MATH20902: Discrete Maths, Solutions to Problem Set 8

---

(1). The graph in question is above, with the distinguished starting vertex, r, shown in orange.

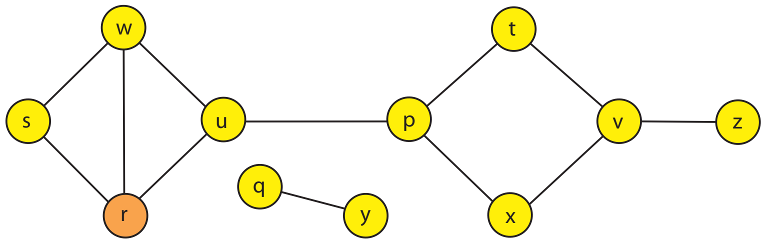


Figure 1: *A representation of the graph that was defined, in the problem statement, in terms of adjacency lists.*

Once the graph is drawn, it's easy to see that there are two shortest paths from r to z, both of length 5. If we describe them as ordered lists of vertices they are

$$(r, u, p, x, v, z) \qquad \text{and} \qquad (r, u, p, t, v, z).$$

The main work in the problem is to use Breadth First Search (BFS) to find the distances $d(r, .)$ from vertex r to each of the others, which are summarised below.

Vertex	p	q	r	s	t	u	v	w	x	y	z
$d(r, \cdot)$	2	∞	0	1	3	1	4	1	3	∞	5

(2) (Basic tropical arithmetic)**.** These exercises are meant to highlight the way tropical arithmetic differs from ordinary arithmetic.

(a) $1 \oplus 3 = \min(1,\ 3) = 1$

(b) $1 \oplus -1 = \min(1,\ -1) = -1$

(c) $0 \oplus 5 = \min(0,\ 5) = 0$

(d) $1 \otimes 3 = 1 + 3 = 4$

(e) $1 \otimes -1 = 1 + (-1) = 0$

(f) $0 \otimes 5 = 0 + 5 = 5$

(g) $0 \oplus \infty = \min(0,\ \infty) = 0$

(h) $0 \otimes \infty = \infty$

(i) $3 \otimes (3 \oplus 5) = 3 + \min(3,\ 5) = 6$

(3) (Tropical matrix arithmetic)**.** The tropical matrix product is formally similar to ordinary matrix products, save that one uses the tropical versions of multiplication and addition:

$$\begin{aligned}\begin{bmatrix} 1 & 2 \\ 0 & \infty \end{bmatrix} \otimes \begin{bmatrix} \infty & -1 \\ 1 & \infty \end{bmatrix} &= \begin{bmatrix} (1 \otimes \infty) \oplus (2 \otimes 1) & (1 \otimes -1) \oplus (2 \otimes \infty) \\ (0 \otimes \infty) \oplus (\infty \otimes 1) & (0 \otimes -1) \oplus (2 \otimes 1) \end{bmatrix} \\ &= \begin{bmatrix} \min((1+\infty),\, (2+1)) & \min((1-1),\, (2+\infty)) \\ \min((0+\infty),\, (\infty+1)) & \min((0-1),\, (2+1)) \end{bmatrix} \\ &= \begin{bmatrix} \min(\infty,\, 3) & \min(0,\, \infty) \\ \min(\infty,\, \infty) & \min(-1,\, 3) \end{bmatrix} \\ &= \begin{bmatrix} 3 & 0 \\ \infty & -1 \end{bmatrix}\end{aligned}$$

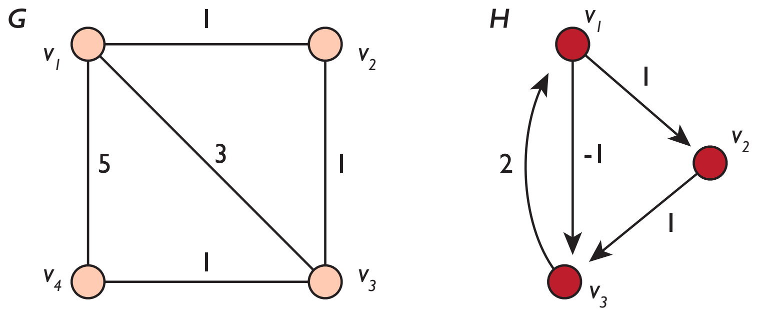
The tropical matrix sum is even simpler:

$$\begin{aligned}\begin{bmatrix} 1 & 2 \\ 0 & \infty \end{bmatrix} \oplus \begin{bmatrix} \infty & -1 \\ 1 & \infty \end{bmatrix} &= \begin{bmatrix} 1 \oplus \infty & 2 \oplus -1 \\ 0 \oplus 1 & \infty \oplus \infty \end{bmatrix} \\ &= \begin{bmatrix} \min(1,\, \infty) & \min(2,\, -1) \\ \min(0,\, 1) & \min(\infty,\, \infty) \end{bmatrix} \\ &= \begin{bmatrix} 1 & -1 \\ 0 & \infty \end{bmatrix}\end{aligned}$$

(4) (Distance in graphs)**.** This question is about using tropical arithmetic to compute the weights of minimal-weight walks. Suppose G is a weighted graph and that we have assembled a weight matrix W with entries given by Eqn. (4.2) below. Then

$$W^{\otimes \ell}_{jk} = \text{minimal weight for walk from } v_j \text{ to } v_k \text{ with } \ell \text{ or fewer edges} .$$

One can use this idea as the basis for an algorithm that computes the minimal weight for a walk connecting all pairs of vertices in G and the exercise was designed to verify this approach for the two graphs illustrated below.



(a) The first part of the question asks us to compute, by inspection, the weight of the minimal-weight walks between all pairs of vertices and to tabulate the results in a matrix D. Here they are:

$$D_G = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix} \quad \text{and} \quad D_H = \begin{bmatrix} 0 & 1 & -1 \\ 3 & 0 & 1 \\ 2 & 3 & 0 \end{bmatrix}. \tag{4.1}$$

Note that paths contributing to these weights involve at most three edges.

(b) Next we are asked to construct weight matrices W whose entries are

$$W_{j,k} = \begin{cases} 0 & \text{if } j = k \\ w(v_j, v_k) & \text{if } j \neq k \text{ and } (v_j, v_k) \in E \\ \infty & \text{otherwise} \end{cases} \tag{4.2}$$

and these are

$$W_G = \begin{bmatrix} 0 & 1 & 3 & 5 \\ 1 & 0 & 1 & \infty \\ 3 & 1 & 0 & 1 \\ 5 & \infty & 1 & 0 \end{bmatrix} \quad \text{and} \quad W_H = \begin{bmatrix} 0 & 1 & -1 \\ \infty & 0 & 1 \\ 2 & \infty & 0 \end{bmatrix}.$$

(c) Finally, we are asked to compute some tropical powers of the weight matrices. Given that G has $n = 4$ vertices, the desired result is

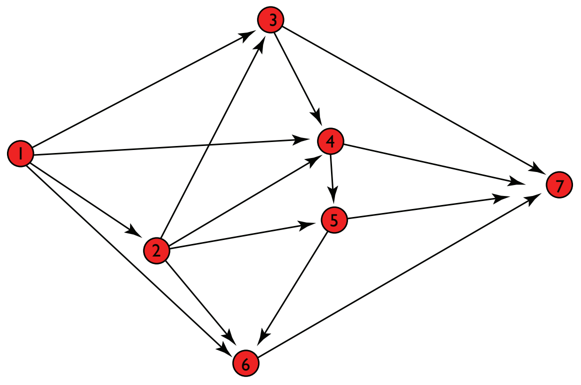
$$W_G^{\otimes(n-1)} = W_G^{\otimes 3} = \begin{bmatrix} 0 & 1 & 2 & 3 \\ 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 1 \\ 3 & 2 & 1 & 0 \end{bmatrix},$$

and as one expects, $W_G^{\otimes 3}$ agrees with the matrix D_G in Eqn. (4.1). Similar calculations with W_H yield

$$W_H^{\otimes 2} = \begin{bmatrix} 0 & 1 & -1 \\ 3 & 0 & 1 \\ 2 & 3 & 0 \end{bmatrix}$$

which again agrees with Eqn. (4.1).

(5) (After Jungnickel's Exercise 2.6.9)**.**
The numbering on the vertices below has the desired property:

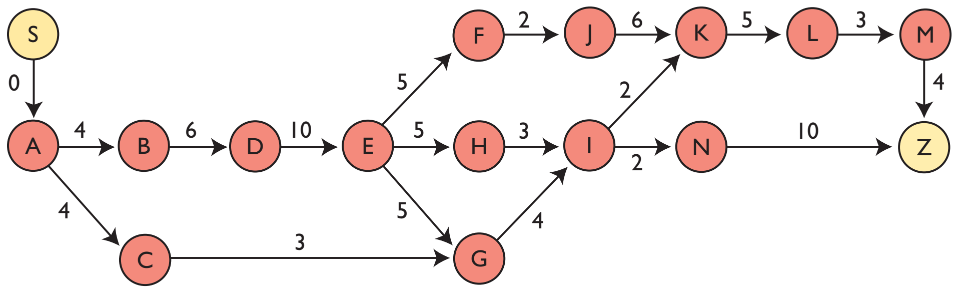


Indeed, one can always find a topological sorting for an acyclic directed graph.

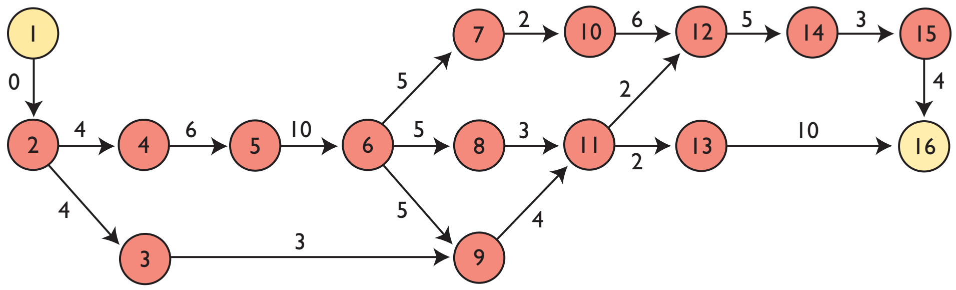
(6) (A scheduling problem)**.** This problem came from the first chapter of

> J. A. Dossey, A. D. Otto, L. E. Spence and C. Vanden Eynden (2006), *Discrete Mathematics*, 5th edition, Addison Wesley. ISBN: 0321305159.

(a) The main work in solving this problem is to draw a suitable graph. Below is one version, in which the vertices corresponding to tasks are reddish while the two special vertices—S for the start and Z for the finish—are a pale yellow. The numbers next to the directed edges are the time, in days, needed to complete the task corresponding to the vertex at the start (or tail) of the edge.



(c) One way to find the necessary times is to make a topological sorting of the graph and then solve a certain set of Bellman-like equations that we discussed in lecture. First recall that a topological sorting of a digraph G with vertex set V and edge set E is a numbering scheme for the vertices which ensures that for all edges $(i, j) \in E$, we have $i < j$. Here is a version of the project's scheduling graph with suitable numbers.



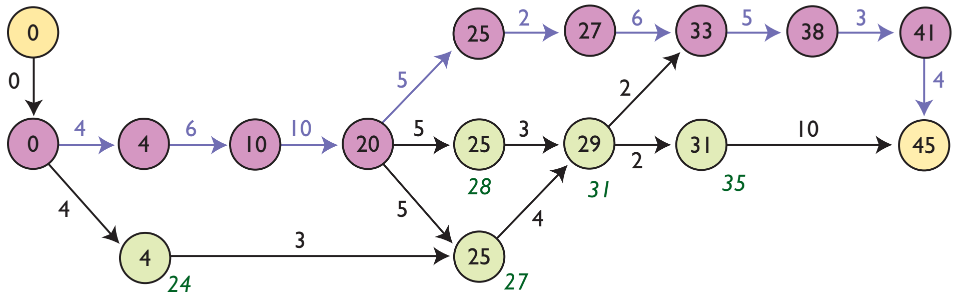
Now we define t_j to be the earliest time at which we could start task j given that we must complete all its prerequisite tasks. When the vertices are labelled with a topological sorting, these times satisfy

$$t_1 = 0 \qquad \text{and} \qquad t_j = \max_{\substack{k<j \\ (k,j)\in E}} t_k + w_{k,j}$$

and so are especially easy to solve. For example

$$\begin{aligned}
t_2 &= & t_1 + w_{1,2} &= & 0+0 &= 0 \\
t_3 &= & t_2 + w_{2,3} &= & 0+4 &= 4 \\
t_4 &= & t_2 + w_{2,4} &= & 0+4 &= 4 \\
t_5 &= & t_4 + w_{4,5} &= & 4+6 &= 10 \\
\vdots &= & \vdots &= & \vdots &= \\
t_9 &= & \max\left(t_3 + w_{3,9},\, t_6 + w_{6,9}\right) &= & \max\left(4+3,\, 20+5\right) &= 25 \\
\vdots &= & \vdots &= & \vdots &= \\
t_{16} &= & \max\left(t_{13} + w_{13,16},\, t_{15} + w_{15,16}\right) &= & \max\left(31+10,\, 41+4\right) &= 45
\end{aligned}$$

The figure below has the vertex label j replaced with t_j, the earliest time at which the j-th task can start.



The critical path is shown in purple: it involves the sequential completion of the tasks that were originally labelled *A, B, D, E, F, J, K, L, M* and requires a minimum of 45 days.

(d) Defining $T_{16} = t_{16} = 45$ and working backward according to

$$T_j = \min_{\substack{k>j \\ (j,k)\in E}} T_k - w_{j,k},$$

one can also find T_j, which is the latest time at which task j can start if it is not to delay the project. For tasks on the critical path $t_j = T_j$, but for the others, whose vertices are pale green in the figure above, there is a bit of slack and $T_j > t_j$. For these vertices, C, G, H, I and N, T_j is given in italics. The most dramatic example is vertex C, Drains and Services: although the builder could, in principle, start this task after only 4 days, the work doesn't become critical to timely completion of the project until day 24.

(7) (after Jungnickel's Example 3.1.2)**.**
Given that the probability of getting caught on a given edge e is $p(e)$, the probability of *not* getting caught is $(1 - p(e))$ and so, using the assumption of independence, the probability of not getting caught on a whole journey that traverses the edges $(e_1, e_2, \ldots, e_n)$ is

$$P = \prod_{j=1}^{n} (1 - p(e_j))$$

Now consider the log of this quantity:

$$\begin{aligned}\log(P) &= \log\left(\prod_{j=1}^{n}(1-p(e_j))\right) \\ &= \sum_{j=1}^{n}\log(1-p(e_j)) \qquad (7.3)\end{aligned}$$

Certainly it would be just as good to maximize this, as $\log(P)$ is a monotonically increasing function of P. But one should bear in mind that $\log(P)$ is, necessarily, a negative number: $0 \leq (1-p(e)) \leq 1$ and so $\log(1-p(e)) \leq 0$ for any edge e. Thus

$$|\log(P)| \;=\; -\log(P) \;=\; \sum_{j=1}^{n} -\log(1-p(e_j))$$

and the problem of maximizing $\log(P)$ is the same as the problem of *minimizing* $|\log(P)|$. And that means that we can minimize (7.3) by solving a shortest-path problem with edge weights given by $w(e) = -\log(1-p(e))$. All these weights are obviously positive, so the graph can't contain any cycles of negative length and the shortest path problem is well-posed.

(8) (Interesting, but not examinable: Tropical polynomials)**.** A *tropical monomial* is one of the terms in a tropical polynomial and has a very simple meaning in ordinary arithmetic:

$$a \otimes x^{\otimes n} \;=\; a \otimes \underbrace{x \otimes \cdots \otimes x}_{n \text{ times}} \;=\; a + \underbrace{x + \cdots + x}_{n \text{ times}} \;=\; a + (n \times x).$$

A tropical polynomial is the tropical sum of its monomials and so, for the quadratic in the problem, we have

$$\begin{aligned}p(x) &= (1 \otimes x^{\otimes 2}) \;\oplus\; (2 \otimes x) \;\oplus\; 5 \\ &= \min(2x+1,\, x+2,\, 5). \qquad (8.4)\end{aligned}$$

Which of these three terms is minimal clearly depends on x: when $x \gg 0$ the least of the three will be 5, but when $x \ll 0$ it will be $2x+1$. The value of $p(x)$ is plotted for an illustrative range of x in Figure 2.

Afterword:
People who work in this area often talk about the *roots* of a tropical polynomial. They occur at those values of x where two (or more) tropical monomials are equal, so that our $p(x)$ would have two roots, one at the place where

$$2x+1 = x+2 \implies x = 1$$

and the other where

$$x+2 = 5 \implies x = 3.$$

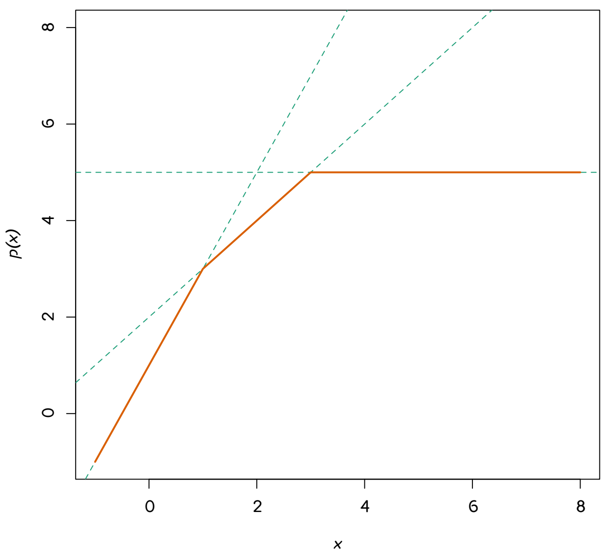


Figure 2: *The tropical quadratic $p(x)$ from Eqn. (8.4) is shown as a solid, orange piecewise-linear curve, while the values of the three monomials are given by the dashed green lines $y = 2x + 1$, $y = x + 2$ and $y = 5$.*

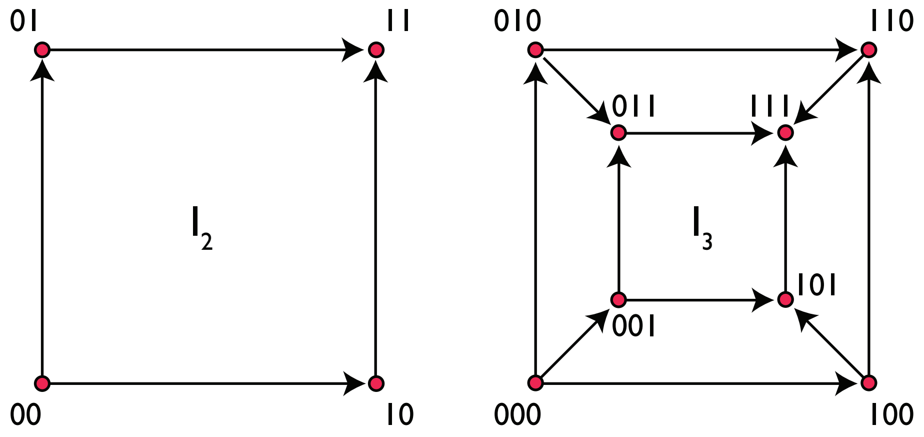
It begins to seem more sensible to call these values of x "roots" when one learns that they permit one to factor the polynomial into a tropical product that looks similar to the factorisations of ordinary polynomials:

$$p(x) \;=\; (1 \otimes x^{\otimes 2}) \;\oplus\; (2 \otimes x) \;\oplus\; 5 \;=\; 1 \otimes (x \oplus 1) \otimes (x \oplus 3).$$

(9) (After Jungnickel's Exercise 3.1.3)**.**
I will give two solutions: one that's easy to think about, but very expensive to compute and another that is less obvious, but uses a much smaller graph. Jungnickel gives yet a third solution, though it is similar in spirit to the second one here.

The most straightforward way to solve the problem is via a modified version of the cube graph I_n. Number the items 1 to n. Now say that a vertex $v \in I_n$ with label $s_1 \ldots s_n$, where $s_j \in \{0, 1\}$, corresponds to a knapsack that contains item k iff $s_k = 1$. Thus if $n = 4$ the vertex labelled 0000 corresponds to an empty knapsack, while the vertex with label 0101 corresponds to a knapsack containing only items 2 and 4.

Now think about the edges in the graph: they connect vertices whose labels differ by one symbol. It's easy to see that this means that for each edge (u, v), one of the labels of the vertices will have exactly one more 1 than the other and so will correspond to a rucksack containing exactly one more item. Convert I_n into a directed graph by making each of its edges run from the vertex with fewer 1's to the one with more. The figure below shows the result of applying this procedure to I_2 and I_3



With this approach, the directed edges also have an interpretation in terms of adding an item to the knapsack. For example, when $n = 3$, the directed edge $e = (000, 010)$ means "put item 2 into the empty knapsack". It is thus natural to assign this edge the weight $w(000, 010) = c_2$, the value of item 2. More generally, assign edge weight $w(u, v)$ to be the value of the single item that is added to the knapsack when passing from vertex u to v.

We still haven't taken account of the weight limit b. To do this, add one last vertex, called, say, t for "terminus", to the graph. Then compute the weight

$$W(s_1, \dots s_n) = \sum_{j \,|\, s_j = 1} a_j \tag{9.1}$$

of the knapsack that corresponds to each vertex and then draw a directed edge from any vertex with $W(s_1, \dots s_n) \leq b$ to the terminal vertex t and assign $w(v, t) = 0$ for each of this last group of edges. Then a longest (in the sense of the edge weights c_j) path from the vertex corresponding to the empty knapsack to t gives the most valuable load that the pedlar can carry.

As I_n has 2^n vertices, this first solution can lead to a very big graph. And in order to enforce the condition on the total weight, we had actually to compute something—the total weight (9.1)—for each of these many vertices.

Here is another setup that requires many fewer vertices. It corresponds to the process of considering the objects in numerical order and never adding an object whose index is less than that of one already in the knapsack. Begin the graph with a starting vertex s that corresponds to an empty knapsack. Now make a $n \times b$ rectangular array of other vertices whose n rows correspond to the n distinct objects and whose b columns correspond to the allowed positive values $1, \dots b$ of the total weight of the knapsack. Label the entries in this block as $[j, W]$ where $1 \leq j \leq n$ indicates an object and $0 \leq W \leq b$ is an allowed total weight for the knapsack.

Draw a directed edge from the starting vertex s to each of the n vertices whose labels are of the form $[j, a_j]$ and give each of these edges weight

$$w(s, [j, a_j]) = c_j.$$

These correspond to the idea that adding the j-th item increases the weight of the empty knapsack to a_j and its value by c_j. Next, draw a directed edge from vertex

$[j_1, W_1]$ to vertex $[j_2, W_2]$ whenever $j_2 > j_1$ and

$$W_2 = W_1 + a_{j_2}.$$

Give these edges weight (in the sense of path length)

$$w([j_1, W_1], [j_2, W_2]) = c_{j_2}$$

Here again, a directed edge corresponds to adding an object to the knapsack, but the vertices are now labelled by the index of the most recently-added object and the total weight of the knapsack's contents. This part of the construction simultaneously enforces the rule that we consider adding the objects to the knapsack in order of increasing index (we require $j_1 < j_2$) and the bound on weight (there are no vertices with $W > b$).

After all the edges from the previous step have been added we finish the construction by adding a terminal vertex, t, and edges running to t from every vertex $[j, W]$ with the property that $\deg_{\text{in}}[j, W] > 0$. This final group of edges all receive weight $w([j, W], t) = 0$.

Now think about a path from s to t, described as a sequence of vertices

$$(s, [j_1, W_1],\ [j_2, W_2], \ldots, [j_k, W_k],\ t).$$

This corresponds to a knapsack containing the items numbered $j_1, j_2, \ldots, j_k$ and its contents have total weight

$$\begin{aligned} W_k &= W_{k-1} + a_{j_k} \\ &= W_{k-2} + a_{j_{k-1}} + a_{j_k} \\ &\vdots \\ &= \sum_{i=1}^{k} a_{j_i}. \end{aligned}$$

and total value given by

$$\sum_{i=1}^{k} c_{j_i} \;=\; w(s, [j_1, W_1 = a_{j_1}]) + \sum_{i=1}^{k-1} w([j_i, W_i], [j_{i+1}, W_{i+1}]).$$

Clearly a longest path here produces the most valuable bundle of goods, but the graph has many fewer vertices, so the computation should be quicker.