Thursday, November 10

$\boxed{1}$ Welcome!

$\boxed{2}$ HW 7 due Tuesday

$\boxed{3}$ Project Presentation Plan

*original: 12/6, 12/8, 12/15 (w/ flexibility)

* or: 12/8 , 12/15 (12/6 work day)

12/8 ↓ 5 single, 1 duo

12/15 ↳ rest

$\boxed{4}$ 11/22: Tues. before Thanksgiving → progress report due.

$\boxed{5}$ Questions?

$\boxed{6}$ CPM

$\boxed{7}$ Outro → finish hw 7, keep @ project.

Math 494: Discrete Optimization 11/10

Today we're looking at an interesting applications of directed paths for planning out projects (maybe a bit on the nose with us spinning up our capstone work!). We'll approach this a bit differently, in that we'll work a few examples to try to develop the question and the method on the fly. So let's look at an example!

Example: Making a D&D character involves a lot of decisions, and many of them depend on each other. I need to choose a race, a class, an alignment, a subclass, a patron/deity, a background, and a name.

- I cannot pick a subclass before I pick a class.
- It helps if I have picked a class before I pick a race.
- My subclass and race will affect both which patron/deity I serve and my alignment.
- Both my patron and my alignment will influence my background.
- I've gotta pick my name last.

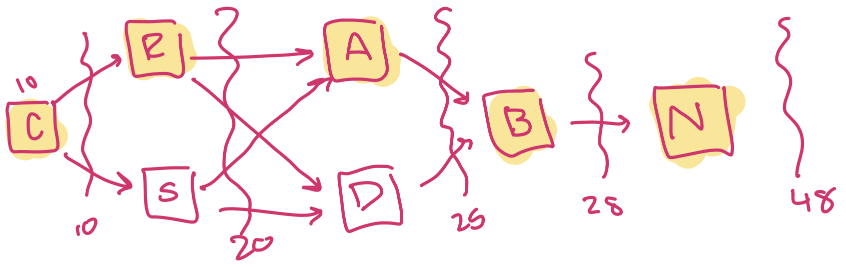
a.) In what order should I make my character choices? Is there some flexibility in this?

class, s/class, race, deity, alignment, background, name

C, S, R, A, D, B, N

C, R, S, A, D, B, N

b.) Can you visualize this decision making process with a directed graph?



c.) Suppose it takes me the following amounts of time for me to make these decisions, all listed in minutes: race (10), class (10), an alignment (5), a subclass (4), a patron/deity (3), a background (3), and a name (20). How quickly can I make a character? Assume that I can multitask decisions that do not depend on each other.

48 minutes

d.) Try to visualize your time on the graph you drew in b. Which events really drive the total time to make a character? → in yellow!

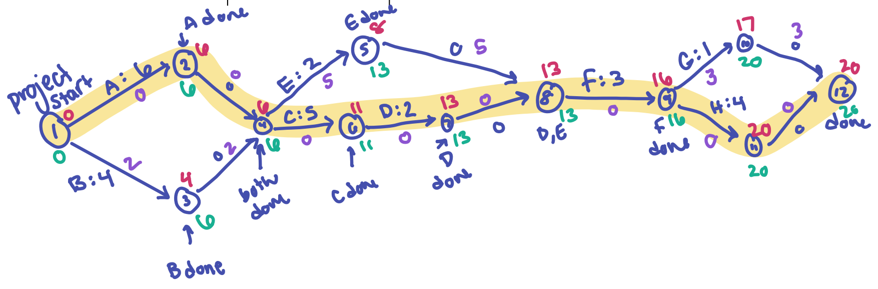
This is something called the *critical path method*. Suppose we are given a task that is divided several smaller activities. Each activity has a duration and a (possibly empty) list of predecessor activities that must occur before it can start.

From this information, we'll make an *activity graph*, a digraph where *edges* are activities, weighted with their duration. Vertices are starts or ends of activities, possibly with "dummy vertices/edges" if needed (we'll see why in a second). Vertices must be ordered in such a way such that for any edge ij, $i < j$.

Example: Remodeling a kitchen can be broken down into several smaller tasks, each of which has been listed (along with its predecessors and duration) below. Draw the activity graph.

Activity	Predecessor	Duration (days)
A: wiring	–	6
B: plumbing	–	4
C: drywall	A, B	5
D: painting	C	2
E: floors	A, B	2
F: cabinets	D, E	3
G: appliances	F	1
H: finishing	F	4

ET
LT
float

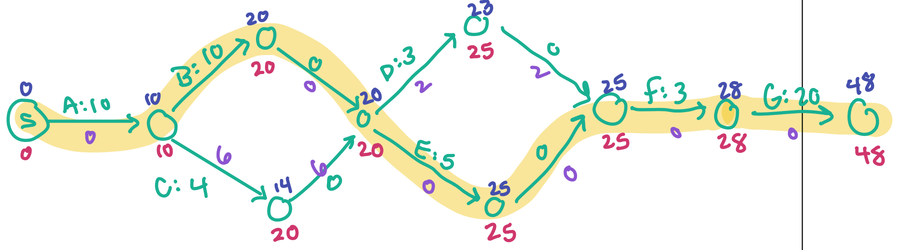


Example: Construct the activity table and graph from our D&D example.

task	pred	duration
A: class	—	10
B: race	A	10
C: s/c	A	4
D: deity	B, C	3
E: align	B, C	5
F: Backg	D, E	3
G: name	F	20

ET

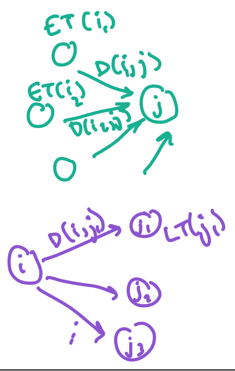
LT



So how do we actually calculate this? Well, we'll process the activity graph twice: once to calculate the earliest possible start time for each activity, once to calculate the latest possible start time for each activity without delaying the whole project.

Algorithm 1 CPM(G, D)

```
1:  G is an activity graph with required vertex ordering.
2:  Initialize $ET(1) = 0$.
3:  for $j \in \{2, 3 \ldots, n\}$ do
4:      Set $ET(j) = \max\{ET(i) + D(i,j) : ij \in E\}$
5:  end for
6:  Initialize $LT(n) = ET(n)$.
7:  for $j \in \{n-1, n-2, \ldots, 1\}$ do
8:      Set $LT(i) = \min\{LT(j) - D(i,j) : ij \in E\}$
9:  end for
10: $FL(i,j) = LT(j) - ET(i) - D(i,j)$
```



Example:

a.) Run the critical path method on the activity graphs we drew earlier.

b.) Which activities do you think are critical? Draw the critical path and then fill out the definitions below.

Definition. *(critical activity, critical path) A critical activity* ^{i,j} *is an activity such that* FL(i,j) = 0
A critical path is a directed $1 - n$ *path such that* each activity is critical.

Essentially, we're finding...

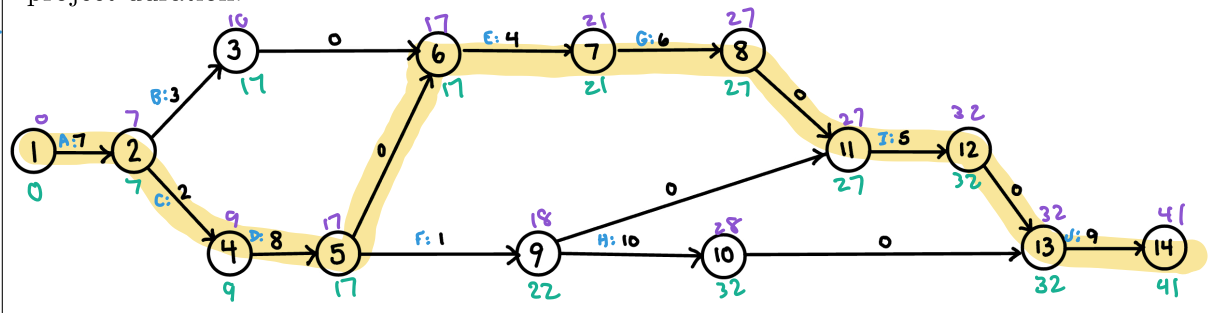
max weighted path from s to t.

Can you see some duality lurking around here?

max weighted path $\longleftrightarrow$ min project duration
$\updownarrow$ CP

Example: Given the activity graph below, determine the activity list and find the shortest project duration.

task	pred	dur
A	–	7
B	A	3
C	A	2
D	C	8
E	B,D	4
F	D	1
G	E	6
H	F	10
I	F,G	5
J	H,I	9



Critical path method gives us a great opportunity to talk a bit about directed acyclic graphs and topological orderings. Essentially, to run CPM, we need an ordering of the vertices such that for any edge $ij \in E$, $i < j$. This is called a topological ordering, and it's possible for directed acyclic graphs.

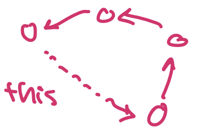
Definition. *(directed acyclic graph, topological ordering) A directed acyclic graph (DAG) is a directed graph with no directed cycles. A topological ordering of a directed acyclic graph is an ordering of the vertices $\{1, 2, \ldots n\}$ such that for every $ij \in E$, $i < j$.*

Example: Prove the following claim.

Claim. *Every directed acyclic graph admits a topological ordering.*

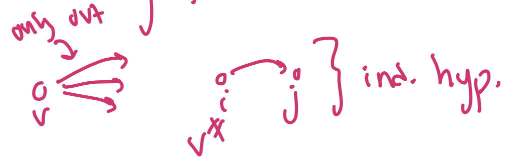
Pf: (ind. on # of vertices, n)

- base: n=1. ① ✓
- hyp: s/p that any dir. acyc. graph on k vert. admits a top. ordering.
- step: let D be a dir. acyc. graph on k+1 vertices. We Cl: ∃ vertex v st $\underbrace{d^+(v)}_{\text{in deg}} = 0$. s/p this was not the case. Then for every vertex, it has an in neighbor. by php, at some point, we must repeat vert. But this forms a directed cycle, ℰ.



G−v is a d.a.g. on k vertices, it has a topo ordering. Take this ordering and construct $v, \underbrace{x_1, x_2, \cdots x_k}_{\text{top. ordering of } G-v}$

this is a top. ordering for G.



Ok, great, they exist. How can we find one?

Example: Use the proof of the claim to describe how we can algorithmically find a topological ordering.

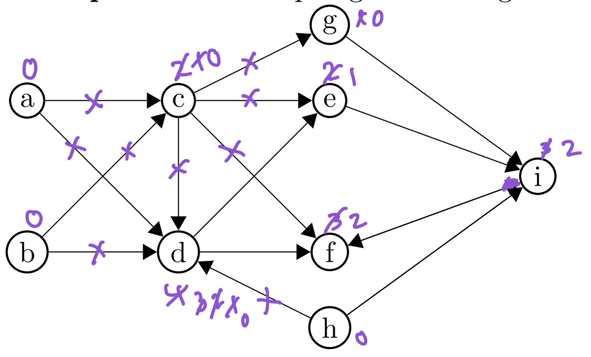
grab all $d^+(v)=0$ first then recurse.

Algorithm 2 BFSTopoSort(G, n)

```
$ScanQ \leftarrow \{\}$
$QSize \leftarrow 0$
$k \leftarrow 1$
for $v \leftarrow 1$ to $n$ do
    Compute $d^+(v)$
    if $d^+(v) = 0$ then
        $QSize \leftarrow QSize + 1$
        $ScanQ[QSize] \leftarrow v$
    end if
end for
while $k \leq QSize$ do
    $u \leftarrow ScanQ[k]$
    for each $v$ such that $u \to v$ do        ↙ anything a in deg =0 vertex points to
        $d^+(v) \leftarrow d^+(v) - 1$        ↙ delete 1 from in deg. → remove the edge!
        if $d^+(v) = 0$ then
            $QSize \leftarrow QSize + 1$      } add it.
            $ScanQ[QSize] \leftarrow v$
        end if
    end for
    $k \leftarrow k + 1$
end while
```

Follow up: what would happen if we fed a digraph with directed cycles in?

Example: Run the topological sort algorithm on the graph below.



ScanQ : [a, b, h, c, d g