lecture 35
graph applications
- garbage collection
- Google page rank

garbage collection

myDog = new Beagle ("Bob")
myDog = new Terrier ("Tim")

Nothing references Beagle Bob.
Bob is wasting memory.
Bob is garbage.

Memory of Java Virtual Machine

When the program starts

Stack space
Call stack has one frame (main)

Object space
no objects

As program runs ...

Eventually....

Call stack

Object space fills up
What to do when object space is full?

- let program crash
- reuse space we don't need anymore (garbage collection)

Garbage collection: two phases

1. identify non-garbage ("mark")
2. remove garbage ("sweep")

How to identify garbage?

An object is "live" (not garbage) if it is referenced:
- by a variable on the call stack
- by a field in a "live" object (recursion)

If these two objects are only referenced by each other, they both are garbage.

How to identify garbage objects?

Consider a graph:

\[ V = \{ \text{reference variables in call stack: "root" vertices} \} \]
\[ + \{ \text{objects: "non-root" vertices} \} \]
\[ E = \{ (\text{root vertex, non-root vertex}) \} \]
\[ + \{ (\text{non-root vertex, non-root vertex}) \} \]

Let each object have a visited bit (called a "mark" bit \( \equiv \) not garbage).

Initialize each object's visited to false.

For each root vertex \( r \), traverse the graph \( (V, E) \), visiting all objects reachable from \( r \)

if object is not visited by any of these traversals, then object is garbage.
How to remove garbage?

1. Identify non-garbage ("mark")
2. Remove garbage ("sweep")

Then, continue program execution

- new objects can be added, wherever there is a gap big enough (adjust the object vs. free lists)
- garbage collection needed again when there is no gap big enough for the new object
- program needs to stop (temporarily) to do garbage collection -> not good for important real-timestuff

Lecture 35

Graph applications
- Garbage collection
- Google page rank

What does Google do?
- Download all the reachable pages on web (web crawler)
- Build graph
  \[ V = \{ \text{web pages} \} \]
  \[ E = \{ (v, w) : v \text{ has link to } w \} \]
- Analyze the "importance" of each page ("page rank")

Google's Page Rank (Brin and Page 1998)

How "important" is a web page \( v \)?

Recursive concept of importance:
You are important if important people think you are important.

Concrete example: in academic research, a published paper is important if it is cited by many important published papers. (especially those important papers that cite relatively few papers)
How important is a web page \( v \)?

Define \( R(w) \) to be the pagerank of \( w \).
Let \( N_{out}(w) \) be the out-degree of \( w \), i.e., the number of outgoing edges.

\[
R(v) = \sum_{(w,v) \in E} \frac{R(w)}{N_{out}(w)}
\]

How to solve it?

- \( R(v) = \frac{1}{N_{out}(v)} \) in \( E \)
- Recursion, but no base case
- \( R^0(v) = \text{constant} \) (initial guess)
- \( R^k(v) = \sum_{(w,v) \in E} \frac{R(w)}{N_{out}(w)} \)
  and iterate until convergence

What does Google do?

- Download all the reachable pages on web
- Build graph
- Analyze the "importance" of each page ("page rank")

What does a user do?

- Enter query words and ask Google search engine for list of pages containing these words

To answer a user's query, Google needs to index words

Example

"Montreal"
"bicycle"
"store"

- Then, compute the intersection of these three sets, giving web pages that contain all three keywords
- Then, sort by pagerank \( R(v) \)

For more details, including the original research papers describing the method, see:

http://infolab.stanford.edu/