NOPT042 Constraint programming: Tutorial 6 – Scheduling

What was in the lecture? Nothing, the lecture was canceled.

Exercise:

- Explain why consistency techniques cannot destroy solutions
- Give an example of an instance where
  - not node consistent
  - node consistent but not arc consistent
  - arc consistent but not solvable

In [1]:
```
%load_ext ipicat
```

```
Picat version 3.9
```

In [2]:
```
%%picat
import cp.
main => not_nc(X), solve([X]).

not_nc(X) =>
    X :: 0..9,
    X #> 2,
    X #< 8.

nc_not_ac(Y,Z) =>
    [Y,Z] :: 0..9,
    Y + Z #= 5.

ac_not_solvable(U,V,W) =>
    [U,V,W] :: 0..1,
    U #!= V,
    V #!= W,
    W #!= U.
```

Beware of time2/1!

In [3]:
```
%%picat
import cp.
main =>
    pigeonhole(L),
    time(solve(L)), print(L).
    % time2(solve(L)), print(L).

pigeonhole(L) =>
```

```
    L = new_list(11),
    L :: 1..10,
    all_different(L).
```

```
*** error(failed,main/0)
```

Scheduling

Exercise: moving

A simple scheduling problem: Four friends are moving. The table shows how much time and how many people are necessary to move each item. Schedule the moving to minimize total time. (Adapted from R. Barták's tutorial; check the SICStus Prolog model.)

Item	Time (min)	People
piano	45	4
chair	10	1
bed	25	3
table	15	2
couch	30	3
cat	15	1

In [4]:
```
!cat moving/instance.pi
```

```
instance(NumPeople, Items, Duration, People) =>
    NumPeople = 4,
    Items = ["piano", "chair", "bed", "table", "couch", "cat"],
    Duration = [45, 10, 25, 15, 30, 15],
    People = [4, 1, 3, 2, 3, 1].
```

In [5]:
```
#!cat moving/.solution/moving.pi
```

How to improve the model?

The `cumulative` global constraint

For the above problem we can use the following global constraint:

```
cumulative(StartTimes, Durations, Resources, Limit)
```

which means that we have `Limit` of resource available, each item starts at `StartTimes[i]`, takes `Durations[i]` time and consumes `Resources[i]` of the resource.

More about the constraint `cumulative`:

- Picat on GitHub (unofficial)
- Global Constraint Catalog: the cumulative constraint - see the references

Exercise: Meeting scheduling

We have a group of agents. We are given a set of meetings, where each meeting has a location, a duration, and a subset of agents that must attend that meeting. Moreover, we are given a distance (in time units) between each pair of locations.

Our goal is to schedule the meetings into time slots such that each agent can attend all the meetings. (The agents can start and finish at any location.) We want to minimize the end time (when all meetings have concluded).

See https://www.csplib.org/Problems/prob046/ for a description of the problem and more details, including a sample instance.

In [6]:
```
!cat meeting-scheduling/instance.pi
```

```
instance(NumMeetings, NumAgents, MeetingAttendees, MeetingDurations, Distances) =>
    NumMeetings = 5,
    NumAgents = 3,
    MeetingAttendees = [
        [1, 2],
        [2, 3],
        [1, 2, 3],
        [3],
        [1]
    ],
    MeetingDurations = [1, 2, 1, 3, 2],
    Distances = {
        {0, 1, 2, 1, 3},  % Distances from Meeting 1
        {1, 0, 3, 2, 2},  % Distances from Meeting 2
        {2, 3, 0, 1, 2},  % Distances from Meeting 3
        {1, 2, 1, 0, 3},  % Distances from Meeting 4
        {3, 2, 2, 3, 0}   % Distances from Meeting 5
    }.
```