Lab 2 - Hashing States – Artificial Intelligence

Teacher: Stephan Schiffel

January 13, 2017

Note: For this lab, you can work together in teams of up to 4 students. However, this is not a necessity. The assignment is small enough to do it alone in which case you may get more experience.

You will need a Java Development Kit (JDK) and a Java IDE (any text editor should do as well).

Time Estimate

1 hour in addition to the time spend in the lab class assuming you understand how a hash map works.

Tasks

1. (3 points) Many environments have the property that there are several paths that lead to the same state. Which of the search algorithms that we covered in class (breadth-first, depth-first and uniform-cost search) benefit from detecting repeated states? For each algorithm give a short explanation how detecting repeated states can make it better.

2. (2 points) Which data structure is typically used for detecting repeated states and why?

3. (12 points) Download the material below, implement a hash function for State objects that passes all the tests.

4. (2 points) Report and comment on the results you get. How many hash collisions and bucket index collisions do you get? Is this good or bad?

5. (1 point) Why is it important to have few hash collisions if hashing is used to detect repeated states?

Material

The Java project for the lab can be found on Skel. See, instructions below on how to get it.

The project contains a State class that could be thought of implementing a state of some robot. The state consists of a position, an orientation and a boolean holding the information of whether or not the robot is turned on. The project also contains code to test how good the hashCode() method of the State class is. To run the tests simply run the Main class or use “ant run” on the terminal.

Hints

- You need to implement the methods `int hashCode()` and `boolean equals(Object o)` in classes State and Position appropriately.
- Read the documentation for int hashCode() and boolean equals(Object o) methods to make sure you understand what properties they need to have!
- For two states s1 and s2 that you'd consider the same, s1.equals(s2) must return true and false otherwise.
- For any two objects o1 and o2, if o1.equals(o2) then o1.hashCode() == o2.hashCode() must be true!
- Objects that are not equal are allowed have the same hash code. However, if that happens too often then hash maps become inefficient.
- Delegate the computation of hashCode and checking of equality when possible! E.g., State should check for the equality of the positions by using Position's equal method.

Handing In

You must hand in this assignment through Skel (skel.ru.is) or it will **not** be graded. Only one member of the team should hand in the assignment. During submission you need to provide the RU usernames (skel logins) of all team members.

Connect to Skel using your favorite ssh client and unpack the assignment into your home directory by running the following commands:

```
[student14@skel ~]$ tar xvf /labs/arti17/lab2/lab2.tar
[student14@skel ~]$ cd arti17/lab2
[student14@skel lab2]$ ls
answers build.xml dist Makefile questions src
```

You can copy the code to your own machine for development by using any SCP or SFTP client (e.g., WinSCP). However, you need to copy it back to skel into the same place before handing in.

To answer the questions run `make answers` while being in the directory containing "Makefile". Single answers will be put into files called "answers/answerX.Y.txt", which you can also edit. To go back to a particular question run the following commands with `X.Y` being replaced by the number of the question.

```
[student14@skel lab2]$ cd questions
[student14@skel questions]$ ./questionX.Y.sh
```

Finally, to handin your answers run `make handin` while being in the directory containing "Makefile". This should produce a file "/labs/arti17/.handin/lab2/student14/handin.tar.gz". You can check if it exists using the ls command.