** of the dates indicated.
- Submitted parts must be corrected according to the teacher's comments/suggestions.
  The code project must be developed in Java and using the Git version control tool.
- The code project and the report should be placed in a ZIP file with the name: **PROFINITALSLABCLASS-NºStudent-NºStudent.zip**, for example **MFC2DM-1101350-1101460.zip**
- The presentation/evaluation of the final work will be individual, on a date fixed with the laboratory teacher class, during the week of **7-12 December**