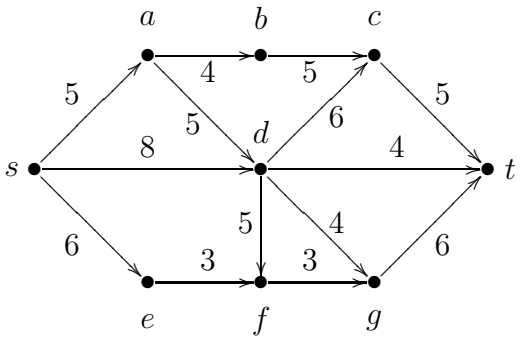
7 Networks: Critical Path Method

We use the term **network** to mean a directed graph (V, E) and the weight function $w : E \to \mathbf{R}^+$. One important application of networks is as follows. Consider a project which has a number of separate jobs which need to be carries out in a specified order. For example, in construction work, we have ground clearance and pouring of foundations, followed by various construction tasked, followed by finishing and decoration. The weight function corresponds to the length of time required to do the job. The directed graph structure is used to specify which jobs are prerequisites for other jobs. The aim is to schedule the jobs so that the work can be completed in minimal time.

The directed graph below describes a system which has 13 jobs required to proceed from the initial vertex s to the final vertex t. The weights on the edges represent the time required to do the corresponding job. The critical path method is an algorithm for finding the minimal time required subject to the constraints of the jobs being carried out in the order specified by the directed graph.



(7.1)

In order to find the **earliest completion time** for each vertex $E(y)$ we start with $E_0(v) = 0$ for all vertices and maintain Then we set $E_n(y) := \max_{(x,y)\in E} \Big(E_n(x)+w(x,y)\Big)$. We make repeated passes through all vertices until $E_n(v) = E_{n-1}(v)$ for all vertices. If $E_n(v)$ continues to increase, this indicates that the graph has a directed loop: this indicates an error in the formulation of the graph for the project. The earliest completion time $E(x) = E_n(x)$, once $E_n = E_{n-1}$. Here is the computation for the above graph:

	s	a	b	c	d	e	f	g	t
E_0	0	0	0	0	0	0	0	0	0
E_1	0	5	4	6	8	6	5	4	6
E_2	0	5	9	14	10	6	13	12	12
E_3	0	5	9	16	10	6	15	16	19
E_4	0	5	9	16	10	6	15	18	22
E_5	0	5	9	16	10	6	15	18	24
E_6	0	5	9	16	10	6	15	18	24

Note that it takes 5 steps for E_n to become stable. This is the length of the longest directed path from s to t.

Once E is computed, on can compute the **latest completion time** $L(x)$ for each vertex. This is a very similar calculation, but one works from t toward s. Initially one sets $L_0(x)$ to equal the earliest completion time $E(t)$ for the terminal vertex t. Then one uses the iteration $L_n(x) := \min_{(x,y)\in E} L_{n-1}(y) - w(x,y)$. After the same number of iterations required to produce a stable $E_n(v)$, L_n becomes stable. This is the function $L(v)$. Once E and L are computed we define the **float time** $F(x,y) = L(y)-E(x)-w(x,y)$. The job

corresponding to (x, y) may start at any time between $L(x)$ and $L(x) + F(x, y)$ while the total project is completed within $E(t)$. Those jobs with $F(x, y) = 0$ are called critical. There is at critical path, i.e., a path from s to t consisting of edges corresponding to critical jobs.

Here is the computation of L for the current example:

	s	a	b	c	d	e	f	g	t
L_0	24	24	24	24	24	24	24	24	24
L_1	16	19	19	19	18	21	21	18	24
L_2	10	13	14	19	13	18	15	18	24
L_3	5	8	14	19	10	12	15	18	24
L_4	2	5	14	19	10	12	15	18	24
L_5	0	5	14	19	10	12	15	18	24
E_5	0	5	9	16	10	6	15	18	24

We have included $E = E_5$ as the final line of the array. Vertices which have $E(v) = L(v)$ lie on a critical path.

There are variations of the above method which use E_n rather than E_{n-1} to compute E. The best of these order the edges in such a way that whenever both (x, y) and (y, z) are edges, then $(x, y) \prec (y, z)$, where $\prec$ denotes the order of the edges. The computation is first to set $E(v) \equiv 0$. Then $E(y) := \max_{(x,y)\in E} \Big(E(x) + w(x, y)\Big)$ is computed in a single pass taking the vertices in **increasing** $\prec$ order. There can be many orderings of the vertices which satisfy the requirement given above.

Single pass computation of earliest completion times $E(x)$ using ordered edges (the edges are listed in $\prec$ order):

se sa sd ab ad ef bc dc df dg dt fg ct gt

s	a	b	c	d	e	f	g	t
0	5	9	16	10	6	15	18	24

Once we have an order $\prec$ so that $(x, y) \prec (y, z)$ whenever both both (x, y) and (y, z) are edges, then L may be computed in a single pass. The computation begins with $L(v) \equiv E(t)$. Then $L(x) := \min_{(x,y)\in E} \Big(L(y) - w(x, y)\Big)$ is computed in a single pass using **decreasing** $\prec$ order.

Single pass computation of latest completion times $L(x)$ using ordered edges:

gt ct fg dt dg df dc bc ef ad ab sd sa se

s	a	b	c	d	e	f	g	t
0	5	14	19	10	12	15	18	24

Such an order can be found by using any weight function $w(x, y)$ which is strictly positive on the edges. Then E is computed by the *slow* method. Once E is known, then one chooses any order $\prec$ which makes $(u, v) \prec (x, y)$ whenever $E(u) < E(x)$. One could use L instead of E or require $(u, v) \prec (x, y)$ whenever $E(v) < E(y)$. Any of these orderings will suffice.