AP Computer Science
Dr. Paul L. Bailey

Responses 0403
Thursday, April 2, 2020

Question 1. What is the function of hashcode method?

Answer. The main point is this: if two objects are equal, then their hashcodes should be equal. Thus, it is considered "best practice" to override the hashcode method whenever you override the equals method. Among other things, the hashcode can be used as a quick check to rule out equality.

For example, suppose we have a list of long arrays of integers. If we wanted to check if any of them equals another array, we could first check their lengths. If the lengths are different, there is no need to loop through the whole array and compare every entry. So, by checking the length first, we saved a lot of processing time.

The hashcode of an object is an integer which should be relatively easily computed. Standard library methods can then use the hashcode to group your objects, and potentially rule out that they may be equal.

Stack Overflow is useful for this sort of thing:

`https://stackoverflow.com/questions/3563847/what-is-the-use-of-hashcode-in-java`

□

Question 2. I finished the method, but I have a few clarifying questions. If a Walk is defined by three consecutive vertices does that mean 1-¿2-¿4 or something like that in increasing order? Or does it mean for any 3 vertices if there exits edges between them there is a walk? So for example would 4-¿1-¿3 be considered a walk? Lastly, will we ever have to write code to determine the length of some walk in the graph?

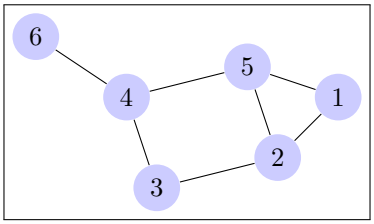
Answer. A walk is a finite sequence of vertices such that any two adjacent vertices in the sequence are involved in the same edge. So $(4, 1, 3)$ would be a walk if and only if there is an edge between 4 and 1, and an edge between 1 and 3.

Yes, we will eventually want to write code to find the shortest walk between two vertices. □

Question 3. I am still kind of confused on 35b, what does the "Degree()" method do?

Proof. The *degree* of a vertex is the number of edges in which it is involved.

Consider the following graph:



Then

$$\deg(1) = 2, \deg(2) = 3, \deg(3) = 2, \deg(4) = 3, \deg(5) = 3, \deg(6) = 1.$$

□

Question 4. How do we use recursion in isConnected()?

Answer. Actually, it turns out you don't need recursion for adjacents, associates, or isConnected. □