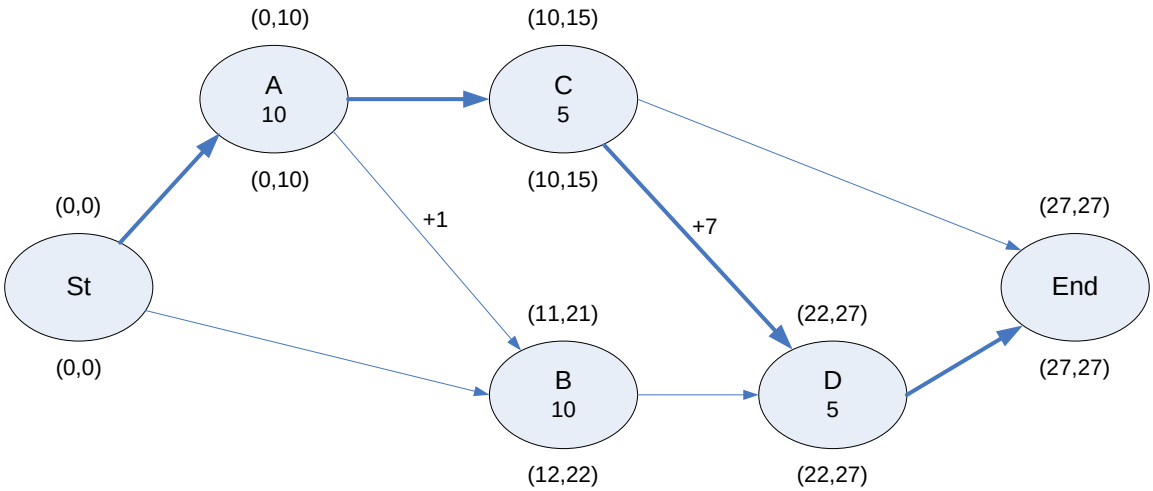
Université d'Ottawa · University of Ottawa

School of Electrical Engineering and Computer Science

ELG 5100, Assignment 1 Solution

Q1 a)

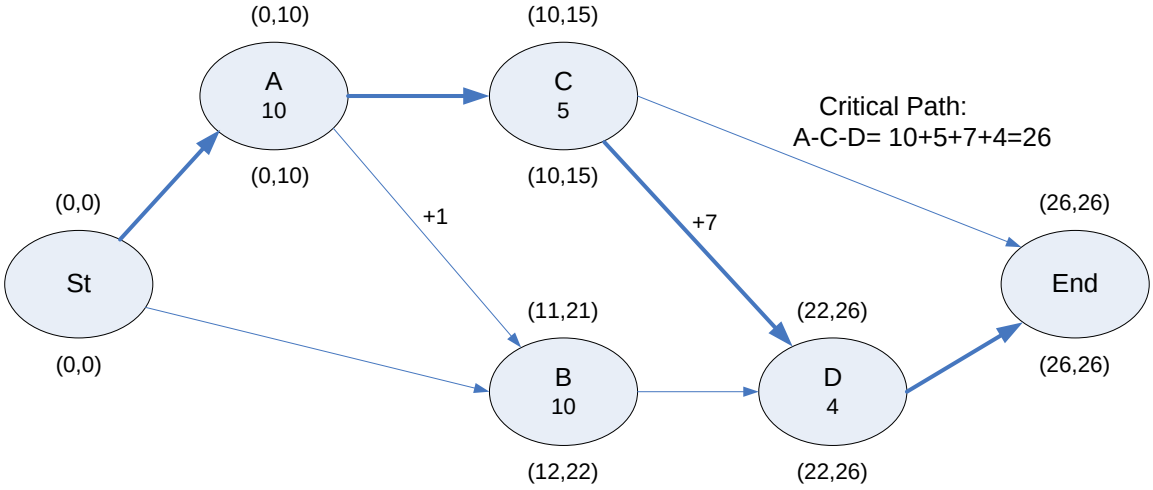


The critical path is A-C-D.

b) Initially the critical path is A-C-D.
Step 1: As the cheapest activity to crash on the critical path is D, we will crash it at step 1.

Activity	Maximum Crashing (months)	Cost per Month Crashed ($)
A	0	N/A
B	3	1000
C	2	800
D	(1-1)=0	500

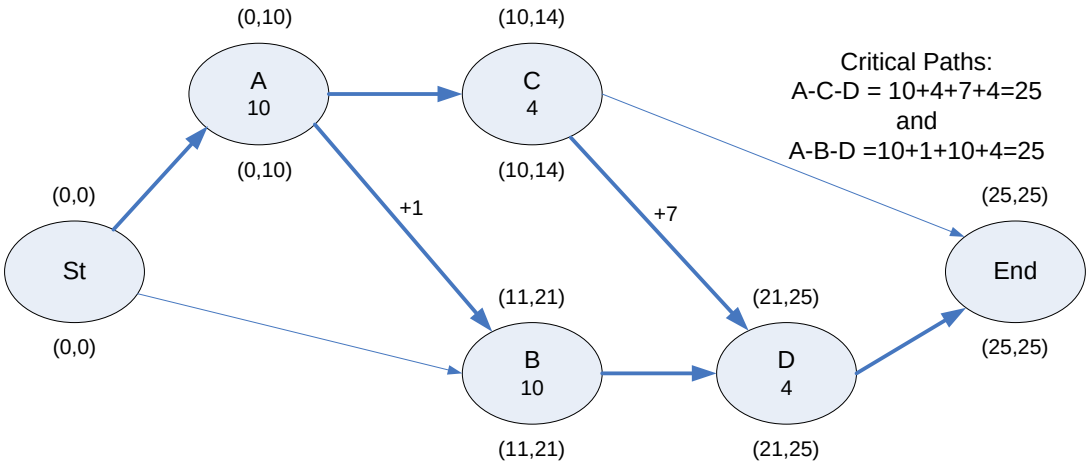
Total Extra Cost ($) = 500



Step 2: As D has already reached its maximum crashing limit, the cheapest activity to crash on the critical path is now C. We will crash C at step 2.

Activity	Maximum Crashing (months)	Cost per Month Crashed ($)
A	0	N/A
B	3	1000
C	(2-1)=1	800
D	0	500

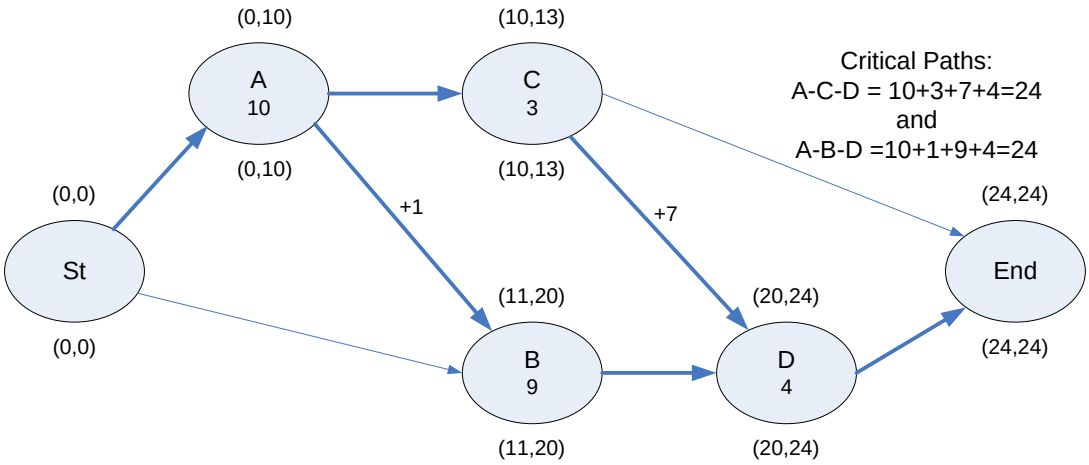
Total Extra Cost ($) = 500+800



Step 3: Now as we have two critical paths, we will have to find the cheapest activity on each of the critical paths and crash those simultaneously. On A-C-D the cheapest activity to crash is C and on A-B-D the cheapest activity to crash is B as D has reached its maximum crashing limit.

Activity	Maximum Crashing (months)	Cost per Month Crashed ($)
A	0	N/A
B	(3-1)=2	1000
C	(1-1)=0	800
D	0	500

Total Extra Cost ($) = 500+800 +800+1000



Step 4: Again we have two critical paths. On A-C-D we cannot crash any activity as both C and D have reached their maximum crashing limits. On A-B-D, we can crash B, but that will not reduce the length of the project since A-C-D will still be the critical path and the project will still be 24 months long. So there is no point in crashing B.

Therefore, the shortest finishing time is **24 months** and total extra cost is **$3100**.

Q2. When estimating size using Function Points, we multiply the raw estimate by a Complexity Adjustment Factor that is given by CAF = 0.65 + 0.01×N; where N is the weighted sum of all the 14 environmental factors, to obtain the adjusted estimate.

Now, we know that, for a given project, each of the 14 environmental factors is rated from 0 to 5. Therefore:

If we consider 0 for all:

CAF = 0.65 + 0.01×(14×0)
= 0.65 + 0
= 0.65, i.e. 1-.35 meaning -35%

If we consider 5 for all:

CAF = 0.65 + 0.01×(14×5)
= 0.65 + 0.7
= 1.35, i.e. 1+.35 meaning +35%

So, we can conclude that CAF is essentially a ±35% accuracy multiplier.

Q3. This is basically the phenomena of project cost as a function of time schedule. Sometimes it is described in articles, with a wry tone, as "nine women can't have one baby in one month". The reasoning is two-fold: resource de-allocation issues and communication overhead lead to this issue where more resources thrown at a project end up costing the project much more in terms of time and therefore cost. The reason we see shortened duration at all is on the account of parallelism and task division, which allows more effective critical path planning. But this is only achieved to a point, at some point communication becomes an issue.

Q4. Since the estimates are not drastically different (say, by 300% or more), and we are estimating the most important component of the software, the Project Manager's choice becomes less of a technical one, and more a question of being absolutely sure. To reduce risks, you should go with the scenario of 6000 LOC. An experienced PM would not squander efforts in technical analysis in this case.

Q5. If you are developing software very similar (almost like a duplicate) to what you have developed before, a Waterfall approach might be suitable. Also, if your customer is "old fashioned" and not yet caught up with the modern software development processes, like some military or government organizations.

Q6. Fixing design breakage increases in cost proportionally over time from the moment of introduction of the breakage. The cost is minimized at source, because there is no footprint on the next components or pieces of software; it is just on paper, and it is very easy to track the impact to a particular source. As development proceeds, every aspect described increases in complexity: you are building or testing software as opposed to a design, therefore you have to deal with the interrelationship between software components; it is very difficult to trace design breaks to all the corresponding locations in the code; instead "quick fixes" are applied that will decrease the robustness and quality of the software. Many additional considerations that have modified the code base since the design must also be considered.

Q7 a) At the beginning of the project, we want to make sure that the project is actually feasible and can be done. Therefore, we must make sure that the unknow/riskiest component of the project are put to trial first, usually through the construction of a proof-of-concept. This way, if the riskiest components need more time to be sorted out, we still have time (we are at the beginning), or, in the worst case, if the project is not doable, we stop it before incurring significant cost.

b) In the inception phase (the output of which is the proof-of-concept and the baseline architecture) and to a lesser extent also the elaboration phase (the out put of which is the prototype and the complete architecture).

c) By the end of the elaboration phase and for sure before the beginning of the construction phase.

Q8. For Basic Semiattached software project,

$$Effort(E) = 3.0 \times (Size)^{1.12} \quad \text{.....Eq. 1}$$

$$Time\ to\ Develop\ (TDEV) = 2.5 \times (E)^{0.35} \quad \text{.....Eq. 2}$$

Where, E is measured in staff-months, TDEV is in months and size is in KLOC.

According to the question, *E = 200 staff-days*

We assume that there are 20 working days in a month (other reasonable assumptions will also be accepted)

$$Therefore,\ E = 200/20 = 10\ staff\text{-}months$$

From Eq. 1 we get,

$$(Size)^{1.12} = E/3.0$$

$$\text{or, } Size = (E/3.0)^{(1/1.12)}$$

$$\text{or, } Size = (10/3.0)^{(1/1.12)} \qquad \text{[Substituting the value of E]}$$

$$\text{or, } Size = 2.93\ (approx.)$$

Therefore, the size of the software is approximately **2.93 KLOC**.

By substituting the value of E in Eq. 2 we get,

$$TDEV = 2.5 \times 10^{0.35}$$

$$\text{or, } TDEV = 5.6\ (approx.)$$

Therefore, given adequate resources, it will take approximately **5.6 months** to complete the development of this software.

Q8. The ripple effect refers to the fact that an error in one stage of a project will affect more than one artifact in the next stage of the project, therefore causing a ripple and producing more bugs as it propagates through the next stages. Its effect on the project budget is typically increased cost, because it becomes more expensive to correct those errors in the latter stages of the project.