A GUIDE TO NETWORK ANALYSIS
by
MICHAEL C GLEN

Introduction

The core technique available to Project Managers for planning and controlling their projects is Network Analysis. This short guide will provide a basic understanding of networking principles before applying them to the computer.

Network Analysis or Critical Path Analysis (CPA) or the American "Program, Evaluation and Review Technique" (PERT) is one of the classic methods of planning and controlling the progress of projects.

Tasks or Activities

Effective planning of projects requires careful thought and the application of logic. To illustrate this planning tool, let's consider the manufacture of a small item. Some typical processes might be:

cutting
finishing
assembling
purchasing
machining
testing
designing

All these processes are called '**ACTIVITIES**' or '**TASKS'**

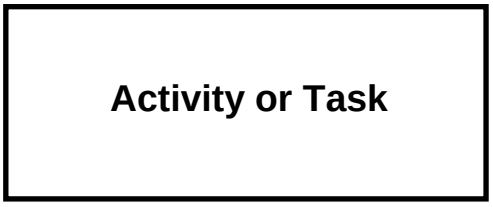
Step 1:

List **WHAT** has to be done.

Hint: try thinking of verbs ending in "...***ing"***, like ***machining*** or ***testing***.

Do not consider at this stage who is going to do what, concentrate on **WHAT.**

An activity or task is represented by a rectangle, thus:



Step 2:

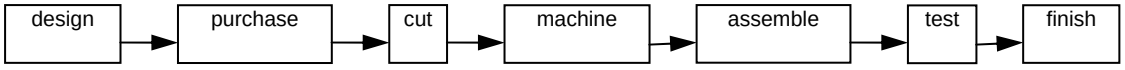
Decide the **ORDER** in which it is to be done.

Some steps are obvious: we, perhaps, cannot ***test*** until ***assembly*** has been completed, which cannot be done until the various parts have been made. So we have a logical relationship between the start of one task and the beginning of the next. We could order our list of tasks thus:

designing purchasing cutting machining assembling testing finishing

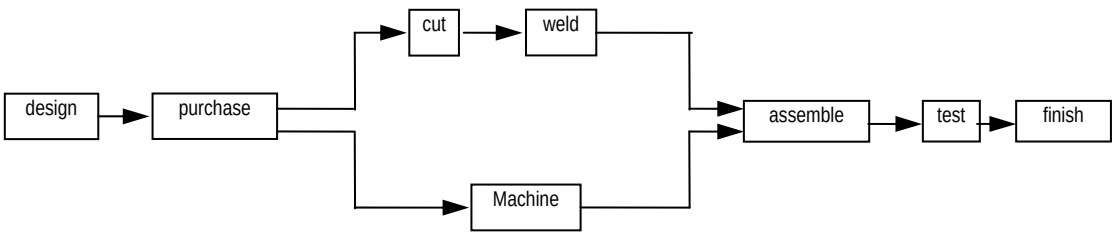
Logic Network or PERT Chart

Writing this out as a network:

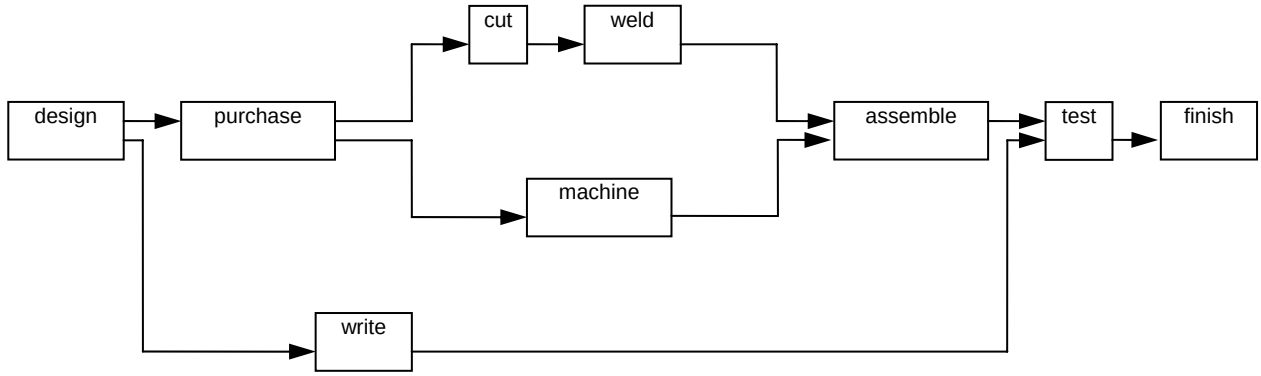


We put the tasks into rectangles and join them with arrows to show the sequence or precedence: the logical relationships between them.

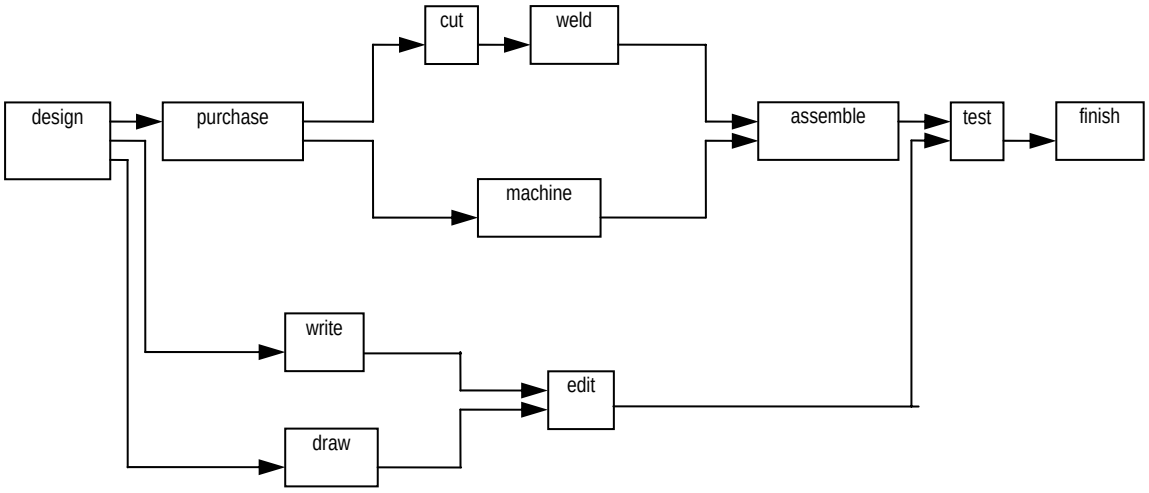
Suppose that once we have bought the materials, some need cutting to size and others need turning on a lathe. The tasks of ***machining*** and ***cutting*** could run in parallel rather than consecutively, assuming we have the appropriate resources. But let's add a bit more. Say the cut parts need to be ***welded*** together before assembling — like this:



Let's add another task: the ***writing*** of a set of test instructions. Where would ***writing*** fit in? Well, the ***writing*** cannot really start until the ***design*** is finished, though it could be carried out at the same time as the fabrication, but it must be ready before the ***testing*** can begin. Applying such logic to the relationships, we can add the task ***writing*** like this:



Now let's say we need to have our draughtsman produce some illustrations for our test instructions. When the ***writing*** and the ***drawing*** are finished, we will then need to ***edit*** the whole:



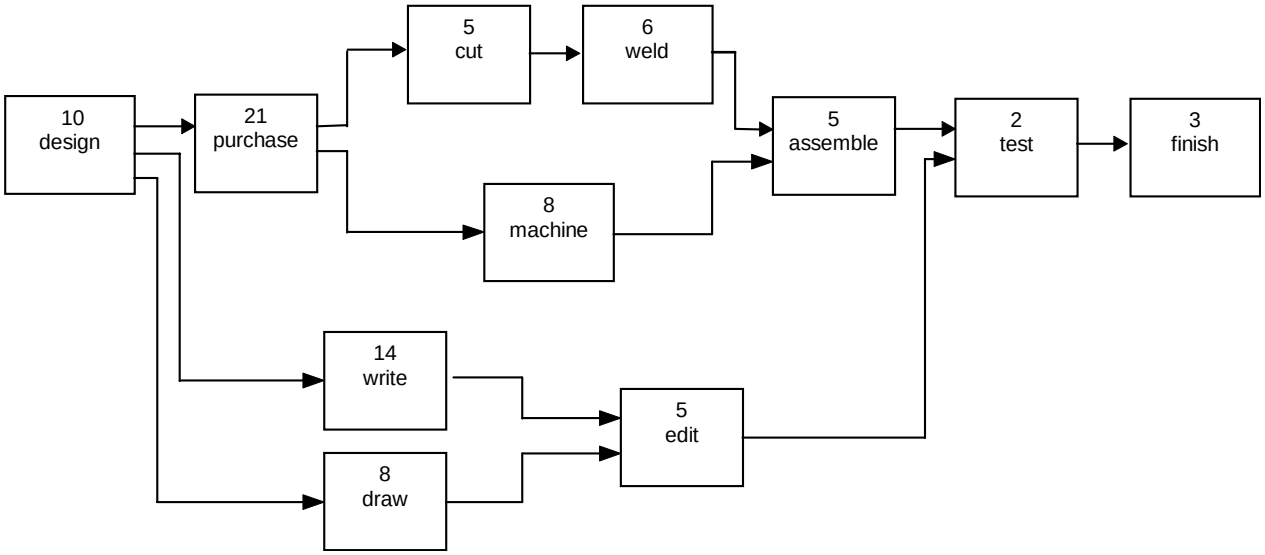
And so the network is built up, often cuing the mind to missing tasks.

In this step always assume you have <u>infinite resources</u> so that who does what does not cloud the issue – concentrate only on the **<u>LOGIC</u>**.

Time Analysis

Duration

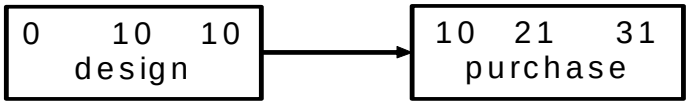
Having completed the network, we can begin the analysis. Firstly, we need to know the duration of each task and write it into the network. For convenience, we will write the durations in days, thus:



In this step, always reduce the resource requirement to the duration of **ONE** person to give maximum flexibility for you to add further resources later when the project begins to run late. [However, there are rare cases when tasks cannot physically be performed by one resource, in which case consider the time taken for both of them working together. For example, carrying a very long plank that requires a person at each end, adding more resources will not necessarily reduce the duration and might even slow it down if they get in the way of each other, but you must have two. Checking the brake lights on a car is another example.]

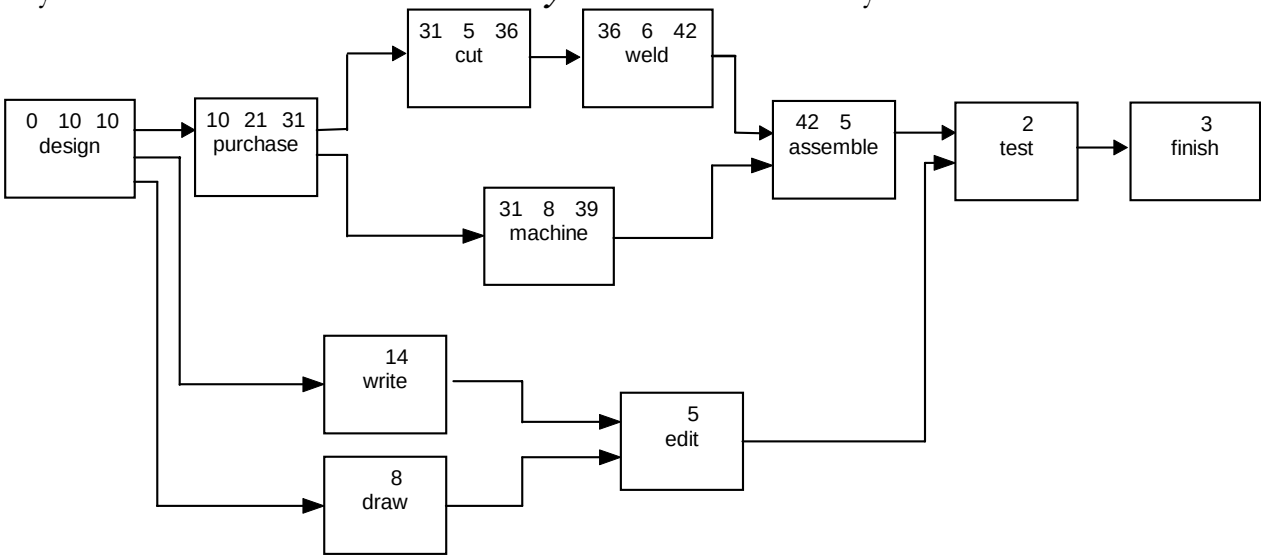
The Forward Pass

Now we can calculate how long the project will take. We start by calculating the shortest time or **'earliest finish'**. So if we start at time **'0'** the earliest day that ***design*** can be finished is day **10.** ***Purchasing*** therefore cannot start until day **10:** ie the earliest we can start ***purchasing*** is day **10**, and thus the earliest finish for ***purchasing*** is **10 + 21** = day **31.** We write the earliest start time in the top left-hand corner of the task box, and the earliest finish time over the right-hand corner of the task box.

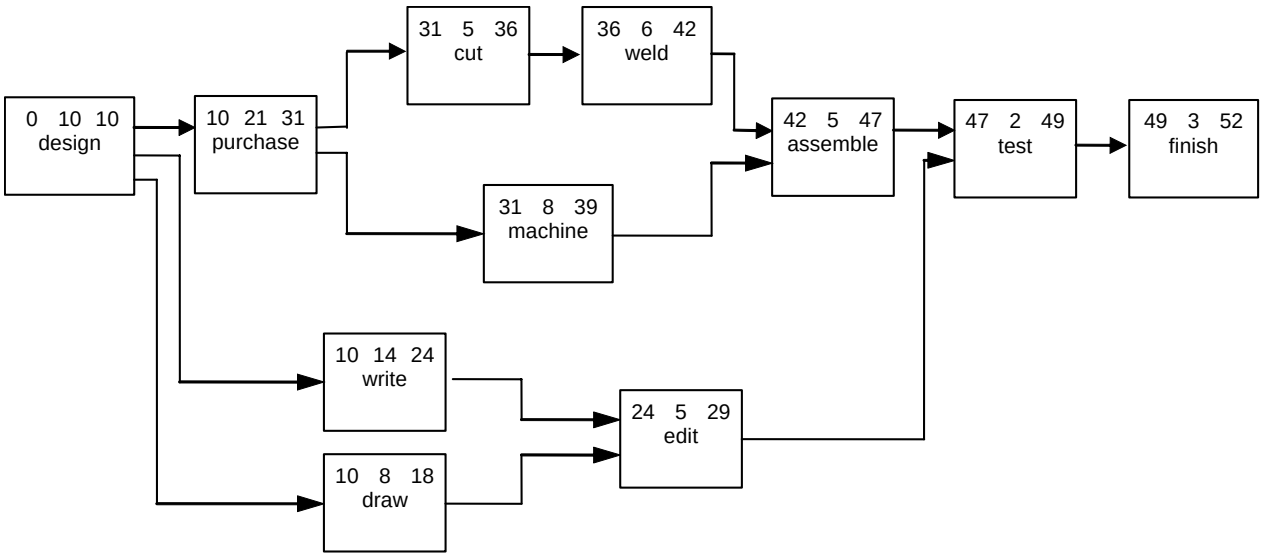


For the ***purchasing*** task, we thus say the **Earliest Start Time (EST)** is **10** and the **Earliest Finish Time (EFT)** is **31**

The earliest time both ***cutting*** and ***machining*** can start is day **31**. So the earliest finish for cutting will be day **36** and for machining day **39.** However, ***assembly*** cannot start at day **39** because the ***welding*** has not yet been done. The earliest ***welding*** can start is after the ***cutting*** is finished, ie day **36**. So the earliest time ***assembly*** can start must be day **42**.



Note that the task ***machine*** has some **float** or **slack**: ie it cannot start before day **31** and must finish before day **42**, but as it only takes **8** days, there is a slack of **42 - 31 - 8 = 3** days. When calculating the earliest times, you must consider all the paths or arrows coming into the task box, and select the **largest** or longest time. Now if we complete the timings, or **"Forward Pass"** as it is called, the picture will look like this:



So, the calculations show that the whole project will take **52** days or the **'earliest finish time'** is **52** days.

Slack

You will remember that when we considered the task ***machine*** we found that it had some **slack** available. You will have noted that the tasks along the paths ***draw***, ***write*** and ***edit*** also had some slack. The other tasks (ie those with no slack) are **critical** in that any delay in their completion will cause the whole project to be late. It is thus very important to discover the **critical tasks** and thus identify the **critical path** through the network.

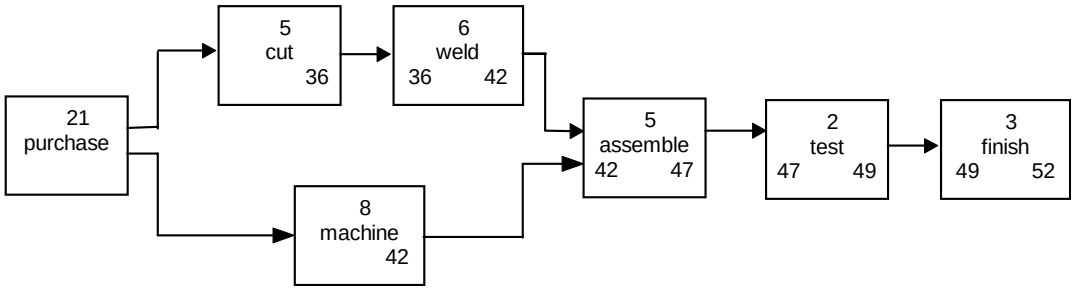
Critical Path

The next part of the analysis of the network is to find the **CRITICAL PATH**. By definition the **Critical Path** is the shortest time path through the network. In such a simple network, it is easy to calculate the amount of slack available for each task, but in a complicated network, it is not easy to 'see' which tasks have slack and which have none.

The Backward Pass

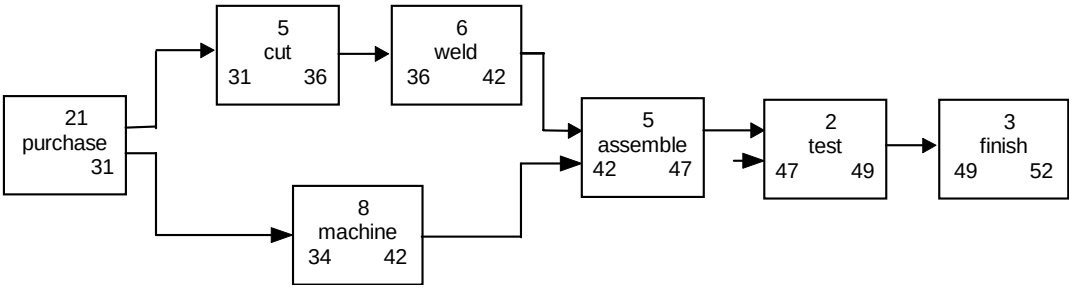
So far we have calculated the **Earliest Start Time (EST)** and the **Earliest Finish Time (EFT)** for each task. The next step in calculating the critical path is to re-time the network starting at the end and calculate the **Latest Start Time** (**LST**) and **Latest Finish Time (LFT**) for each task.

So, beginning with the earliest finish time for the whole project, ie **52** days, we subtract the time of the last task giving **49** days as the latest time the ***finishing*** task can begin without affecting the outcome.



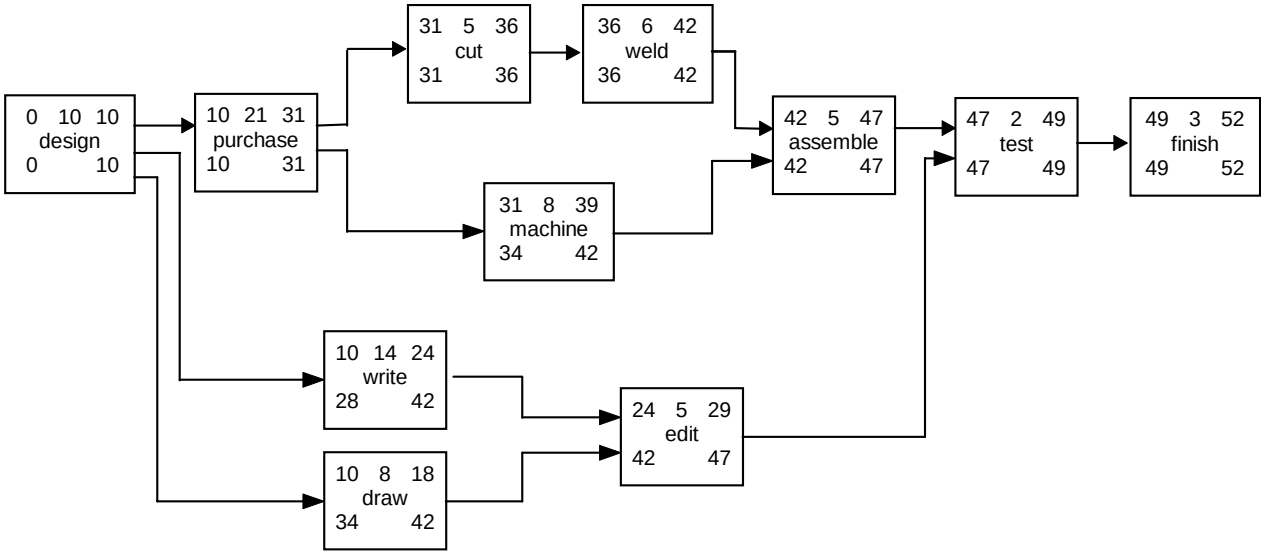
The latest time we can start ***testing*** is **49 - 2 = 47** days, and ***assembly*** is 42 days. Continuing backwards, against the arrows (doing what is called the "**backward pass**") the latest time ***welding*** can start is **42 - 6 = 36** days.

But what about ***cutting*** and ***machining***? What is the latest time ***purchasing*** can finish? Well, the latest time ***machining*** could start is **42 - 8 =** day **34**, and the latest time ***cutting*** could start is **36 - 5 =** day **31**. Hence, the latest time ***purchasing*** can finish is day **31**



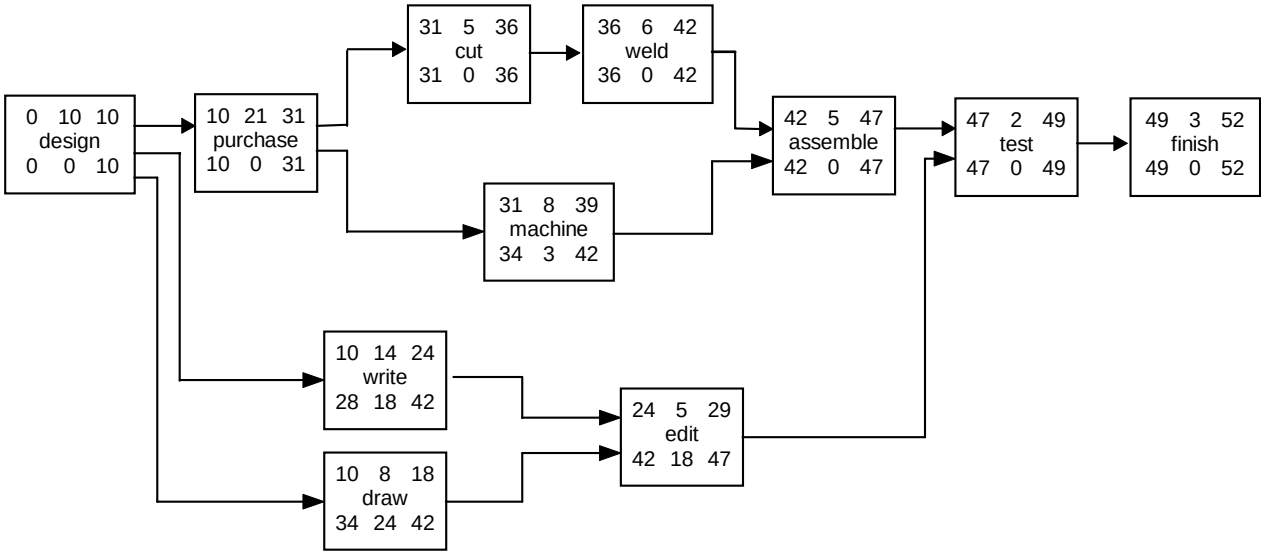
So, when making the backward pass to calculate the latest times, you have to consider all the arrows coming out of a task box and select the lowest or shortest time to that task.

Now to finish the backward pass we calculate the latest times for ***design*** to finish and start. The latest time ***writing*** could start is **42 - 14 =** day **28**, the latest time ***drawing*** can start is **42 - 8 = 34** and the latest time ***purchasing*** can start is **31 - 21 =** day **10**. Therefore, the latest time ***design*** can finish without affecting the project finish time is day **10**. Subtracting the time of the ***design*** task leaves us at day **0**, which is back to our starting point!



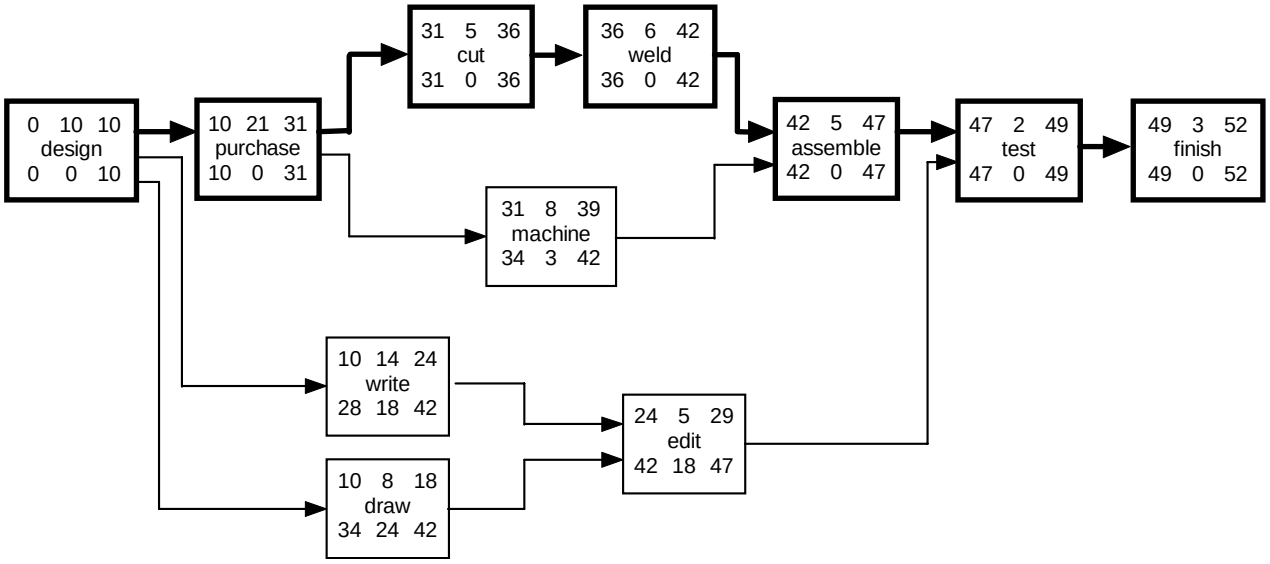
Slack

Now we have completed the timings by a forward and backward pass, we can look for slack. In this simple network, it is easy to see that the tasks ***machine***, ***draw*** and ***write*** have slack. As we have already discovered, ***machining*** can start at day **31** but must finish by day **42**. As it takes only **8** days to complete, the slack is **3** days. We will write the slack in the empty slot in the bottom centre of the box. With ***writing***, the task must finish by day **42** but it can start at day **10**. As it takes **14** days to complete, ***writing*** has a slack of **24 - 42 = 18** days. Similarly, ***drawing*** has a slack of **24** days and ***editing*** **18** days. The remaining tasks have no slack at all.



The Critical Path

So, to find the critical path, we look first at the tasks whose earliest and latest start times are identical. Then we engage brain to determine which path between such tasks has no slack. That path is then, by definition, the **critical path**. On the diagram below, the critical path is indicated by **bold** lines.



Well, there we are!

The critical path has been identified, we know the total time of the project and we know how much slack there is in the non-critical tasks. We have a structured plan that is logical and ordered. All we have to do now is assign resources, put it into action and control the results!