COMP 250

Lecture 32

Graph traversal: Applications

garbage Collection
(mark and sweep)

Google PageRank

Nov. 23, 2018

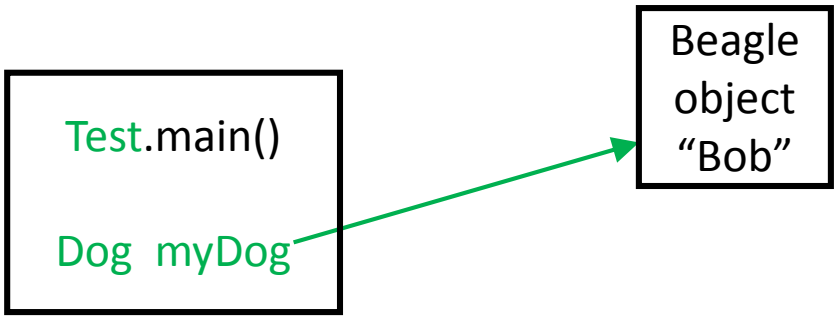
Garbage Collection

```
Dog   myDog  =  new  Beagle("Bob");

      myDog  =  new  Terrier("Tim");
```

Nothing references the Bob the Beagle.

Bob is wasting memory. Bob has become garbage.

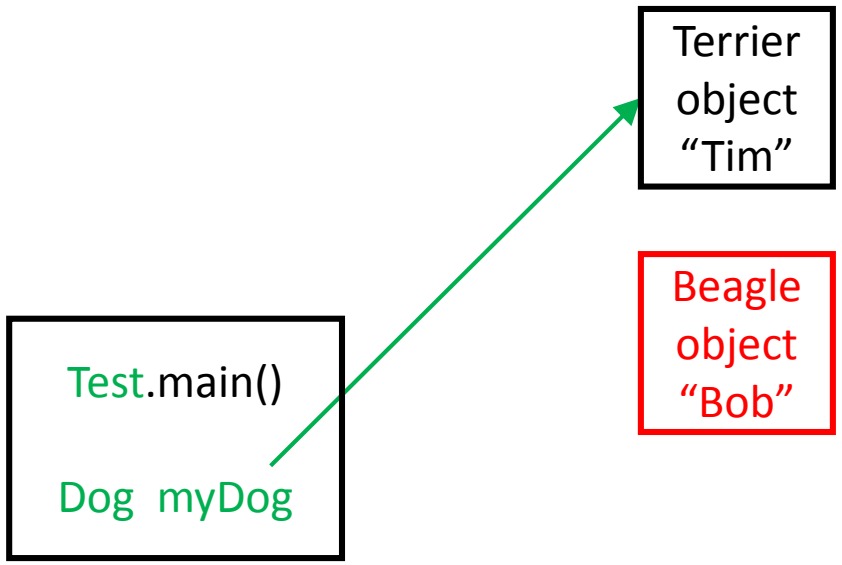
```
Dog    myDog = new Beagle("Bob");

       myDog = new Terrier("Tim");
```



Call Stack Heap

Dog myDog = new Beagle("Bob");

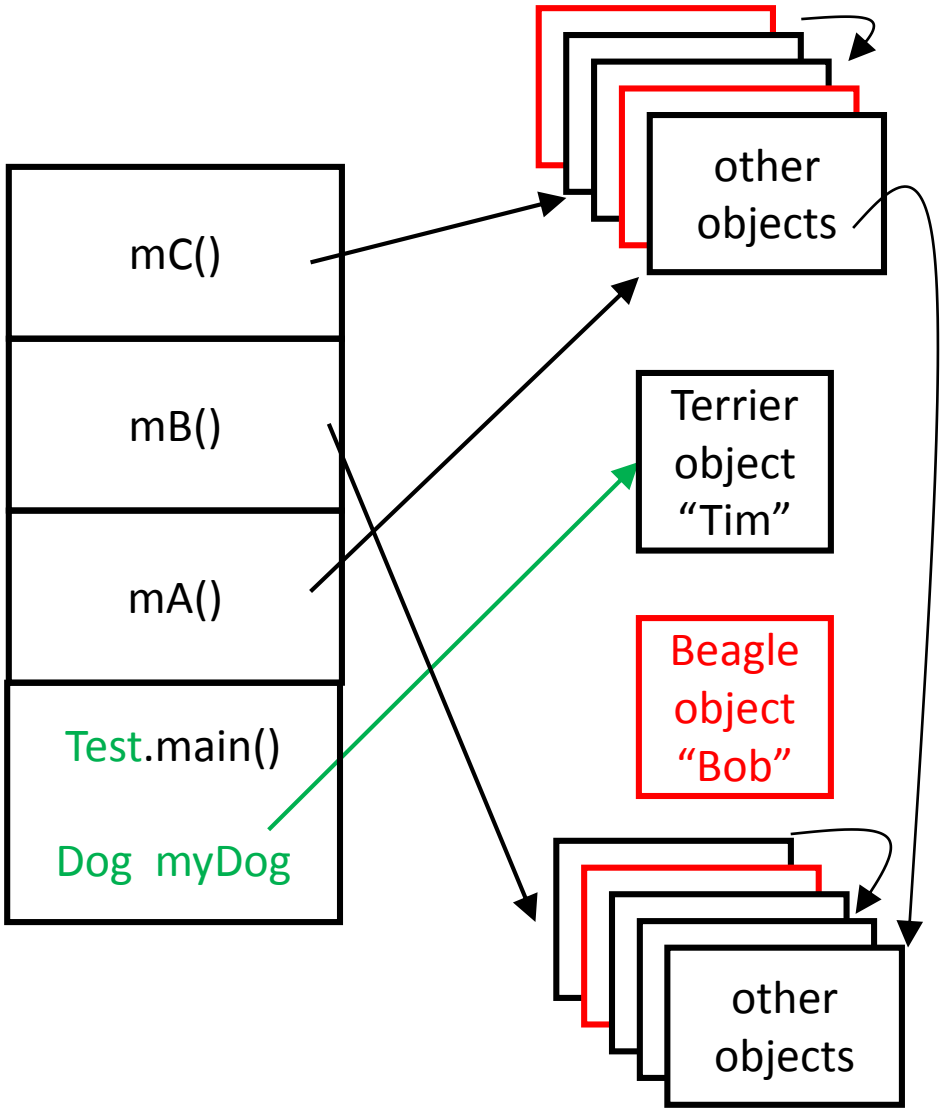
myDog = new Terrier("Tim");



Bob is garbage.

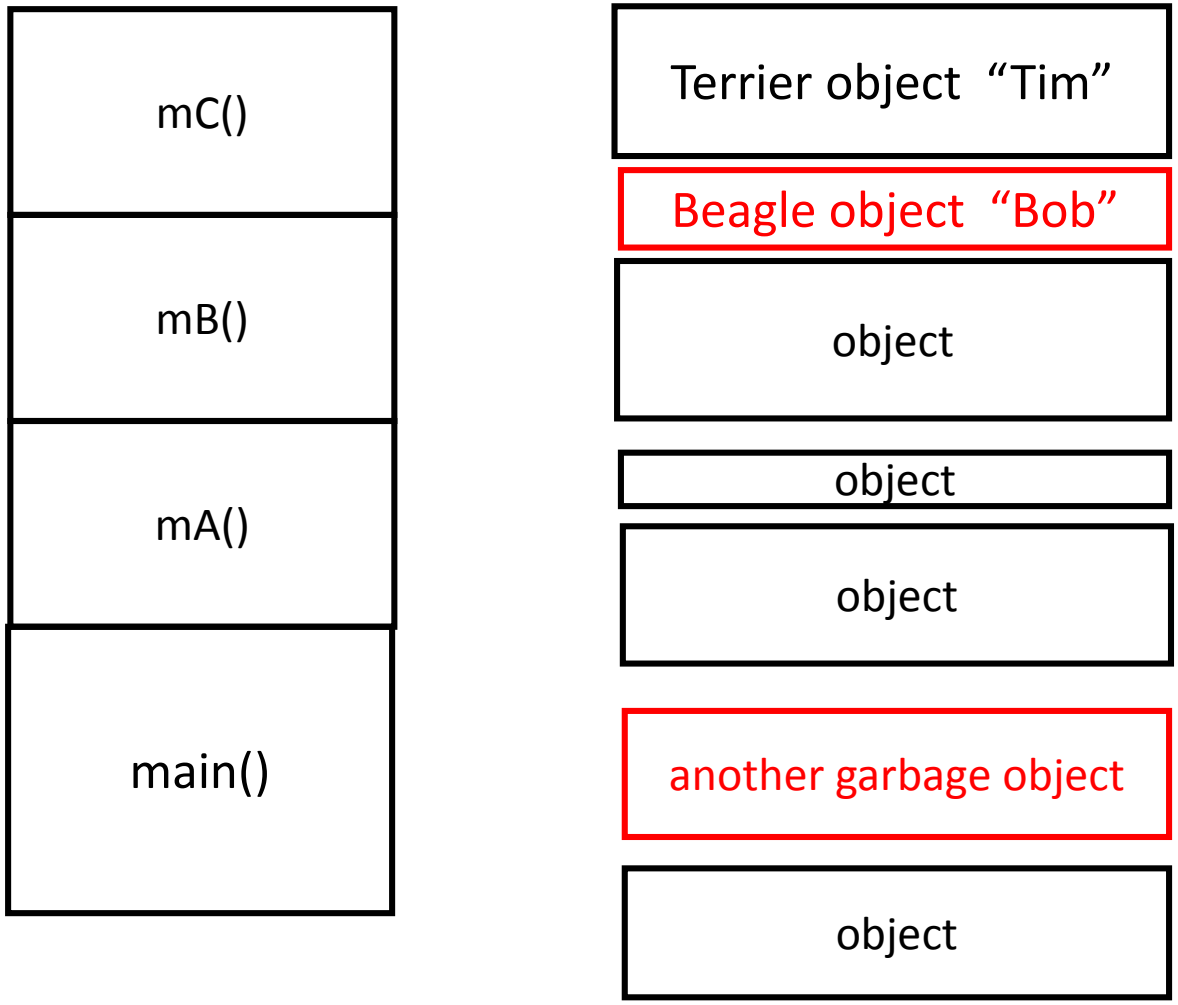
Call Stack Heap

As the program continues to run, more objects are created in the heap space, and more methods appear on the call stack.



Bob is garbage.

Every object has a location in memory: Object.hashCode().

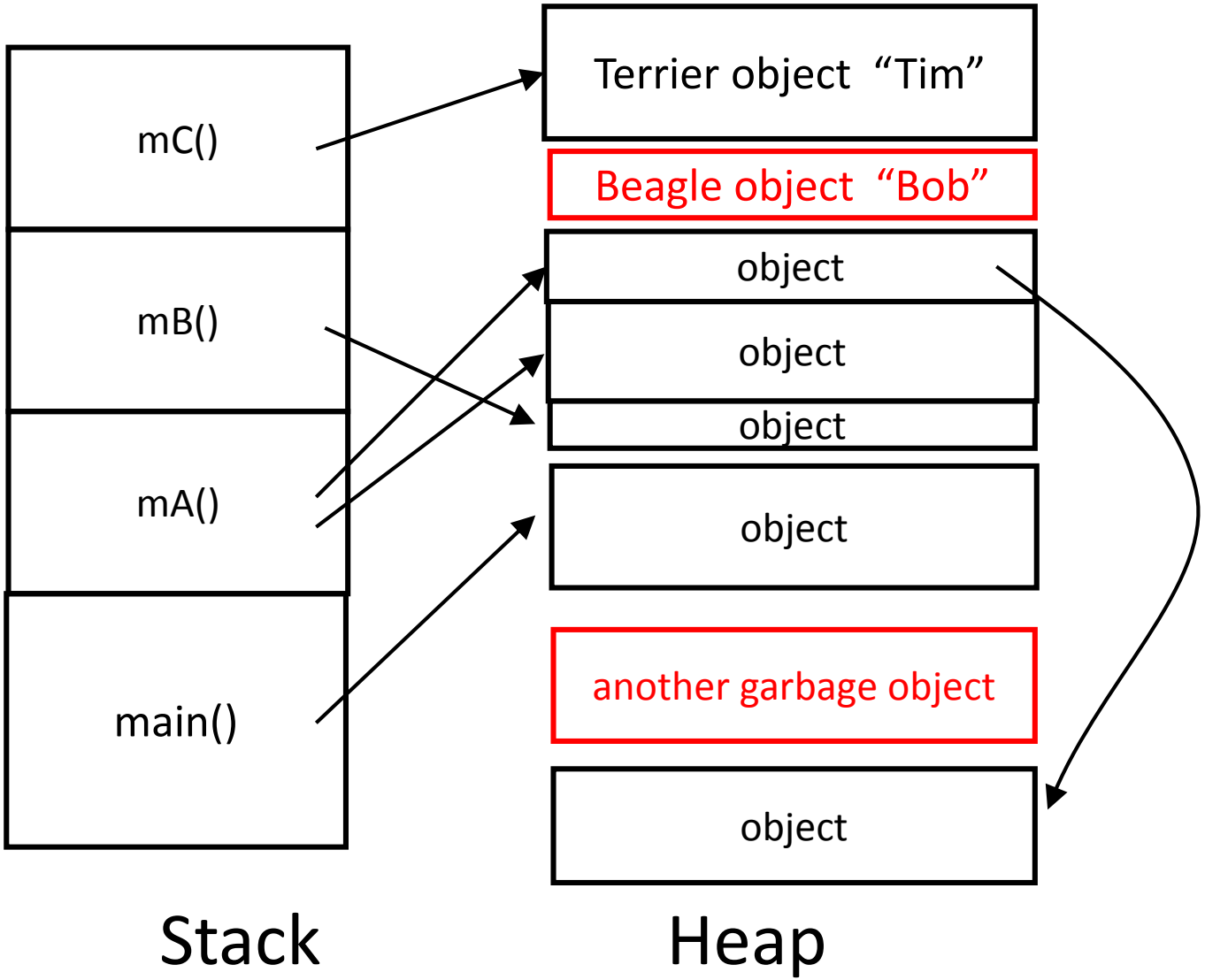


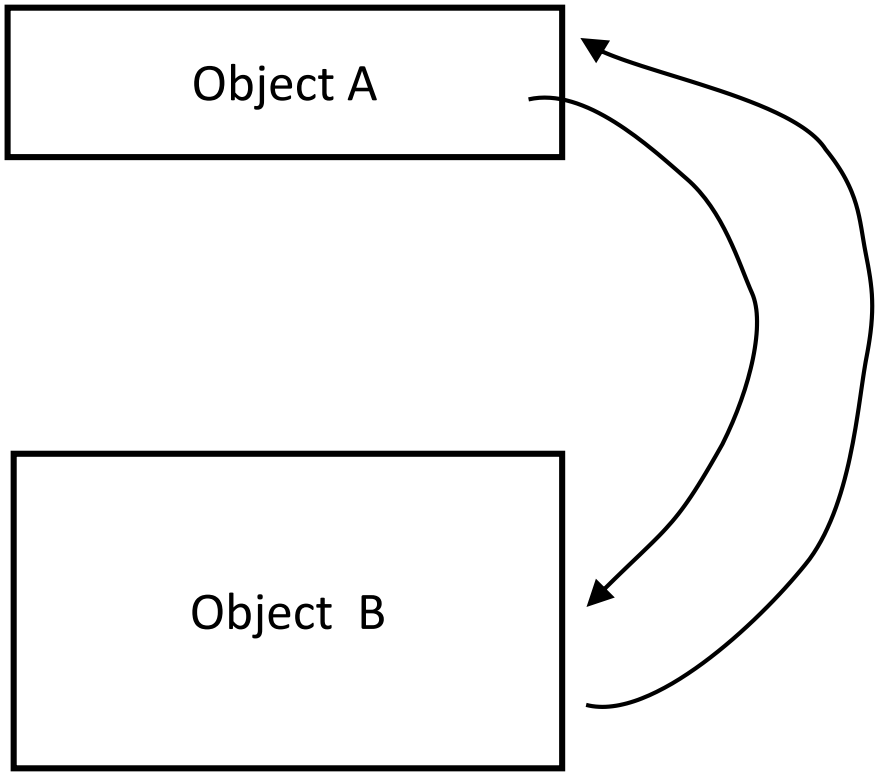
Q: What to do when heap space fills up?

A: Let the program crash.

A: Reuse the space we don't need.
(Garbage collection)

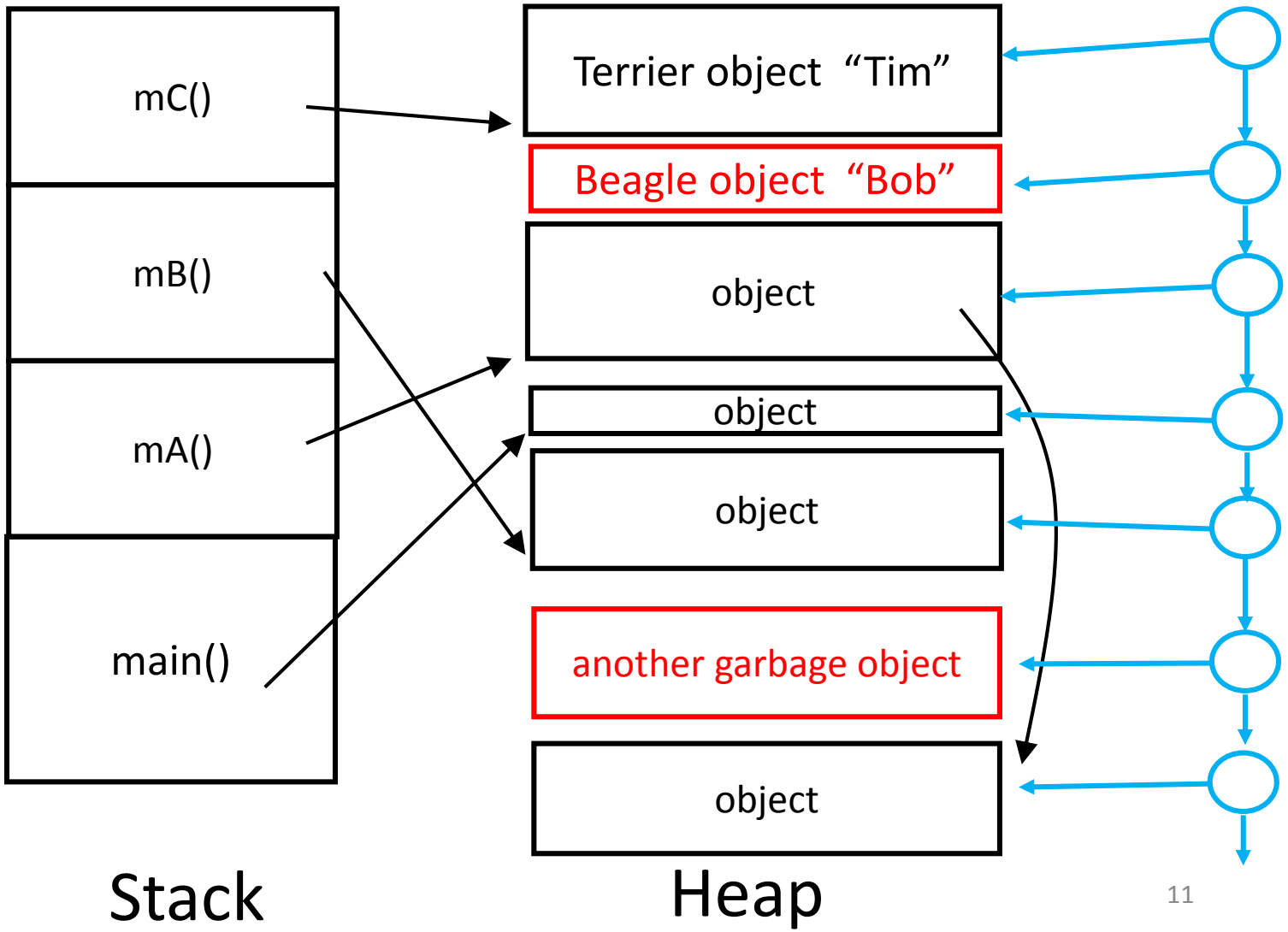
"Live objects" are those referenced either from a call stack variable or from an instance variable in a live object (recursive definition).





These objects are only referenced by each other. They are garbage, because they will never be used by the program.

The Java Virtual Machine (JVM) maintains a linked list of all objects. i.e. The list stores the Object.hashCode() of each object.



Garbage collection: “Mark and Sweep”

1) Build a graph, and identify live objects (“Mark”)

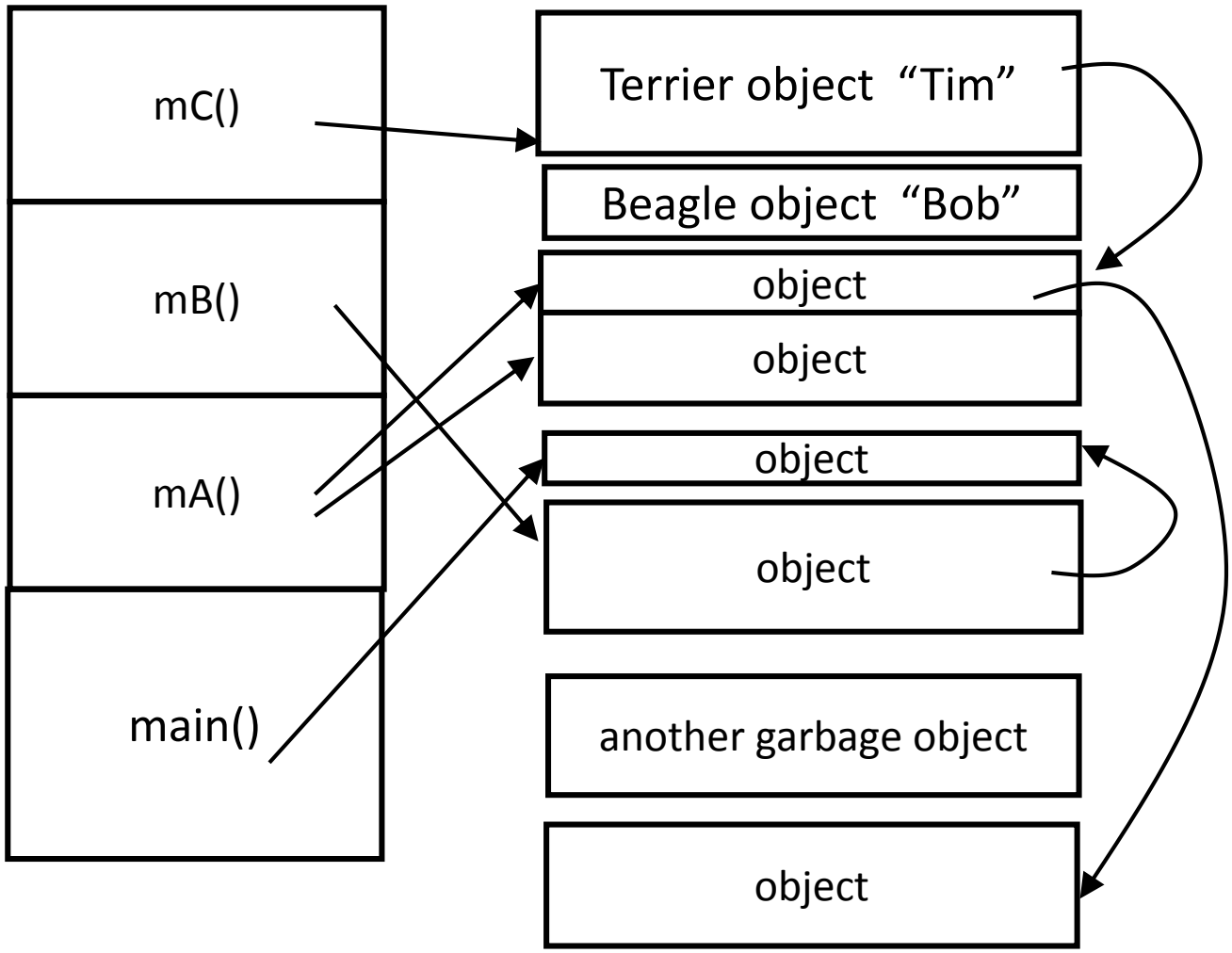
2) Remove garbage (“Sweep”)

1) Build a graph …

Vertices V *are*

- **reference variables** (in call stack)
- **objects** (in heap)

Edges E are references (arrows in figure)

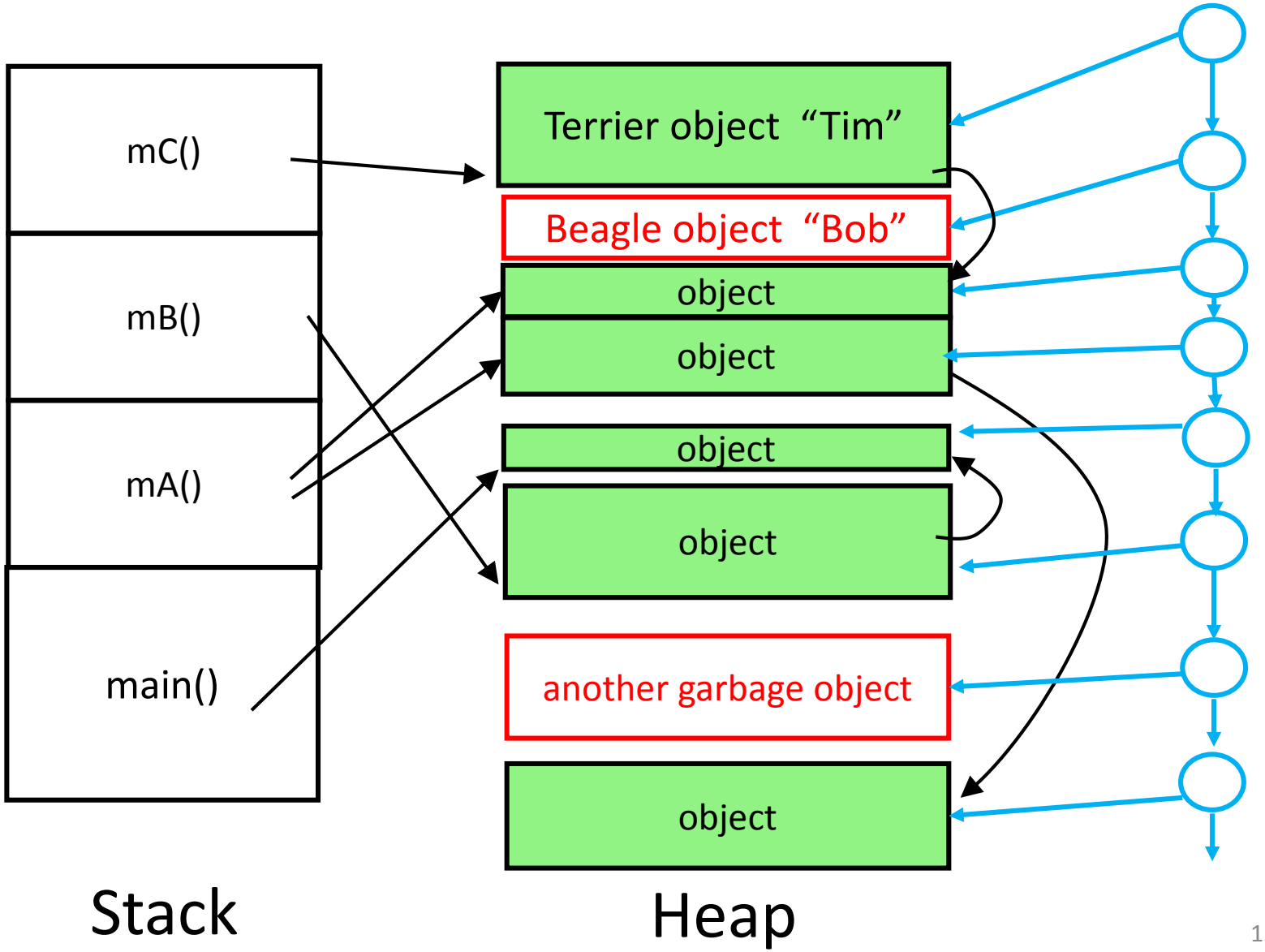


1) Build a graph … and mark live objects

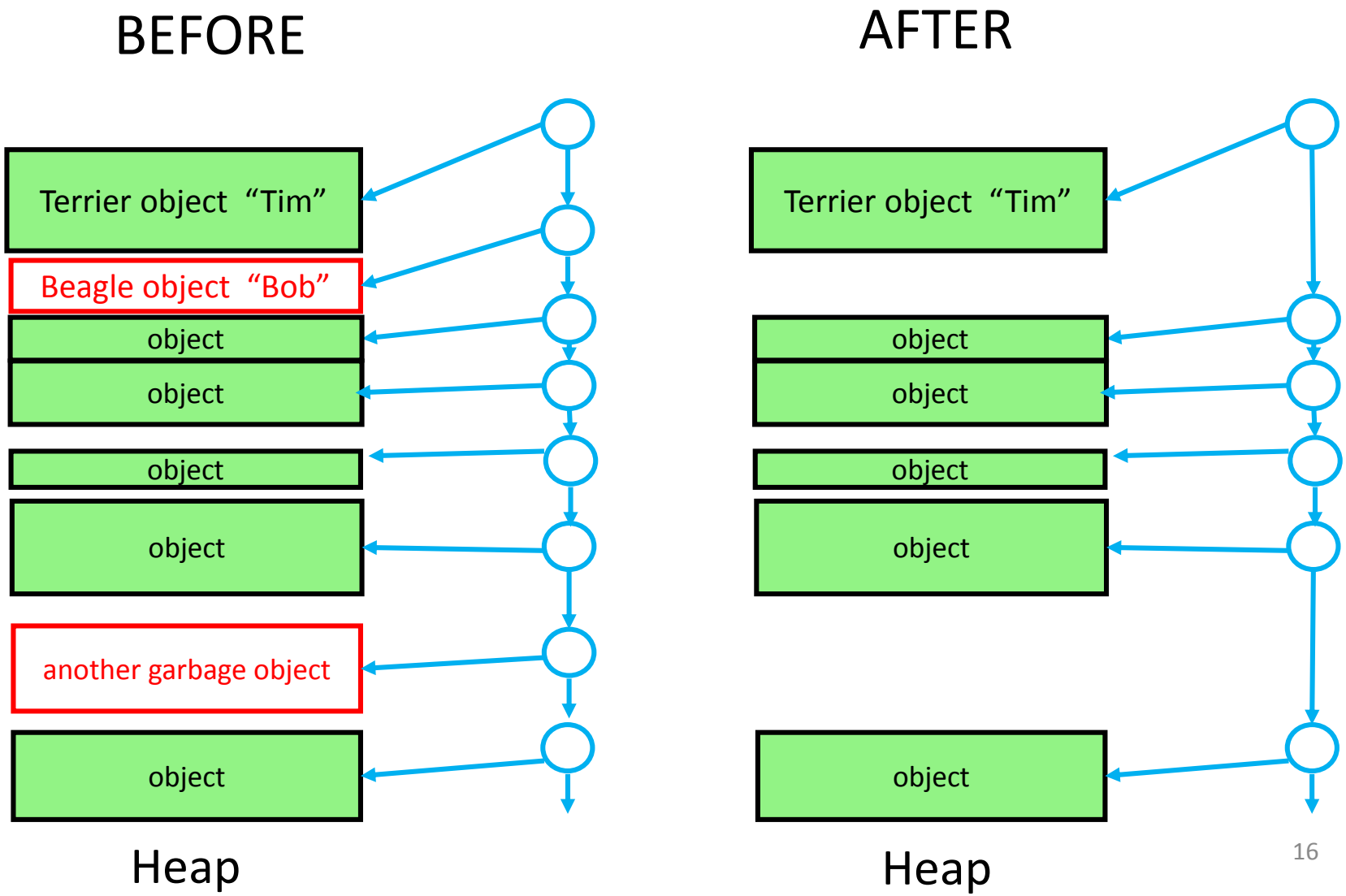
For each reference variable on the call stack {

- traverse the graph, starting from the object that is referenced by that variable, and mark each object that is reached as visited

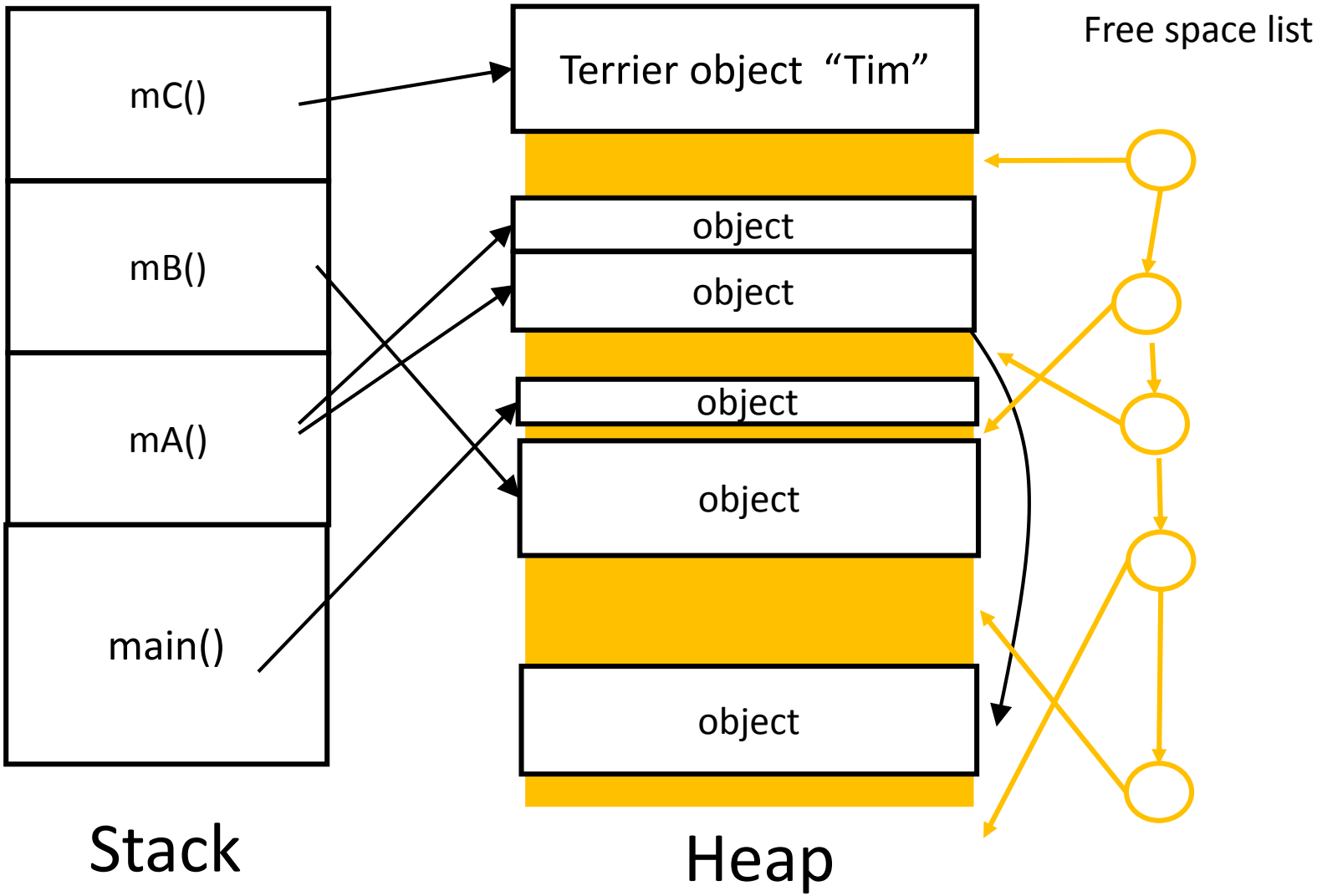
“Marked” = visited = reached



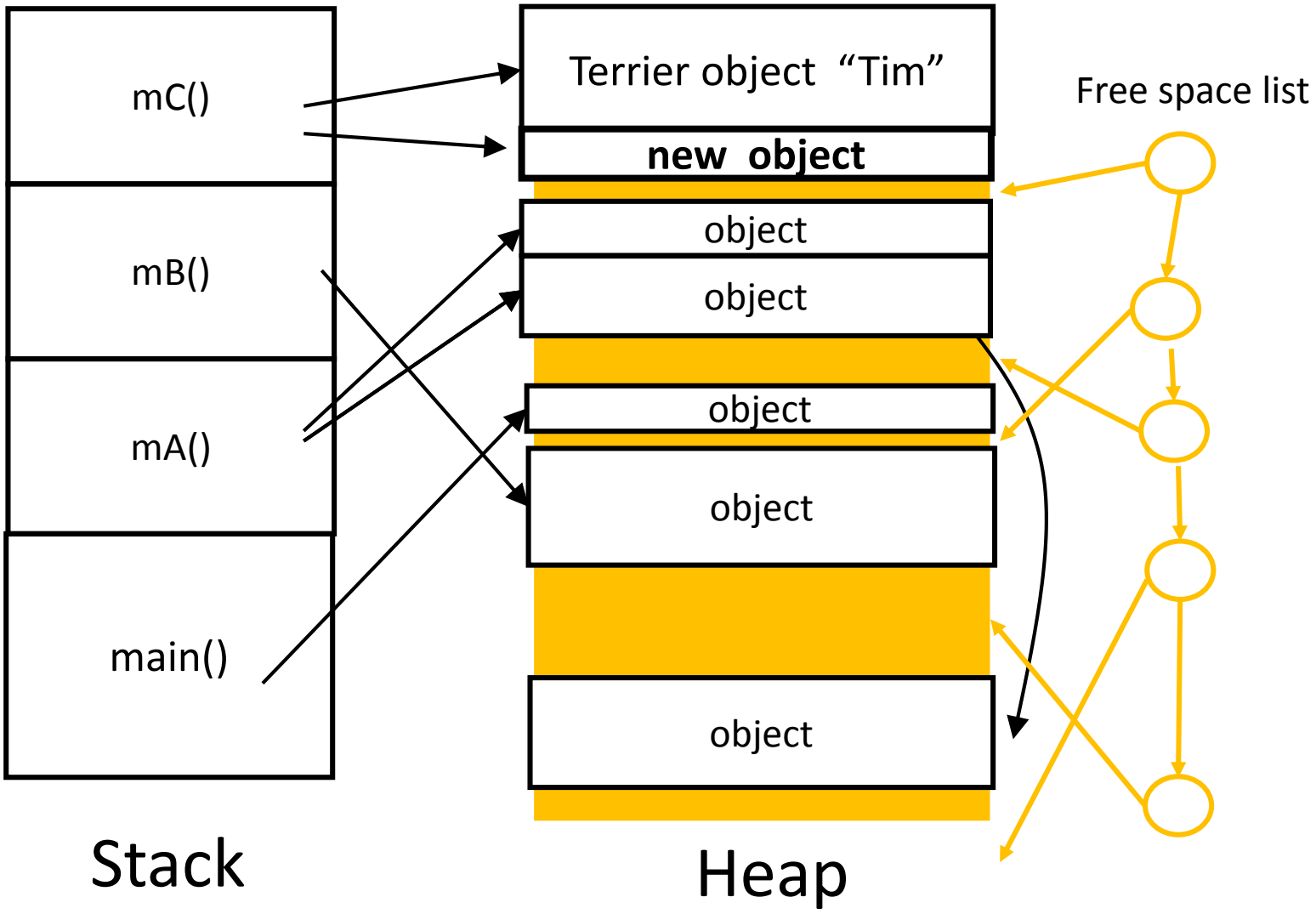
2: "Sweep" away the garbage



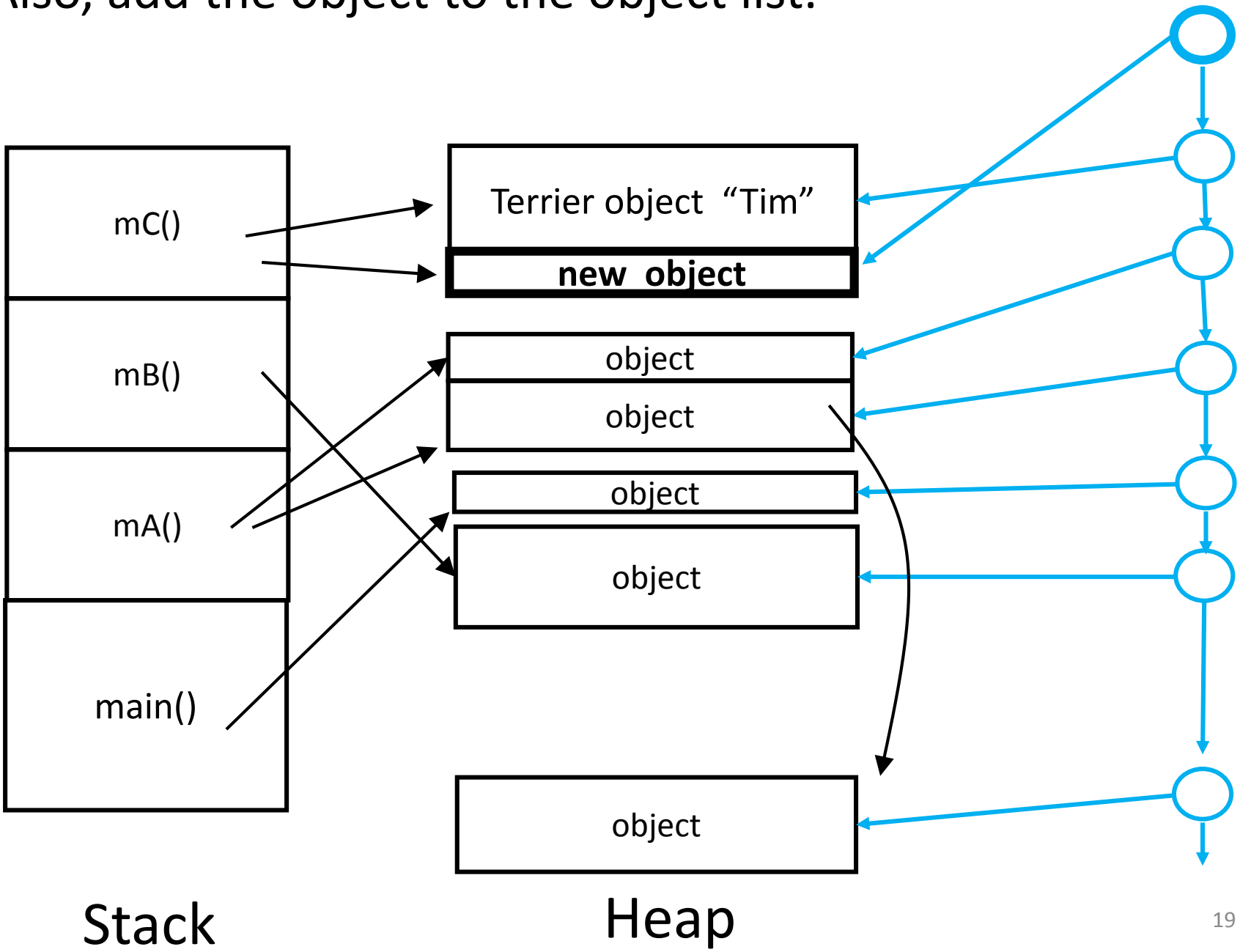
Use another list to keep track of free space between objects. Why?



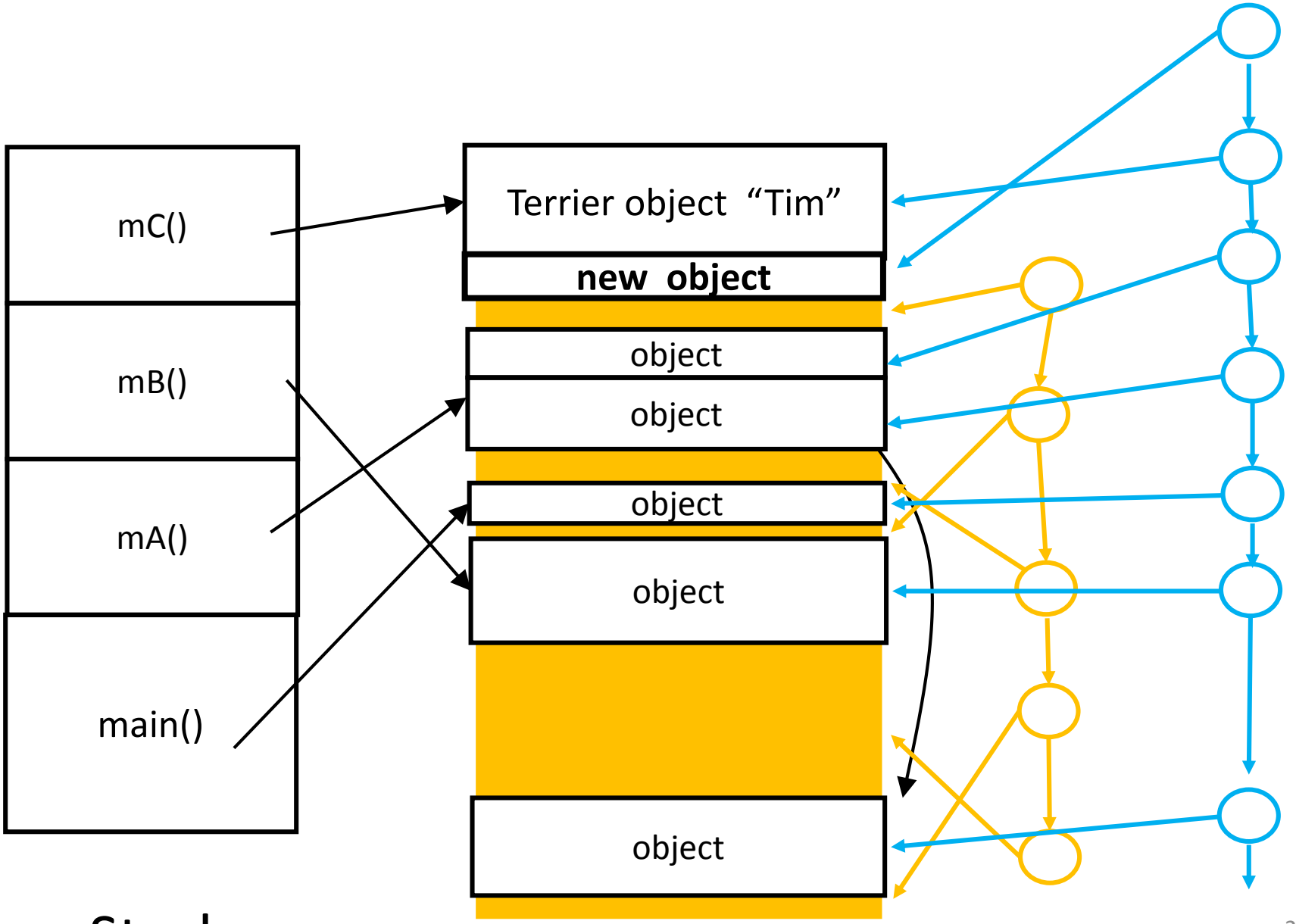
When a new object is created, we can find a place for it. We may need to modify the free space list.



Also, add the object to the object list.



Two lists: free space, live objects



After garbage collection, continue execution..

- New objects can be added, where there is a big enough gap in free space.

- Garbage collection is needed again when there is no gap big enough for the new object. (Objects can also be moved around – if heap space is too fragmented.)

- Program needs to stop (temporarily) to do garbage collection, which is not good for real time applications.

COMP 250

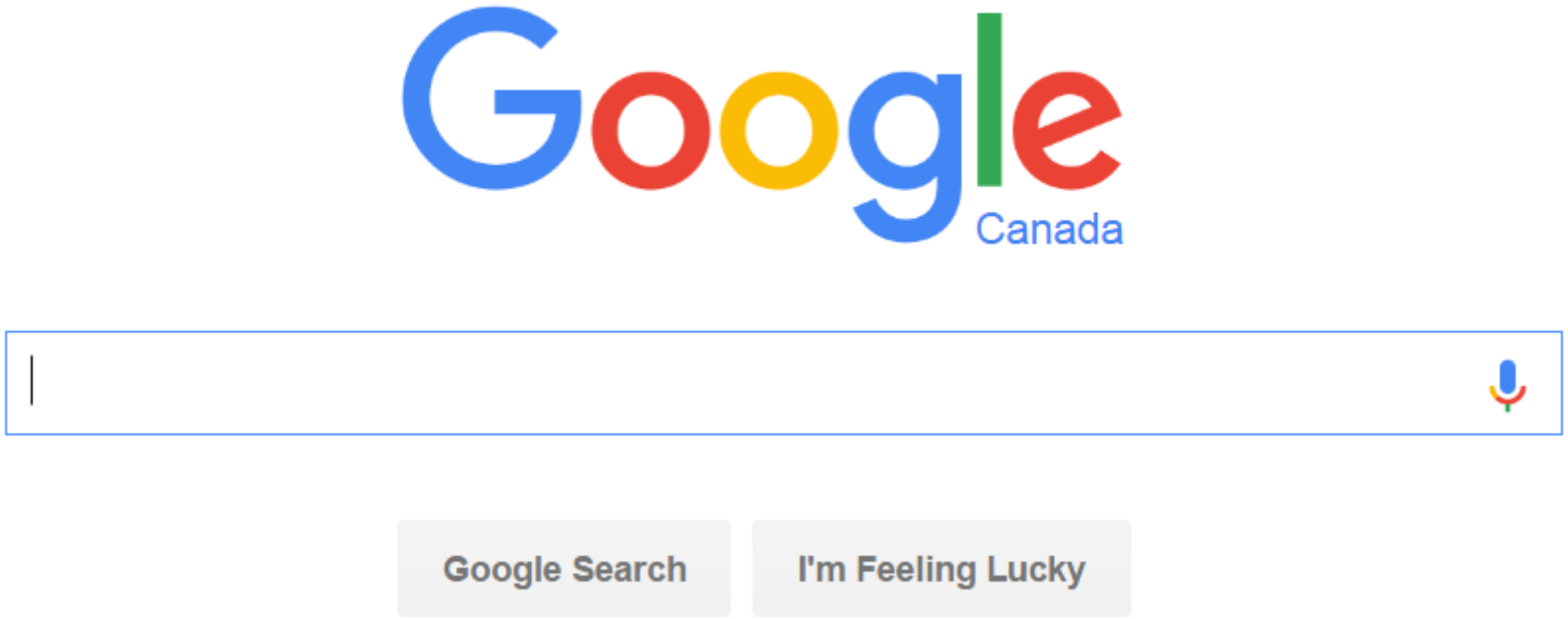
Lecture 32

Graph traversal: Applications

garbage Collection
(mark and sweep)

Google PageRank

Nov. 23, 2018


Google
Canada
Google Search
I'm Feeling Lucky

The set of web pages on the world wide web define a graph. Vertices are web pages. Edges are hyperlinks.

Google “crawls” the web (traversing this graph by following links), retrieves as many web pages as it can find.

The set of web pages on the world wide web define a graph. Vertices are web pages. Edges are hyperlinks.

Google "crawls" the web (traversing this graph by following links), retrieves as many web pages as it can find.

Google then builds a graph data structure:

- the vertices V are the web pages
- the edges E are the hyperlinks within the web pages
- the keys K for the vertex map are the URL's

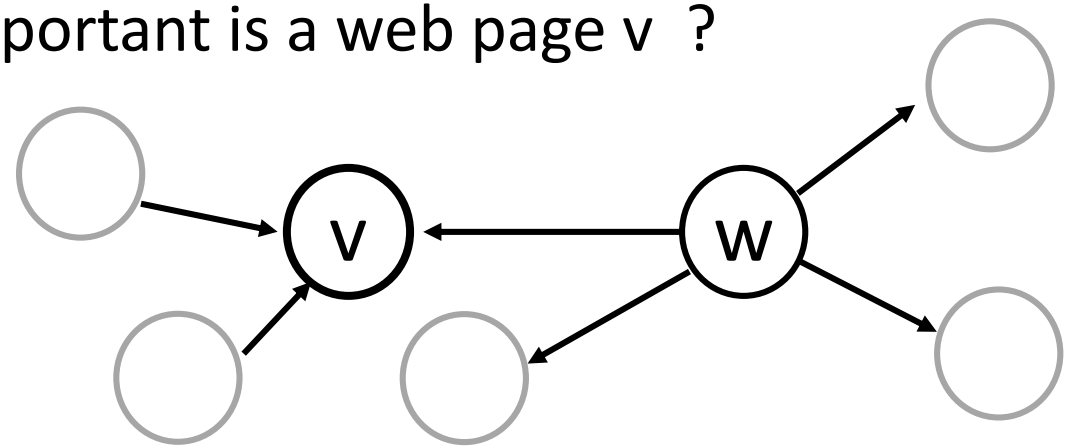
Google PageRank

When you ‘google’ a term, you want to find important web pages for that term (i.e. high rank).

A web page is important if … ?

Google PageRank

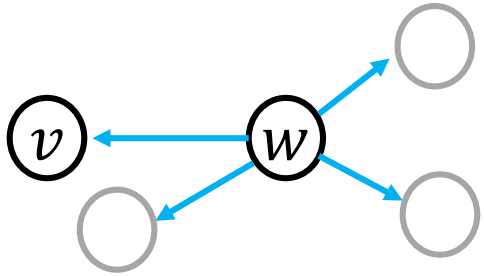
Q: How important is a web page v ?



A:

- Which set of pages { w } link to v, and how important are these pages?

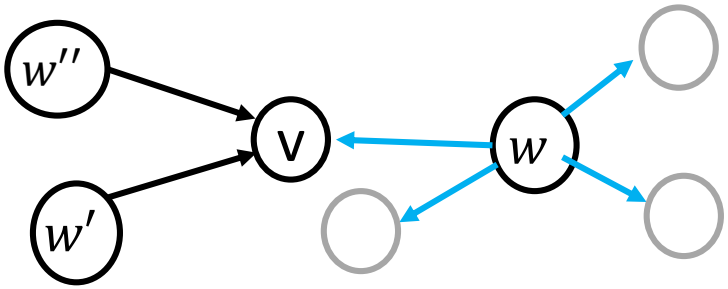
- How many other pages does each w point to ?



To define the “page rank” of v :

Let w be a vertex such that (w, v) is an edge.

Let $N_{out}(w)$ be the ‘out degree’ of *w*.

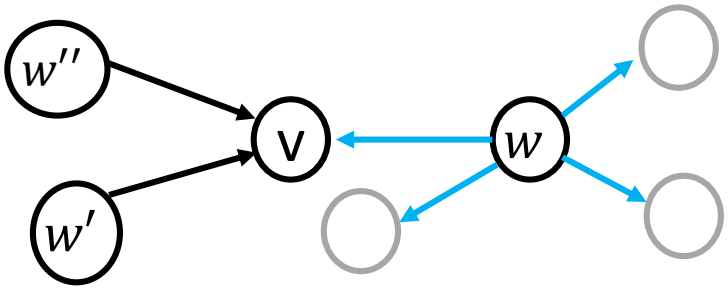


Define the PageRank of v:

$$R(v) = \sum_{\substack{incoming\ edges \\ (w,v)\ to\ v}} \frac{R(w)}{N_{out}(w)}$$

← 'out degree'

To calculate this, we need a list of the incoming edges to each vertex, similar to an adjacency list but now we list the incoming instead of outgoing edges.



Define the PageRank of v:

$$R(v) = \sum_{\substack{incoming\ edges \\ (w,v)\ to\ v}} \frac{R(w)}{N_{out}(w)}$$

This is recursive, but there is no base case.
Instead, initialize it to be uniform and iterate.

What does Google Search Engine do?

offline

What do you the user do?

What does Google Search Engine then do?

online

What does Google Search Engine do?

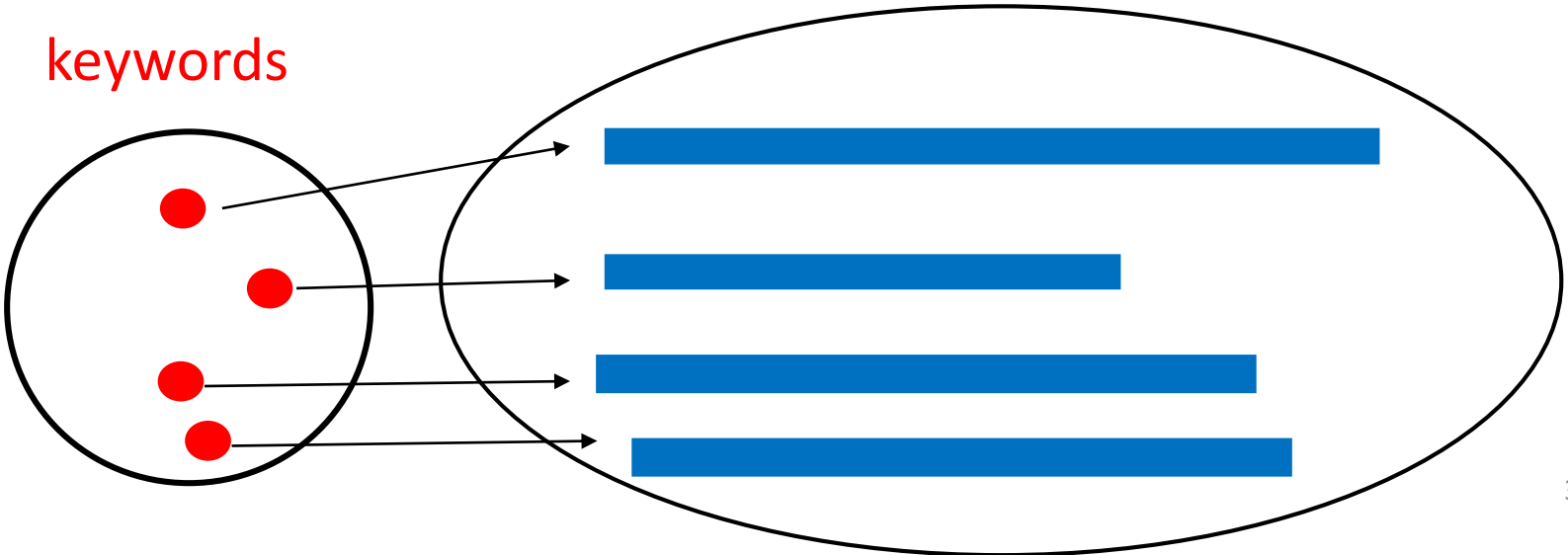
- Web crawler downloads all reachable web pages
- Build a graph (update)
- Compute page rank for each web page

What does Google Search Engine do?

- Web crawler downloads all reachable web pages
- Build a graph (update)
- Compute page rank for each web page
- Build a map:

offline

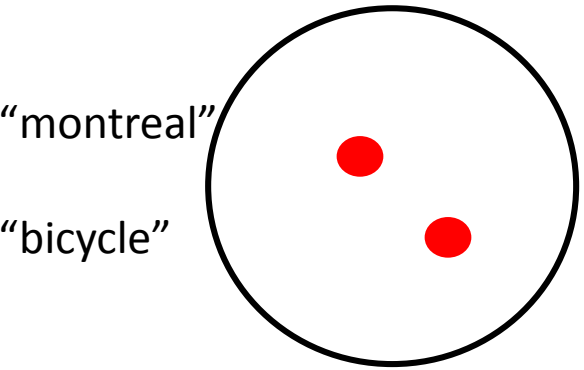
Values: list of web pages (URLs) containing keyword



What do you the user do?

- Enter query words (keywords)

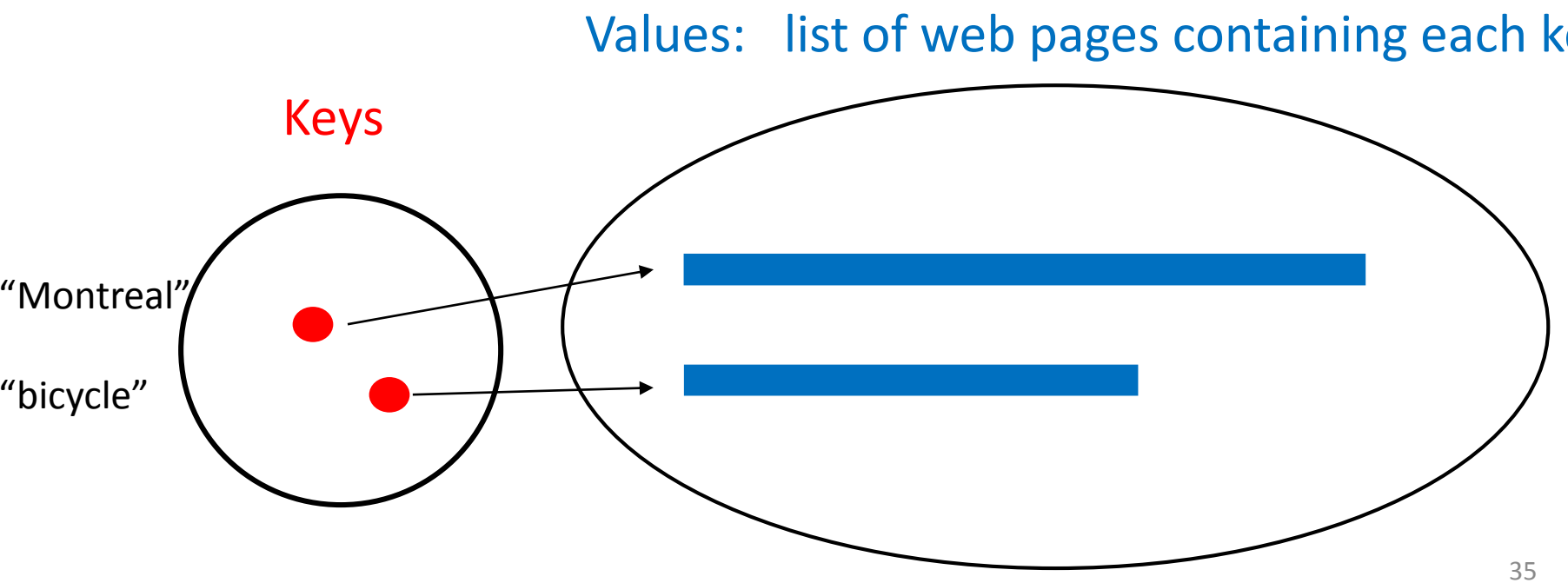
online



What does the search engine do?

online

- For each key, get the value (list of pages containing key, ordered by pagerank)
- If there are multiple keys in the search, compute the intersection of the lists of web pages
- Output the resulting list (in order of PageRank)



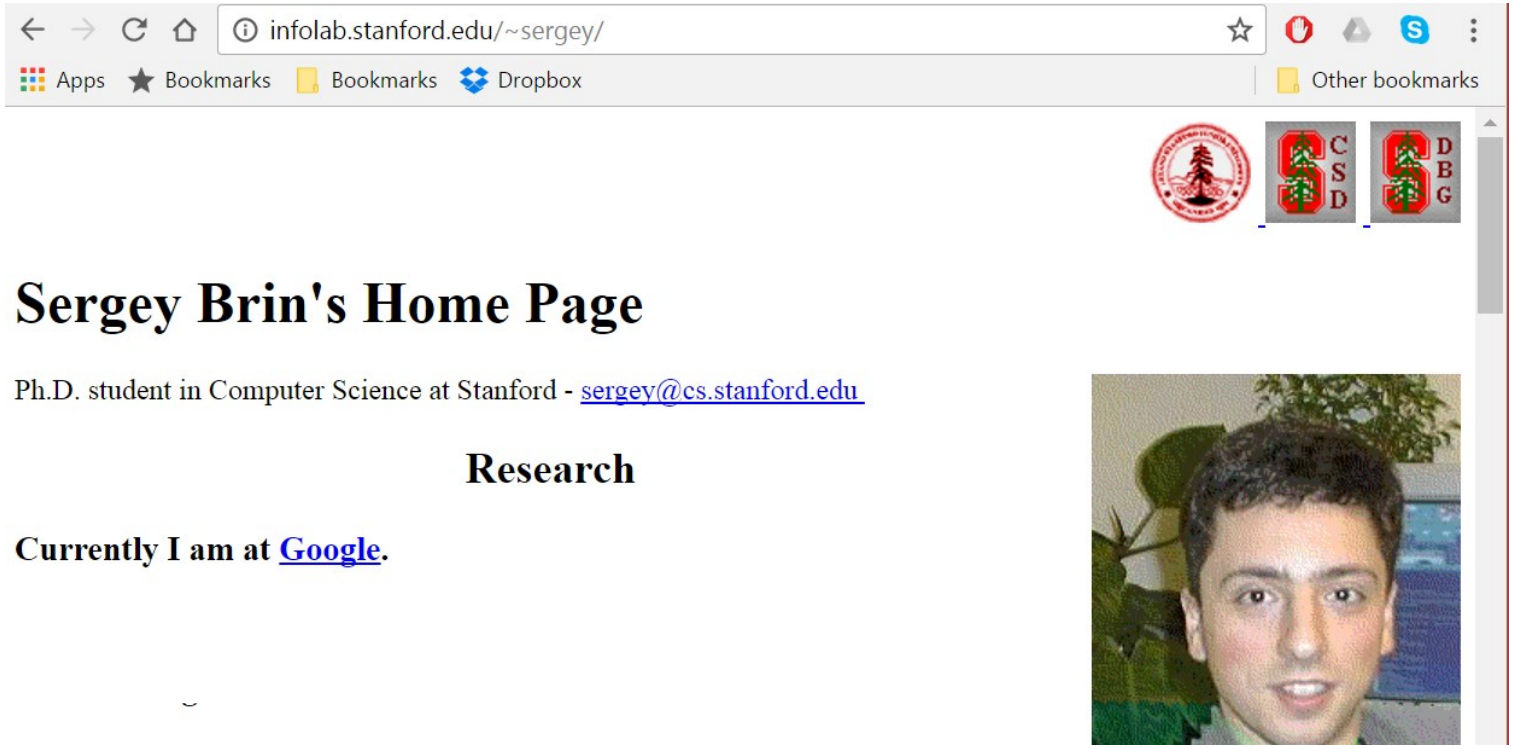
What I just described was the original PageRank idea.

In the last 5-10 years, google search has become much more complicated:

e.g. It is tuned to individual users (e.g. Your geographic location and previous searchers)

For more details, including the original research papers describing page rank, see Sergey Brin's old home page from the late 1990's:

http://infolab.stanford.edu/~sergey



Google's other founder is Larry Page (no pun intended)
https://engineering.stanford.edu/about/heroes/larry-page