Exercise 1: Machinery, Overtime, or Nothing?

Consider again the same very simple example. We have done additional market research, and we now know that demand will increase with probability at least 0.6, and at most 0.65.

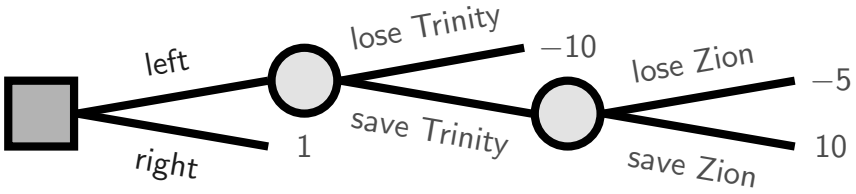
What advice can we give the manager now? Investigate with each optimality criterion.

Hint: $\mathcal{P} =$

	p_1	p_2
increase	0.6	0.65
stay	0.4	0.35

Exercise 2: Saving Zion (Or Maybe Not?)

There are two doors. The door to your right leads to the Source and the salvation of Zion. The door to your left leads back to the Matrix, to her. . . and to the end of your species. As you adequately put, the problem is choice. But we already know what you are going to do, don't we?



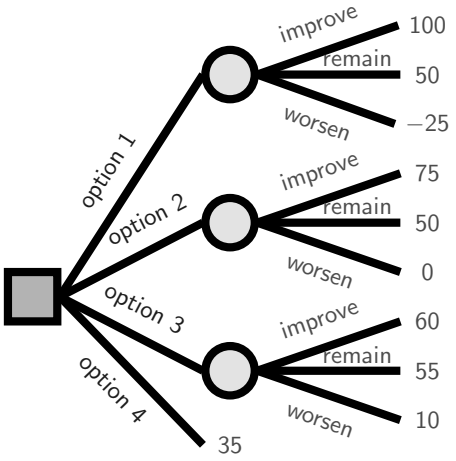
$\mathcal{P} =$

	p_1	p_2	p_3
lose Trin	0.1	0.4	0.3
save Trin & lose Zion	0.45	0.3	0.2
save Trin & save Zion	0.45	0.3	0.5

Left, or right? Investigate with your favorite optimality criterion.

Exercise 3: A Risky Investment

You have the option to invest some money. The market can either improve, remain, or worsen. The set of probabilities for your lower prevision are tabulated below. You have the choice between 4 options, summarized in the decision tree below.



$$\mathcal{P} = \begin{array}{c|cc} & p_1 & p_2 \\ \hline \text{improve} & 0.0 & 0.3 \\ \text{remain} & 0.6 & 0.3 \\ \text{worsen} & 0.4 & 0.4 \end{array}$$

Which options should you definitely not consider? First consider interval dominance, then consider maximality. Which of these two criteria gives the better answer?

Exercise 4

The following is again the very simple example, solved in Python, for Γ-maximin, Γ-maximax, interval dominance, and maximality, respectively:

```
>>> from improb.lowprev.lowpoly import LowPoly
>>> from improb.decision.opt import *
>>> lpr = LowPoly(2, credalset=[[.5, .5], [.8, .2]])
>>> gambles = [[440, 260], [420, 300], [370, 370]]
>>> opt = OptLowPrevMaxMin(lpr)
>>> list(opt(gambles))
[[370, 370]]
>>> opt = OptLowPrevMaxMax(lpr)
>>> list(opt(gambles))
[[440, 260]]
>>> opt = OptLowPrevMaxInterval(lpr)
>>> list(opt(gambles))
[[440, 260], [420, 300], [370, 370]]
>>> opt = OptLowPrevMax(lpr)
>>> list(opt(gambles))
[[440, 260], [420, 300], [370, 370]]
```

Solve Exercises 1, 2, and 3 similarly.
[Hint: The first argument of `LowPoly` denotes the number of events.]

Exercise 5 (*)

Let g be any gamble on $\mathcal{X}$, with lower prevision L and upper prevision U. Let c be any constant. Suppose you have the choice between the uncertain gain g, or the certain gain c.

Under each of the criteria, determine which of g or c (or both!) are optimal, under the following circumstances:

- $c < L$
- $L \leq c \leq U$
- $c > U$

In Exercise 3, option 4 corresponds to an investment without risk, as it yields the value $c = 35$ independently of the market, however, we found that this value was too low relative to the other options to be optimal.

For what values for c would you change your mind? Again, investigate this using each of the criteria.

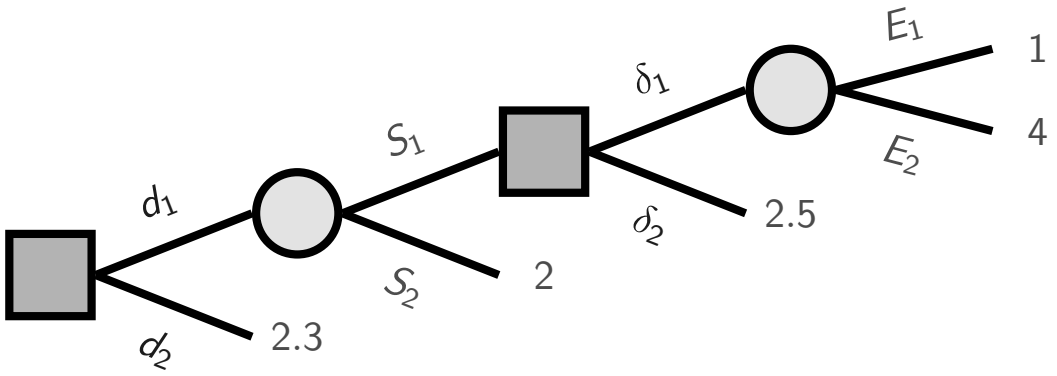
Exercise 6

State one advantage, and one disadvantage, of solving a sequential decision problem by normal form backward induction, compared to solving it by normal form.

Can you think of a situation in which normal form backward induction would be less efficient than normal form?

Exercise 7

Solve the following sequential decision problem for maximality, using either normal form, or normal form backward induction.



$$\mathcal{P} = \begin{array}{c|cc} & p_1 & p_2 \\ \hline S_1E_1 & 0.2 & 0.1 \\ S_1E_2 & 0.3 & 0.4 \\ S_2 & 0.5 & 0.5 \end{array}$$

$$\left[\text{Hint: } \mathcal{P}|S_1 = \begin{array}{c|cc} & p_1 & p_2 \\ \hline E_1 & 0.4 & 0.2 \\ E_2 & 0.6 & 0.8 \end{array}\right]$$

Exercise 8 (*)

The following program solves the previous exercise in Python for maximality, by normal form backward induction:

```
>>> from improb.lowprev.lowpoly import LowPoly
>>> from improb.decision.opt import *
>>> from improb.decision.tree import *
>>> pspace = PSpace(["S1", "S2"], ["E1", "E2"])
>>> E1 = pspace.make_event(["S1", "S2"], ["E1"], name="E1")
>>> E2 = pspace.make_event(["S1", "S2"], ["E2"], name="E2")
>>> S1 = pspace.make_event(["S1"], ["E1", "E2"], name="S1")
>>> S2 = pspace.make_event(["S2"], ["E1", "E2"], name="S2")
```

(continued on next slide)

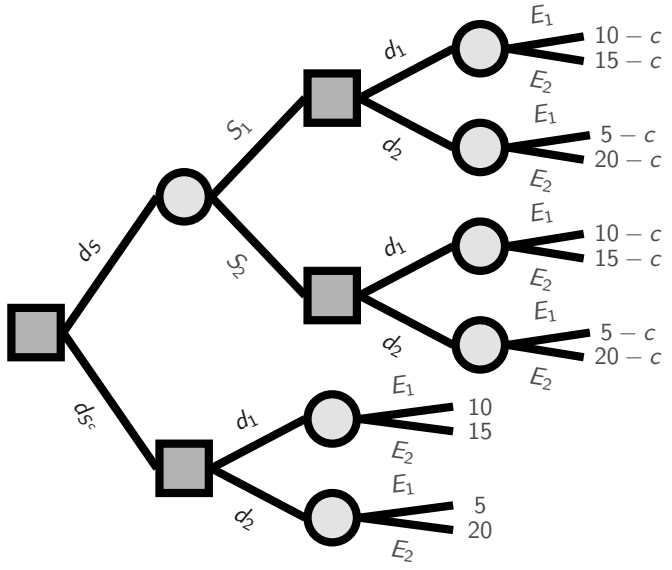
(continued from previous slide)

```
>>> t3 = Chance(pspace)
>>> t3[E1] = 1
>>> t3[E2] = 4
>>> t2 = Decision()
>>> t2["delta1"] = t3
>>> t2["delta2"] = 2.5
>>> t1 = Chance(pspace)
>>> t1[S1] = t2
>>> t1[S2] = 2
>>> t0 = Decision()
>>> t0["d1"] = t1
>>> t0["d2"] = 2.3
>>> print(t0)
>>> lpr = LowPoly(pspace)
>>> lpr.set_lower(S1 & E1, 0.1)
>>> lpr.set_lower(S1 & E2, 0.3)
>>> lpr.set_precise(S2, 0.5)
>>> opt = OptLowPrevMax(lpr)
>>> for gamble, normal_form_decision in t0.get_norm_back_opt(opt):
...     print(normal_form_decision)
```

Verify the solution. For what value of d_2 does d_2 become uniquely maximal? If you found this easy, also solve Exercises 9, 10, and 11, using Python.

Exercise 9 (*)

Tomorrow, a subject is going for a walk in the lake district. It may rain (E_1), or not (E_2). The subject can either take a waterproof (d_1), or not (d_2). But the subject may also choose to buy today's newspaper, at cost c, to learn about tomorrow's weather forecast (d_S), or not (d_{S^c}), before leaving for the lake district. The forecast has two possible outcomes: predicting rain (S_1), or not (S_2). *Solve for maximality, with $c = 1$.*



$\mathcal{P} =$

	p_1	p_2	p_3	p_4
S_1E_1	0.378	0.378	0.378	0.478
S_1E_2	0.162	0.162	0.262	0.162
S_2E_1	0.072	0.172	0.072	0.072
S_2E_2	0.388	0.288	0.288	0.288

Exercise 10 (**)

Consider again the lake district exercise.

For which values of c is it no longer maximal to buy the newspaper?

(This is the value of information of the newspaper.)

Exercise 11 (*)

Complete the details of the oil wildcatter example which we discussed during the lecture, by normal form backward induction, and thereby verify the solution.

(If you have Python, Octave, or Matlab, you can also try to verify the solution by normal form.)