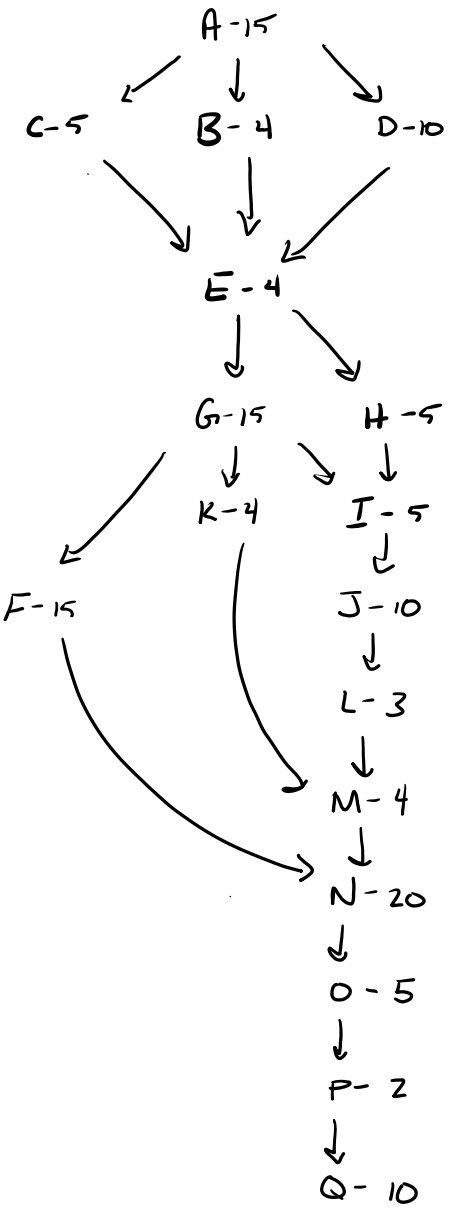
Applied Risk Analysis - L04



Goal - Schedule events Optimally

Step 1 - Identify the quantities we want
↳ Good way to identify variables.

We want the time that event starts.

Let $N(v)$ be the start time of event v
↳ Our variables! They are non-negative. (why?)

Step 2 - What are we optimizing?

Our ending time. In this case that is

$\text{Min} \rightarrow N(Q) + 10$ ← Not necessary, but then the objective value is total time.

Step 3 - Identify Constraints.

If event d is after event s

(or $s \rightarrow d$ then

$$N(d) \geq N(s) + \text{run-time}(s)$$

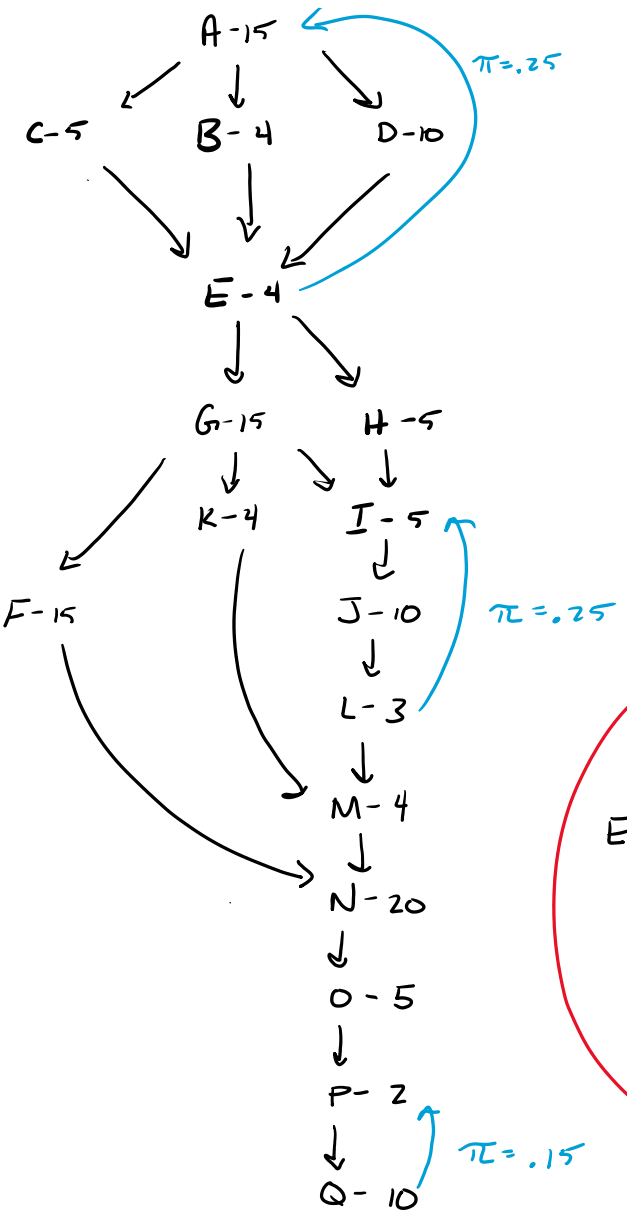
Note: This doesn't need to be an optimization problem. There is a very simple recursive algorithm that will solve this.

Translate Model into Julia

Model	Julia
	Step 0: `model = Model(_.Optimizer)` (Solver name)

Step 1: Variables
N[v] for each Vertex v.
	Step 1: Variables
	`@variables(model, begin` (model name; start listing variables)
	`N[V] >= 0`
	→ New line for more
	`end)`

Step 2: Objective N[Q] + run-time[Q]	Step 2: Objective `@objective(model, Min, N[:Q] + run_time[:Q])` (Can be Max; Notice the : This means symbol; objective function)
Step 3: Constraints If s → d then N(d) ≥ N(s) + run-time(s)	Step 3: Constraints `@constraint(model, subs[s=V, d=V; w[s,d] != 0],` `N[d] >= N[s] + run_time[s]` (Name; indices; In general, neither needed; Condition)
Step 4: Solve Magic	Step 4: Solve `optimize!(model)` (Julia convention, means modify first input.)
	Step 5: View Solution `value.(model[:N])` → View all of N (Julia syntax: apply function to each element of array.) OR `value(model[:N][:B])` → Just view N[B]



Goal: Find start times of events.

Step 1: What's changing?

① $N[v]$ → Start time of v.

② $ED[v]$ → Duration of v.
↳ bounded below by run time.

Step 2: Objective

Min → $N[Q] + run\_time[Q]$

Step 3: Constraints

If $s \to d$

$N[d] \geq N[s] + run\_time[s]$

For each $s \in V$ if $\sum_{d \in V} \pi[s,d] \neq 0$

$$ED[s] = \underbrace{\left(1 - \sum_{d \in V} \pi[s,d]\right)}_{\text{Prob of leaving } s} \cdot \underbrace{run\_time[s]}_{\text{standard run time}} + \sum_{d \in V} \underbrace{\pi[s,d]}_{\text{Prob of going back to } d} \cdot \underbrace{\left(N[s] - N[d] + ED[s]\right)}_{\text{time cost}}$$

This condition is irrelevant. Eliminating has no effect since $\sum_{d \in V} \pi[s,d] = 0$

So the constraint becomes

$$ED[s] = run\_time[s].$$

Why have it? Fewer constraints usually implies faster model. This model isn't large enough for that to matter.