Knowledge representation (INFO0049-1)

Exercise session 8

21 Apr 2015

1. Palm tree problem

The garden of our hotel on Cyprus has beautiful palm trees, and they need water every morning. The gardener is a nice man who can follow and interpret orders.
He must start every morning exactly at 6, and he has to give each palm tree the exact amount of water prescribed by the manager. She usually is very precise, but she has not realized how much freedom her orders really leave the gardener. Indeed, in an attempt to make sure that he is never idle, she has set up a set of rules he must obey. But before showing you the rules, here is how watering of the palm trees proceeds: the gardener fills his bucket with water at the well, walks to a palm tree, pours the required amount of water, walks to another palm tree, pours, walks, . . . until his bucket is empty; he then walks back to the well, fills the bucket, walks to a palm tree, and so on until all palm trees are watered. He then walks back to the well and his watering job is done. These are the rules the gardener must obey:

Going from one tree to another or to or from the well must be done in straight lines, and the gardener is to walk at a steady pace. We will model this by just giving you the time it takes to walk from one point to another.
The bucket must be empty when returning to the well, except perhaps on the last return.
The bucket must always be filled completely at the well, and this takes a fixed amount of time.
Each tree must be visited exactly once.
The gardener is not doing anything else but filling, pouring and walking.

The gardener knows that when he is finished watering the palm trees, he will have to start digging holes for new trees. He prefers watering over digging, so he exploits the rules to make the watering job as long as possible (of course within the rules set by the manager). This is not such a straightforward task, though, so he needs your program to help him. Your program can use given facts representing the bucket size, the distances between palm trees and the well, and the needs of each palm tree. Here is an example (from which you should be able to make a correct generalization):

```
% list of palm trees

palm(jasmine).
palm(sheherazade).
palm(nina).
palm(elisa).
```

```
% distances between palm trees

palm2palm(jasmine,sheherazade,18).
palm2palm(sheherazade,nina,12).
palm2palm(jasmine,nina,19).
palm2palm(elisa,nina,8).
palm2palm(elisa,sheherazade,20).

% distances from palm trees to well

palm2well(jasmine,13).
palm2well(sheherazade,19).
palm2well(nina,22).
palm2well(elisa,34).

% amount of water per tree

palm_needs(jasmine,2).
palm_needs(sheherazade,1).
palm_needs(nina,2).
palm_needs(elisa,4).

% bucket size

bucket(5).
```

Do not worry about non-Euclidean distances here. The garden has obstacles like swimming pools for the guests, a bar, flowerbeds, etc., and it is not always possible to walk from one tree to the other. You might have noticed that the gardener gives girls' names to his trees.
Write a predicate palmtree/1, which unifies its argument with an optimal compliant palm tree trajectory, or fails if no such trajectory exists. A palm tree trajectory is just a sequence of palm tree names, e.g. [nina,jasmine,elisa,nina] . A palm tree trajectory is compliant if the gardener could visit all the palm trees in the order of the trajectory, while obeying the watering prescriptions from his manager. The previous example is clearly not compliant, since it visits nina twice. The following is a compliant trajectory: [sheherazade,jasmine,nina,elisa]. A compliant trajectory is optimal if the time it takes for the gardener to follow the trajectory during watering is maximal. All distances, durations and water quantities are integers.

---

2. TV problem

I am a TV addict and I have a television in each of the four corners of my living room. They are all on all the time. Wherever I sit, I can always see (at least) one TV. I do not care what I watch, but I want all four TVs to show the same channel. Unfortunately, I can only receive five channels, named 1,2,3,4

and 5. When I wake up in the morning (after falling asleep on the TV room table while watching some late night show), the first thing I want to do is set all TVs to the same channel, not caring which channel. Of course, I use my remote control for this, but it is harder than it sounds. During the night, all TVs switch randomly to one of the five channels. And my remote control only has a next button with which I can choose the next channel (modulo 5). On top of that, my TVs refuse to change channel twice in a row, so, in order to change one particular TV twice, I have to change channels on at least one other TV in between. And this while I am in a hurry to make all TVs play the same channel.
Write a predicate remote/2, which has as its first argument a list of the channels for the four TVs (i.e. a list of four integers between 1 and 5), and which unifies its second argument with a list of TVs I should change consecutively to achieve my goals. Moreover, this list should have minimal length.

For example:

```
?- remote([3,1,3,3],L).
L = [1,2,3,2,4,2]
Yes
```

---

3. Basic search strategies
