A Fast Spreadsheet Implementation of the Critical Path Method

Ron Davis
Department of Marketing and Decision Sciences
College of Business
San José State University
One Washington Square
San José, CA 95193-0069
Tel: 408 924 3547
Fax: 408 924 3445
E-mail: davis_r@cob.sjsu.edu

April 2006

Keywords:

Ref #: I-06-003

A FAST SPREADSHEET IMPLEMENTATION of the CRITICAL PATH METHOD

By Dr. Ron Davis, San Jose State University
College of Business, One Washington Square, San Jose, California 95192
davis_r@cob.sjsu.edu; ron@mathproservices.com;

ABSTRACT

This is the first of two companion papers by the present author appearing in these transactions. This first paper deals with two efficiency enhancements to the spreadsheet critical path implementations of Kala C Seal (Seal, 2001) and Cliff T Ragsdale (Ragsdale, 2003) that are motivated by the need to simulate a project and re-compute the duration of its critical path many thousands of times to ascertain an accurate histogram of total project duration. The second paper deals with use of the beta distribution to carry out stochastic project duration and stochastic project crashing analyses.

Since project network structure is constant throughout the simulation process, efficiency considerations in this context beget a desire to compactly represent the precedence network structure in a form that can be conveniently used during a simulation without repeated re-computation. By judicious computation of pointer lists to compactly represent the predecessor and successor sets for each activity node prior to the simulation, and by use of the OFFSET function to gather relevant sets of early finish and late start times into compact arrays during the simulation, unnecessary re-computation of structural results and unnecessary circular references can be avoided.

In fact, the very same logic that generates the pointer lists can be used instead to compute the argument lists for the MAX and MIN function evaluations required during the forward and backward passes. This enables use of the pointer lists and OFFSET function calls to be omitted altogether, further speeding run times.

1 INTRODUCTION

Formal project management techniques were developed in the late 1950s and early 60s as part of the early missile and space exploration programs. Actually there were two parallel developments, one called the Critical Path Method (CPM, spearheaded by Dow Chemical Corporation) that featured a simple deterministic analysis, and the other called Program Evaluation and Review Technique (PERT, spearheaded by Lockheed Missiles and Space Corporation) that made a simplistic attempt to incorporate uncertainty into the project duration analysis. These methods were based on two closely related network diagramming conventions, called activity-on-node (AON) and activity-on-arc (AOA). Most textbook authors favored the AOA convention, whereas most programs were written using the AON convention. Since the diagramming logic is much simpler for AON in comparison with AOA, the AON approach is now rapidly becoming the standard convention in industrial and

government use of the methodology, and it is seen more frequently in newer textbooks as well.

The US government in general, and the Defense Department in particular, rapidly adopted the cpm/pert methodologies as a required part of their contracting procedures, since it enabled better comparative analysis of competing bids for contract assignments that they were funding. Consequently Operations Research Departments and MBA business curriculums at all major Anglo-American universities taught the methodologies because of their growing use and acceptance as a necessary part of doing business. As graduates of these universities became employees in government, industry and the world of nonprofits as well, the method has been diffused into all aspects of our society. Insight into the breadth of application of these techniques can be gained by a quick perusal of the Special Interest Groups (SIGs) listing of the Project Management Institute (see http://www.pmi.org/info/default.asp). The following table is a partial listing of the currently operating PMI SIGs that are devoted to the special aspects of CPM/PERT that applies in these diverse areas.

- Aerospace & Defense
- Automation Systems
- Automotive
- Configuration Management
- Consulting
- Design-Procurement-Construction
- Diversity
- eBusiness
- Education & Training
- Environmental Management
- Financial Services
- Government
- Hospitality Management
- Information Systems
- Information Technology & Telecommunications
- International Development
- Manufacturing
- Marketing & Sales
- Metrics
- New Product Development
- Oil, Gas & Petrochemical
- Pharmaceutical
- Program Management Office
- Quality in Project Management
- Retail
- Risk Management
- Service & Outsourcing
- Students of PM
- Troubled Projects
- Utility
- Women in Project Management

It is clear from the above listing that these techniques are today used in every area of society, from government agencies to non-profits, and from engineering companies to service industries.

Another organization/site that facilitates interaction between and collaboration amongst members of the Project Management community is the Project Management Forum (see http://www.pmforum.org/warindex.htm). On the home page it states that "The PMFORUM is a resource for information on international project management affairs. The PMFORUM supports the development, international cooperation,

promotion and advancement of a professional and worldwide project management discipline." A perusal of the site reveals not only information on professional standards, an archive of recent journal publications, and a directory of PM organizations around the world, but also provides a global marketplace of expert resources where vendors of software and related products and services have listings that benefit the product and service providers as well as the individuals and organizations that use them. Within this last area there are subheadings for information on Consulting Services, Information Technology, Careers, Services and Support, Software Applications, and Training and Development. A Vendors Search Engine makes it easy to search a Expert Resources Global Market Place Directory for just the type of help that may be needed. For keeping abreast of the latest developments in this field, the PM Forum maintains an online periodical called PM World Today that "contains the latest notices, reports, news and information related to project management from around the world. "

With the widespread acceptance of spreadsheets for quantitative analysis in business and government alike, it is quite natural, therefore, that spreadsheet implementations of the Critical Path Method and PERT would be forthcoming. The two methods upon which the present implementation is based appeared recently in this journal. They were presented by Kala C Seal (Seal, 2001) and Cliff T Ragsdale (Ragsdale, 2003). The implementation presented here incorporates elements from both of these approaches and, through the use of VBA procedures, forms a synthesis that is intended to be neater and cleaner (and faster in the simulation context) than the two preceding implementations. Scalability and portability are also thereby enhanced.

2 A NETWORK MODELING CONVENTION

We will use the example presented in Ragsdale's paper (Ragsdale, 2003) for the discussion here. The first difference in our treatment occurs immediately in the diagramming of the project network. For pedagogical reasons, whenever there are multiple beginning nodes in a project (A and B in this case) a BOP "Beginning of Project" milestone node is inserted before them to provide a unique starting node for the project. Likewise, when there are multiple ending nodes in a project (I and J in this case) an EOP "End of Project" milestone node is inserted at the end to provide a unique ending node for the project. It is understood that such milestone nodes have zero duration. By so doing there is no ambiguity about where the project begins or ends. The resulting network diagram for the example is shown below in Figure 1.

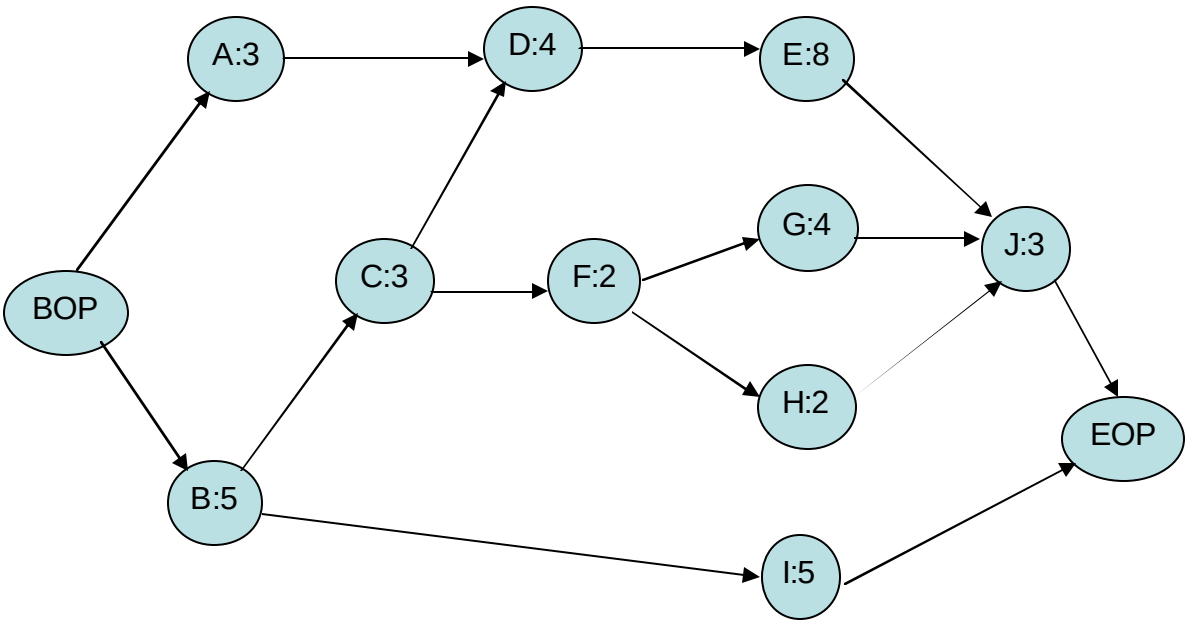

Figure 1: Project Network Diagram

For the spreadsheet implementation we shall discuss, it turns out that it is not necessary to actually insert a row in the project table for the BOP node. On the other hand, it is desirable to put in a row for the EOP at the end of the table. The reason for this is that the project is not done until all of the "dangling loose ends" at the end are completed. Hence there is a maximum value computation that must be done at the EOP in order to find out just how long the project is taking. Moreover, this time is a boundary condition at the end of the Late Finish column that starts the backward pass computations. Hence it MUST be known in order to do the backward pass, and therefore our position is that it MUST be shown in both the project network diagram as well as added as a final row in the project table. Of course there is sometimes no need for an EOP milestone addition, when there is already a unique final activity that is at the end of every path through the network, such as in the example in Seal's paper. But when there are "multiple loose ends" as in Ragsdale's example, we take the position that the EOP MUST be added, both to the network diagram and to the spreadsheet table for the project.

3. SOLUTION TABLE LAYOUT

In order to provide a frame of reference for the pointer list discussion it is convenient to jump ahead to the solved table layout at this point. This is shown in Table 1 below. The first four columns of this table are given data, the last six are computed columns. Formulas for the computed columns (all except one auto fill formulas) will be given shortly, however the present focus is on several structural differences from the table given by Ragsdale. The principal difference is the inclusion of the EOP row at the end where a maximum value is computed to determine the duration of the project. The formula is of the same form as the one just above it, but the forward pass is not complete until the maximum value computation in the EOP row is done. This value is

then used as the boundary condition of the backwards pass by means of a simple value replication formula at the end of the LF column. The minor difference is the addition of another column at the end which converts the slack values into the names of the critical activities using IF functions, with the full name of the critical path computed using the CONCATENATE function in the lower right corner of the table.

A	B	C	D	E	F	G	H	I	J
Activity	Description	Immediate Predecessor	Days	ES T	EF T	LS T	LF T	SLACK	CRITICAL
A	Select Site		3	0	3	5	8	5	
B	Create building plan		5	0	5	0	5	0	B
C	Determine labor needs	B	3	5	8	5	8	0	C
D	Design facility	A, C	4	8	12	8	12	0	D
E	Construct facility	D	8	12	20	12	20	0	E
F	Select personnel to move	C	2	8	10	14	16	6	
G	Hire new personnel	F	4	10	14	16	20	6	
H	Move records	F	2	10	12	18	20	8	
I	Arrange financing	B	5	5	10	18	23	13	
J	Train new personnel	E, G, H	3	20	23	20	23	0	J
K	End Of Project	I, J	0	23	23	23	23		BCDEJ

TABLE 1: CPM SOLUTION TABLE

4. CONSTRUCTION OF THE POINTER LISTS

From inspection of the project network diagram, it is clear that each node (other than the BOP) has a set of predecessors and each (other than the EOP) has a set of successors. By tabulating the pointers to predecessors in one list, and the pointers to successors in another list, we compactly identify and represent the precedence structure of the network that is needed during the forward and backward passes of the CPM. These lists are determined once and for all prior to any simulation studies that may be done based on the variation of activity times.

Ragsdale indicates an aversion to the binary precedence matrix presented by Seal, however the embedded functions that he proposes very neatly yield the very same binary table, as follows. Let "Precedence" be the name of column C in the above table (rows 2 through 12) that contains the information defining the structure of the project network. Then the array formula {=IF(ISERR(FIND("A",Precedence)),0,1)} yields a column vector of (11) zeros and ones where a 1 shows that activity A precedes the activity associated with the row in which the 1 occurs. Hence it shows the set of successors of activity A as a binary vector (this is the first column of what Seal calls the Precendence Matrix). By noting the number of 1s in this column, and where they

occur, we obtain the first row of the successor pointer list for the network, namely [1; 4] where 1 is the number of immediate successors of A, and 4 is the index associated with activity D, the activity immediately following A.

Moving forward now to activity B we find that the array formula {=IF(ISERR(FIND("B",Precedence)),0,1)} yields the binary column vector showing the immediate successors of activity B. It has two 1s in it, in index position 3 and in index position 9, so the second row of the successor pointer list table is [2; 3, 9]. Repeating this process for each activity in turn yields the entire Immediate Successors Matrix and the Immediate Successors Pointer list, as shown below.

Successors Matrix

A	B	C	D	E	F	G	H	I	J	EOP
0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
1	0	1	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	1	1	0	0	0
0	0	0	0	0	0	0	0	1	1	0

TABLE 2: SUCCESSORS MATRIX

NumSucc	Pointer	List
1	4	
2	3	9
2	4	6
1	5	
1	10	
2	7	8
1	10	
1	10	
1	11	
1	11	
0		

TABLE 3: SUCCESSORS POINTER LIST

The first column of the pointer list shows the number of successors for the activities, and the remaining numbers in each row give the index numbers for the succeeding activities. We have constructed the pointer list from the Successors Matrix by hand in this example, but in general this could be automated with a straightforward VBA macro. The label cell in the upper left corner is named NumSucc so that elements of

this Successors Pointer List can be referenced by offset from this cell during the backwards pass computations.

It turns out that the binary Predecessors Matrix is just the transpose of the binary Successors matrix (if activity "j" follows activity "i" then activity "i" precedes activity "j"). The Predecessors Pointer List is then formed by summarizing the number and location of the 1s in each column of the Predecessors Matrix. The results for this example are shown below.

Predecessors Matrix

A	B	C	D	E	F	G	H	I	J	EOP
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	0	0
0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	1	0
0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0

TABLE 4: PREDECESSORS MATRIX

NumPred	Pointer List			
0				
0				
1	2			
2	1	3		
1	4			
1	3			
1	6			
1	6			
1	2			
3	5	7	8	
2	9	10		

TABLE 5: PREDECESSORS POINTER LIST

The label cell in the upper left corner of this table is named NumPred so that the elements of the Predecessors Pointer List may be referenced by offset from this cell. Taken together, the two pointer lists effectively and compactly specify the project network structure based on the precedence relations indicated in column C of the data table. They are computed only one time, prior to any CPM computations and prior to any PERT simulations. We therefore turn to the question of how to use these lists in the context of the MAX and MIN functions associated with the forward and backward passes of the CPM.

5. USE OF THE OFFSET FUNCTION FOR THE ES & LF COMPUTATIONS

The ES times are based on the MAX of a set of preceding EF times, and hence make use of the Predecessors Pointer List. We name the upper left corner of this pointer list table NumPred so that elements of the table can be referred to by offsets from the cell containing the label NumPred. When there are no predecessors, we must return the value 0, and when there are predecessors, we must return the maximum of the indicated EF times. We want to be able to copy this formula down column E to obtain all of the ES times using the same functional form, so we create an index number column in column K that runs from 1 for activity A to 11 for "activity" EOP. By referring to this index number for each activity we are able to construct the ES formula for A as

```
{=IF(OFFSET(NumPred,K2,0)>0,
MAX(OFFSET($F$1,OFFSET(NumPred,K2,1,1,OFFSET(NumPred,K2,0)),0)),
0)}
```

We have formatted this for three lines in order to clarify the IF structure of the statement. First note that this formula is an array formula even though it returns only a single value. This is required since some of the intermediate computations are stated in terms of arrays, notably the argument of the MAX function. The logical expression OFFSET(NumPred,K2,0)>0 checks to see whether or not there are any predecessors for the activity associated with the index number showing in column K. If not, then the IF function returns the value 0, as desired. If the number of predecessors is positive, however, then the IF statement returns the MAX of set of EF values stipulated in the argument of the MAX function call.

The argument of the MAX function call is admittedly a bit "tricky" but on reflection one sees that it does exactly what is needed AND NOTHING MORE. The length of this argument vector is equal to the number of predecessors of the activity, not the total number of activities. Hence for this example, it never involves more than three elements, and usually the count is one or two. The syntax is a bit forbidding because of the nested OFFSET function calls, so to understand it better consider the following rewrite that does not actually run in Excel but clarifies the intent of the expression:

```
MAX(OFFSET($F$1,Array_of_predecessor_pointers,0))
```

The F1 reference is made because the EF times are in the cells from F2 to F12 and we want to use the offset function to refer to them by means of an offset from F1. Then Array_of_predecessor_pointers is just the predecessor pointer list for the present activity. It is specified by the term

```
OFFSET(NumPred,K2,1,1,OFFSET(NumPred,K2,0))
```

The first three arguments of this outer OFFSET call tell it to come down to the indicated row of the predecessors pointer list and move to the right one column to locate the first pointer for the indicated activity. The last two arguments give the dimensions of the

pointer list being referred to, which is 1 row by OFFSET(NumPred,K2,0) columns. Hence the MAX of the EF times for immediate predecessors of each activity is the result. Note that the MAX function is actually evaluated only when OFFSET(NumPred,Ki,0) is positive, so in this case it would not be evaluated when the argument is K2 or K3. But for K4 and above, the number of predecessors is positive so the array IF statement would return the results of the array MAX computation.

The formulas in column F are trivial, since they just sum the corresponding values in columns D and E. Hence, by selecting E2:F2 one can then drag the handle down to auto fill the two formulas down to E12:F12. Again, since the results in row 12 will be used to initialize the backwards pass in column G, we regard the inclusion of the EOP row as an essential feature of this implementation. The formulas for ES and EF of the EOP are of the same form as for the activities up to that point, one just takes the duration of the EOP to be zero.

Turning now to the backward pass, the LF at the EOP is set equal to the EF at the EOP by definition. Hence =F12 is the formula placed in G12. And the formula in H12 is just G12 – D12. Moving back up to row 11, the array formula for the LF in G11 is then

{=MIN(OFFSET(H1,OFFSET(NumSucc,K11,1,1,OFFSET(NumSucc,K11,0)),0))}

The IF statement is not needed in this case because prior to the EOP, EVERY activity has at least one successor so OFFSET(NumSucc,Ki,0) will always be positive for "i" less than 12 and greater than 1. (In fact, the IF statement could be eliminated during the forward pass as well by including a row for the BOP in the solution table). Once again, the reference to H1 is because the LS times are in column H, and the second argument of the first OFFSET function is the array of pointers in the Successors Pointer List for the activity indexed by K11. The formula in H11 is just G11 – D11, and by selecting G11:H11 the two formulas may be copied up to G2:H2 to complete the backward pass.

The slack computation in column I is obtained by the difference LS – ES or LF – EF and the critical activities are displayed in column J by use of the appropriate IF function based on whether the slack is zero or not. Concatenation of the critical activity letters is done at the bottom of column J using the CONCATENATE function.

6. A FURTHER ENHANCEMENT

To the purist, it may seem that use of OFFSET array functions in columns E and G for the ES and LF times should be eliminated at the outset by computing the MAX and MIN argument lists as text strings and then "setting" the appropriate formulas into those cells once and for all. In this case, the simulation iterations would have the benefit of simple MIN and MAX functions in columns E and G rather than the complex array formulas involving nested OFFSET functions, and should therefore run just that much faster. This can be done with a couple of VBA macros, as shown in the appendix to this paper. The result of running the ForwardPass and BackwardPass procedures for the present example is shown in the following table.

	E	F	G	H
1	**EST**	**EFT**	**LST**	**LFT**
2	=MAX(0)	=SUM(D2:E2)	=H2-D2	=MIN(G5)
3	=MAX(0)	=SUM(D3:E3)	=H3-D3	=MIN(G4,G10)
4	=MAX(F3)	=SUM(D4:E4)	=H4-D4	=MIN(G5,G7)
5	=MAX(F2,F4)	=SUM(D5:E5)	=H5-D5	=MIN(G6)
6	=MAX(F5)	=SUM(D6:E6)	=H6-D6	=MIN(G11)
7	=MAX(F4)	=SUM(D7:E7)	=H7-D7	=MIN(G8,G9)
8	=MAX(F7)	=SUM(D8:E8)	=H8-D8	=MIN(G11)
9	=MAX(F7)	=SUM(D9:E9)	=H9-D9	=MIN(G11)
10	=MAX(F3)	=SUM(D10:E10)	=H10-D10	=MIN(G12)
11	=MAX(F6,F8,F9)	=SUM(D11:E11)	=H11-D11	=MIN(G12)
12	=MAX(F10,F11)	=SUM(D12:E12)	=H12-D12	=F12

7. CONCLUSIONS

To recap, the first of two principal benefits of this implementation are that the precedence structure is embodied in the two pointer lists determined once and for all before the CPM or any PERT simulation trials (or in the computed MIN and MAX function calls based on the predecessor and successor relationships). Hence during simulation trials this structure need not be rediscovered on each and every simulation trial. Secondly, use of these pointer lists shortens the arrays fed to the MAX and MIN functions during the forward and backward passes since they are no longer as long as the number of activities in the project, but are only as long as the pointer lists in the successor and predecessor pointer lists. And if the argument lists are pre-computed using the VBA code as in the appendix, then the OFFSET function calls can be entirely eliminated as well. Hence the work required to carry out each CPM computation will tend to grow linearly with the number of activities in the project rather than quadratically. This enhances the scalability of the method.

In addition, we have avoided use of any intentional circular reference such as advocated by Ragsdale. When the EOP is incorporated into the computation, there is no inherent circularity in the logic of the underlying CPM formulas. Hence we regard the use of circular references in Excel as an artificial way of compensating for an incomplete network diagram for the project that should be avoided. Eliminating reliance on circular references speeds run time and enhances portability of the method.

The reader may well wonder why the discussion of pointer lists was retained after it was discovered that in the spreadsheet they need not be explicitly developed to get the desired results, since VBA allows formulas to be computed and saved. The reason is that programmers for websites providing online solutions may not have the luxury of an Excel spreadsheet running on the server. If the solution is to be obtained in some other programming language, then the pointer lists may be developed explicitly and form the basis for fast solutions on a server supporting a website.

BIBLIOGRAPHY

- Albright, S. Christian (2001), ***VBA for Modelers****: Developing Decision Support Systems with Microsoft Excel*, Duxbury, Pacific Grove, CA
- Davis, Ronald E. (2005), "Stochastic Project Duration Analysis using PERT-beta Distributions", INFORMS Transactions on Education 5:2 (TBD)
- Ragsdale, Cliff T. (2003), "A New Approach to Implementing Project Networks in Spreadsheets", INFORMS Transactions on Education 3:3 (76-85).
- Roman, Steven (2002), *Writing Excel Macros with VBA (2nd ed.)*, O'Reilly & Assoc., Inc., Sebastopol, CA
- Seal, Kala C. (2001), "A Generalized PERT/CPM Implementation in a Spreadsheet", INFORMS Transactions on Education 2:1 (16-26).

APPENDIX

The two VBA macros that create the desired MAX and MIN functions (that compute EST and LFT quantities respectively) utilize the InStr function in place of the spreadsheet function FIND. The offset property is used throughout to facilitate moving through the various lists involved. The user may need to edit the Set statements to be sure that they refer to the appropriate columns, depending on the layout of the model at hand. The coding assumes that the four time columns EST, EFT, LST, and LFT will be in consecutive columns and in this order. The sums in the EFT column and the subtractions in the LST column are entered manually, while the two macros fill in the formulas for the EST and LFT columns.

```
Sub ForwardPass()
Dim i, k, NumRows As Integer: Dim ArgString As String
With ActiveSheet
Set ActivityList = Range("A1"): Set PredecessorList = Range("C1")
Set EarlyStartList = Range("E1")
With ActivityList
  NumRows = Range(.Offset(1, 0), .Offset(1, 0).End(xlDown)).Rows.Count
End With
For k = 1 To NumRows
  ArgString = "0"
  If Len(PredecessorList.Offset(k, 0)) > 0 Then
    For i = 1 To k - 1
      If InStr(1, PredecessorList.Offset(k, 0), ActivityList.Offset(i, 0)) > 0 Then
        ArgString = ArgString & "," & EarlyStartList.Offset(i, 1).Address
      End If
    Next
  End If
  If Len(ArgString) > 2 Then
    ArgString = Right(ArgString, Len(ArgString) - 2)
  End If
  EarlyStartList.Offset(k, 0).Formula = "=MAX(" & ArgString & ")"
```

```
Next k
End With
End Sub
============================================================
Sub BackwardPass()
Dim i, k, n, NumRows As Integer: Dim ArgString As String
Set ActivityList = Range("A1"): Set PredecessorList = Range("C1")
Set EarlyStartList = Range("E1")
With ActiveSheet
With ActivityList
  NumRows = Range(.Offset(1, 0), .Offset(1, 0).End(xlDown)).Rows.Count
End With
EarlyStartList.Offset(NumRows, 3).Formula = "=" & EarlyStartList.Offset(NumRows,
1).Address
For n = 1 To NumRows - 1
   ArgString = ""
   k = NumRows - n
   For i = k + 1 To NumRows
      If InStr(1, PredecessorList.Offset(i, 0), ActivityList.Offset(k, 0)) > 0 Then
         ArgString = ArgString & EarlyStartList.Offset(i, 2).Address & ","
      End If
   Next i
   ArgString = Left(ArgString, Len(ArgString) - 1)
   EarlyStartList.Offset(k, 3).Formula = "=MIN(" & ArgString & ")"
Next n
End With
End Sub
============================================================
```