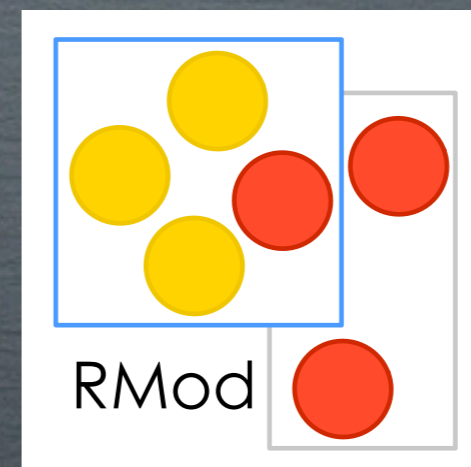


OBJECT SWAPPING ISSUES AND THE IMAGESEGMENT IMPLEMENTATION

I N G . M A R I A N O M A R T I N E Z P E C K

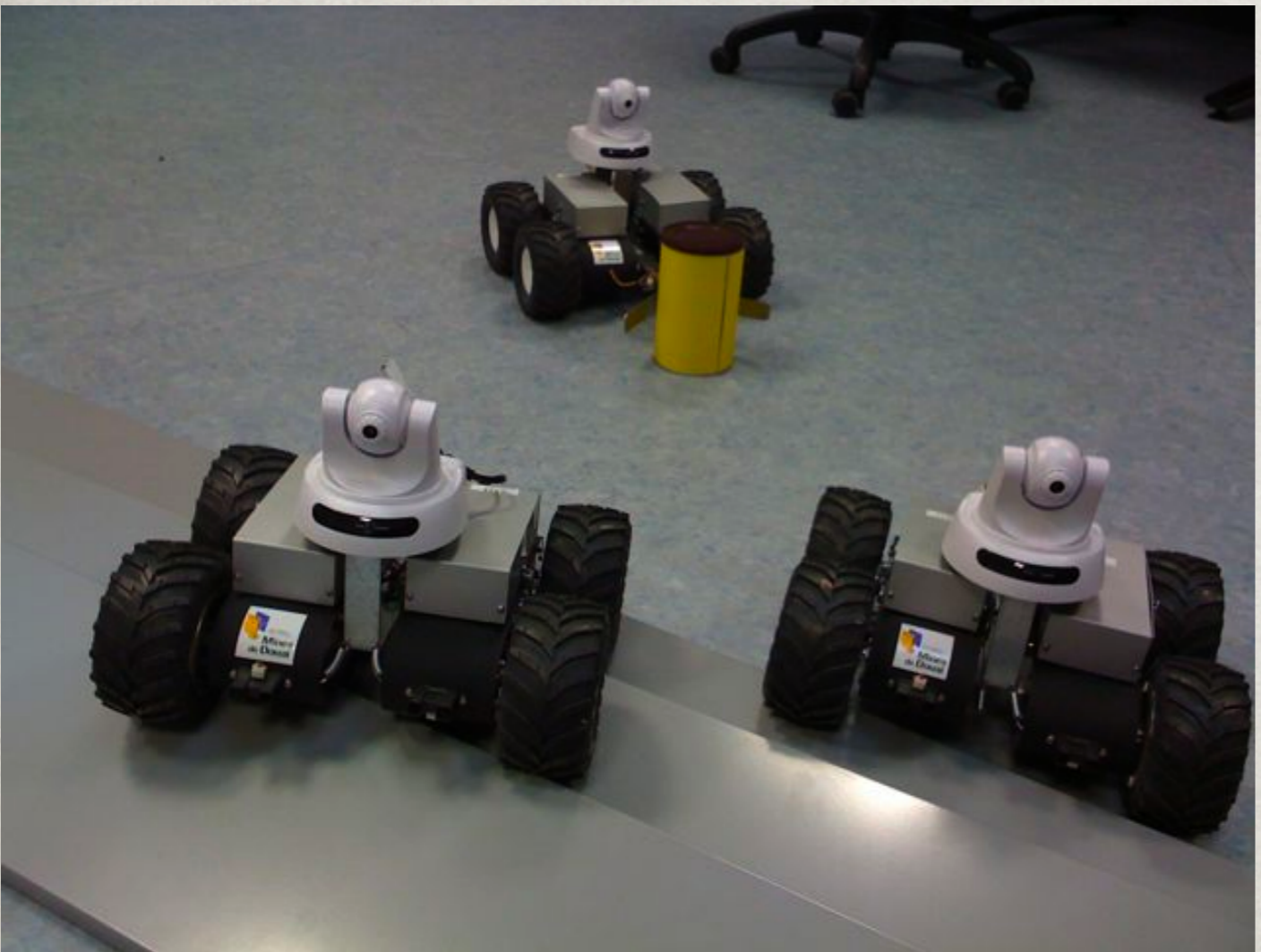
marianopeck@gmail.com

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE



THE CONTEXT

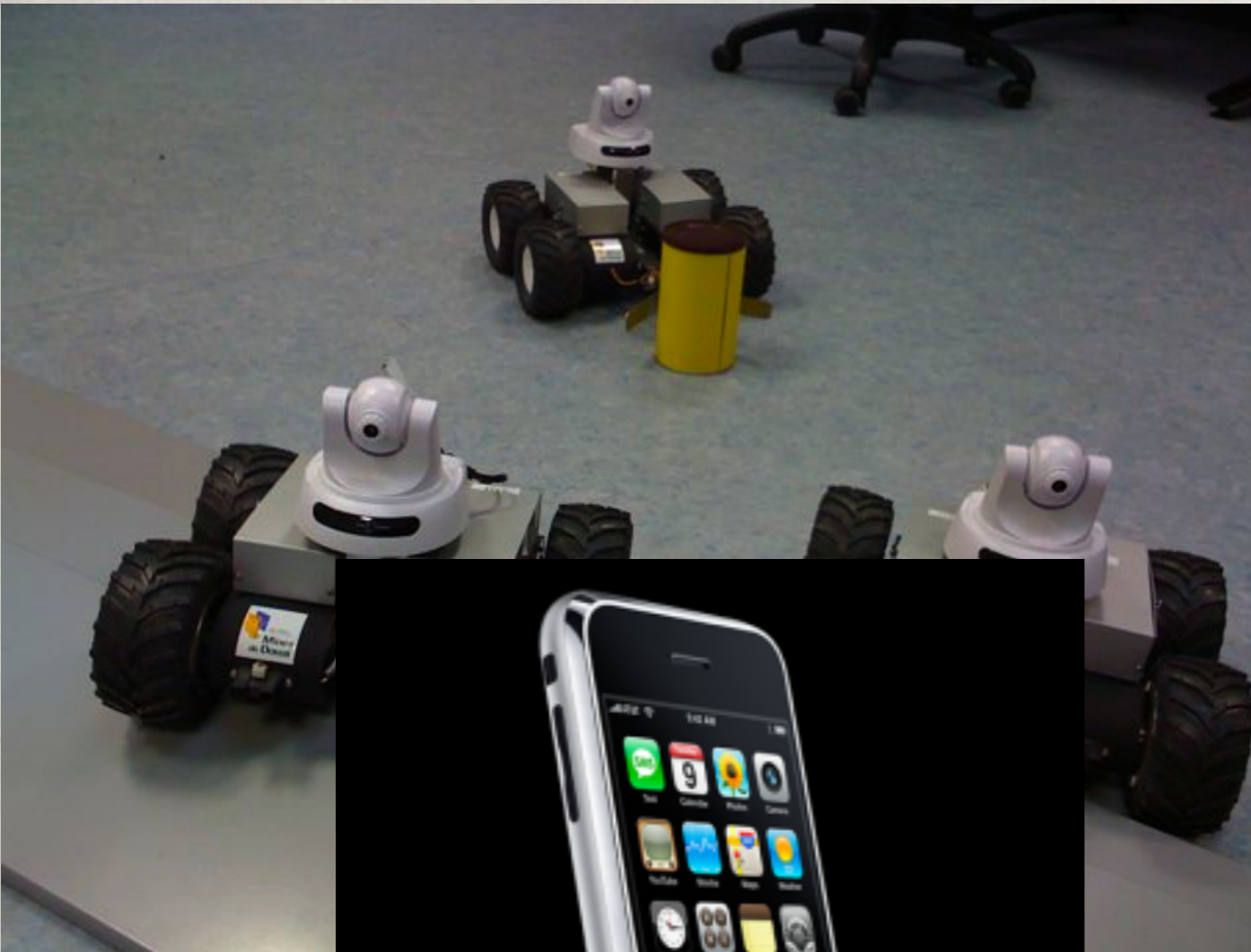
THE CONTEXT



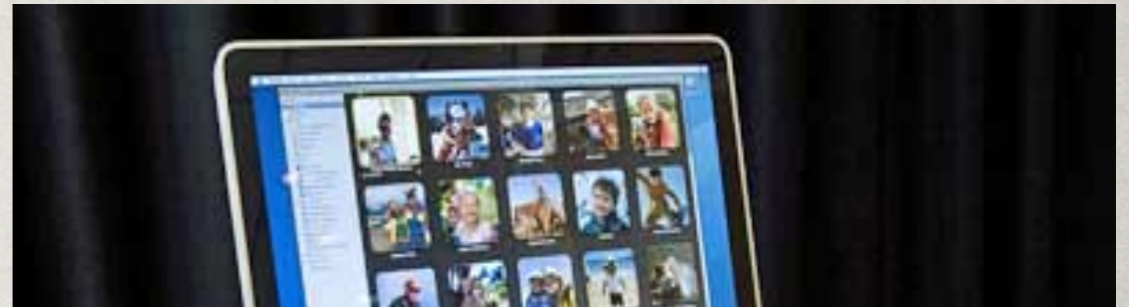
THE CONTEXT



THE CONTEXT



THE CONTEXT

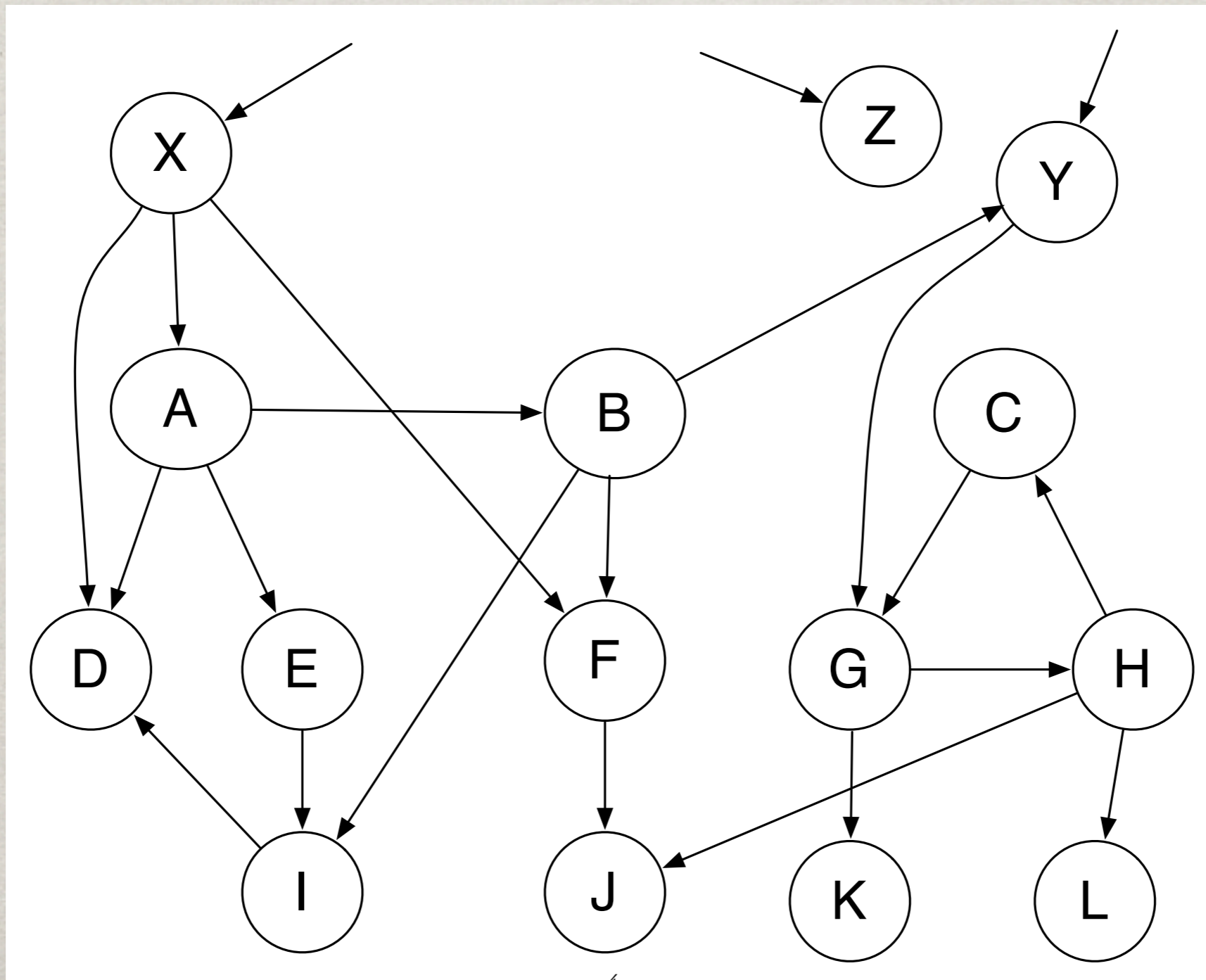


PROBLEM

- ✱ Use more memory than needed.
- ✱ Make OOP languages unsuitable for memory limited devices.
- ✱ **Existence of unused but referenced objects.**

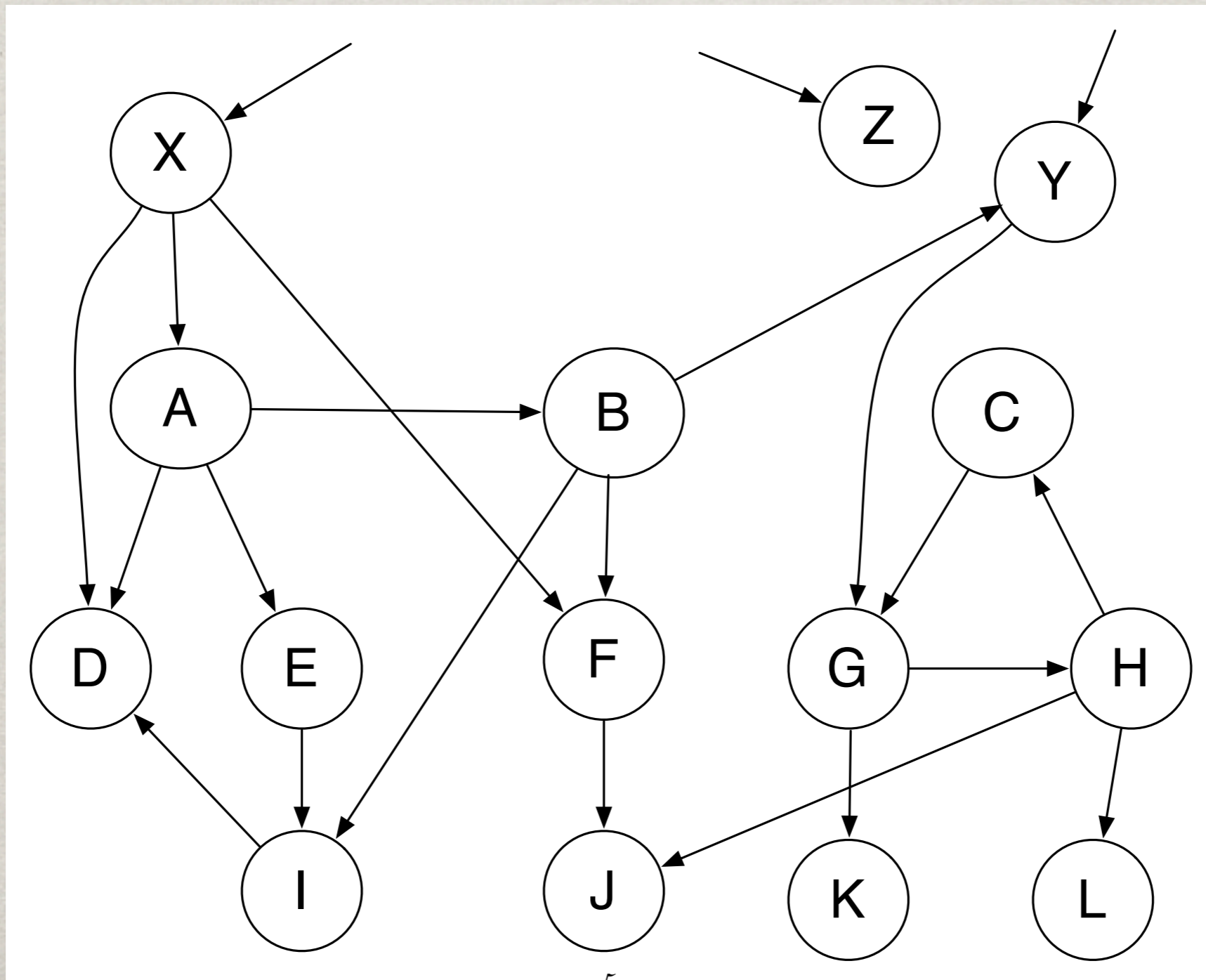
THE CONTEXT

In OOP primary memory is represented by an object graph



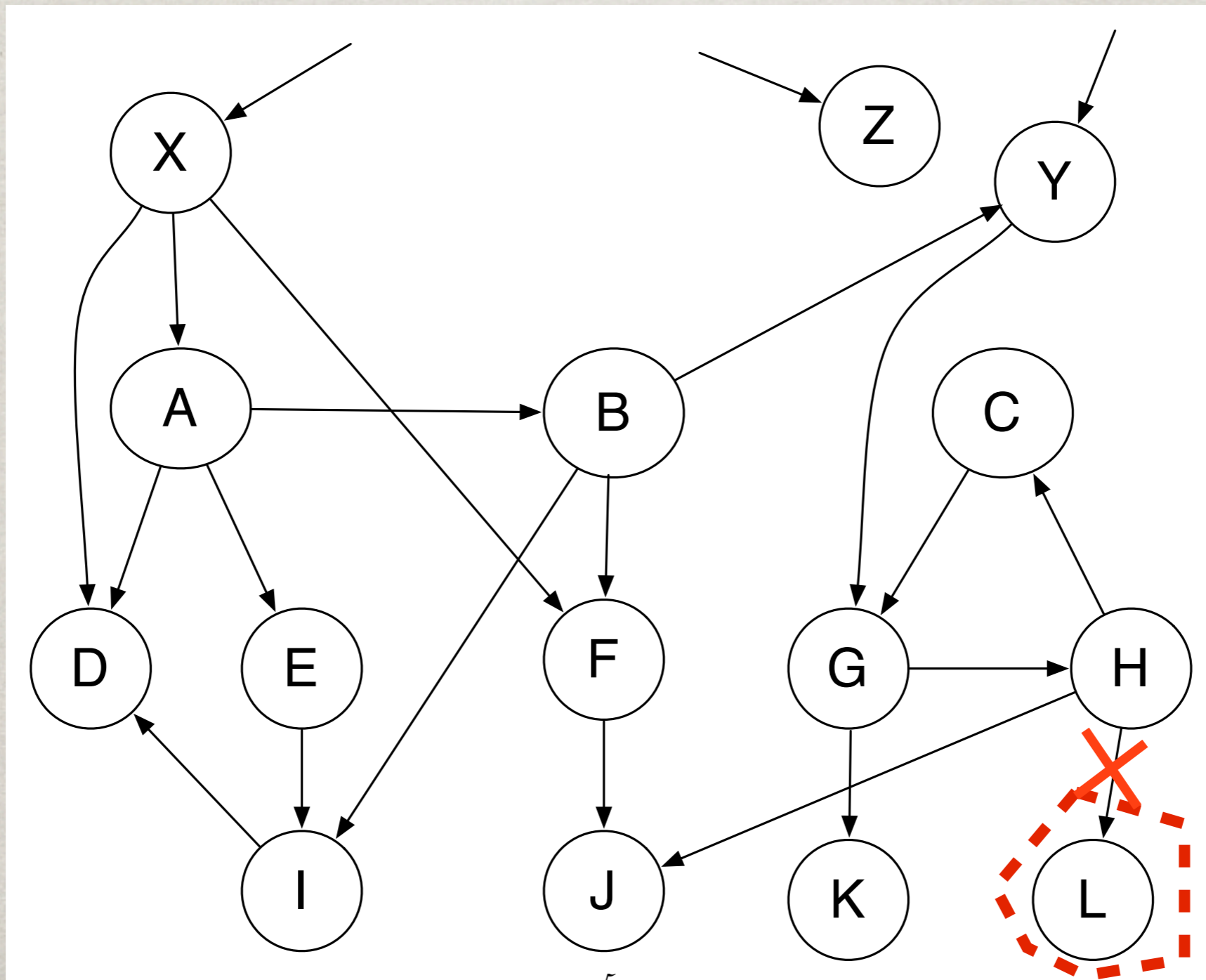
GARBAGE COLLECTOR

Only collects objects that nobody else points to.



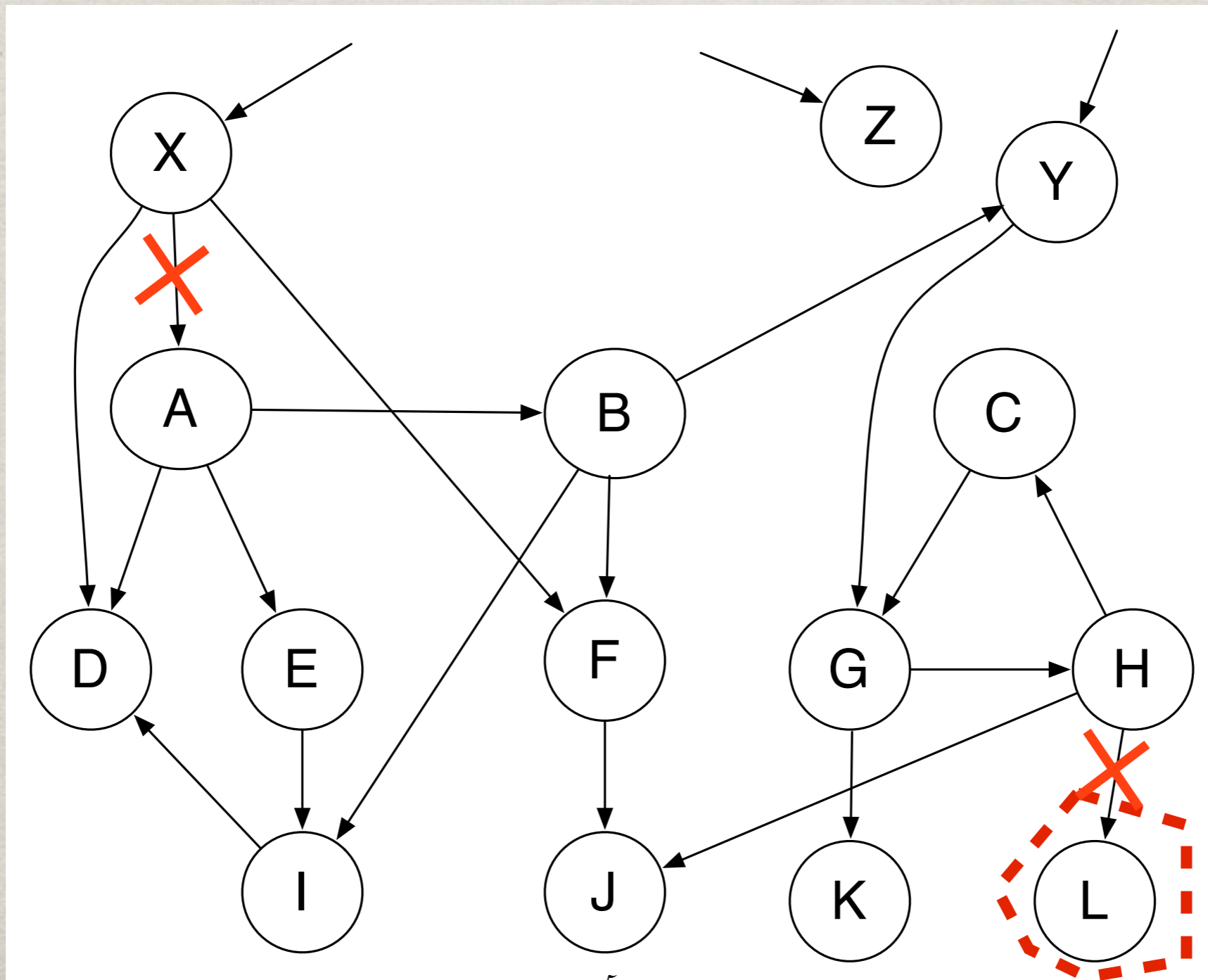
GARBAGE COLLECTOR

Only collects objects that nobody else points to.



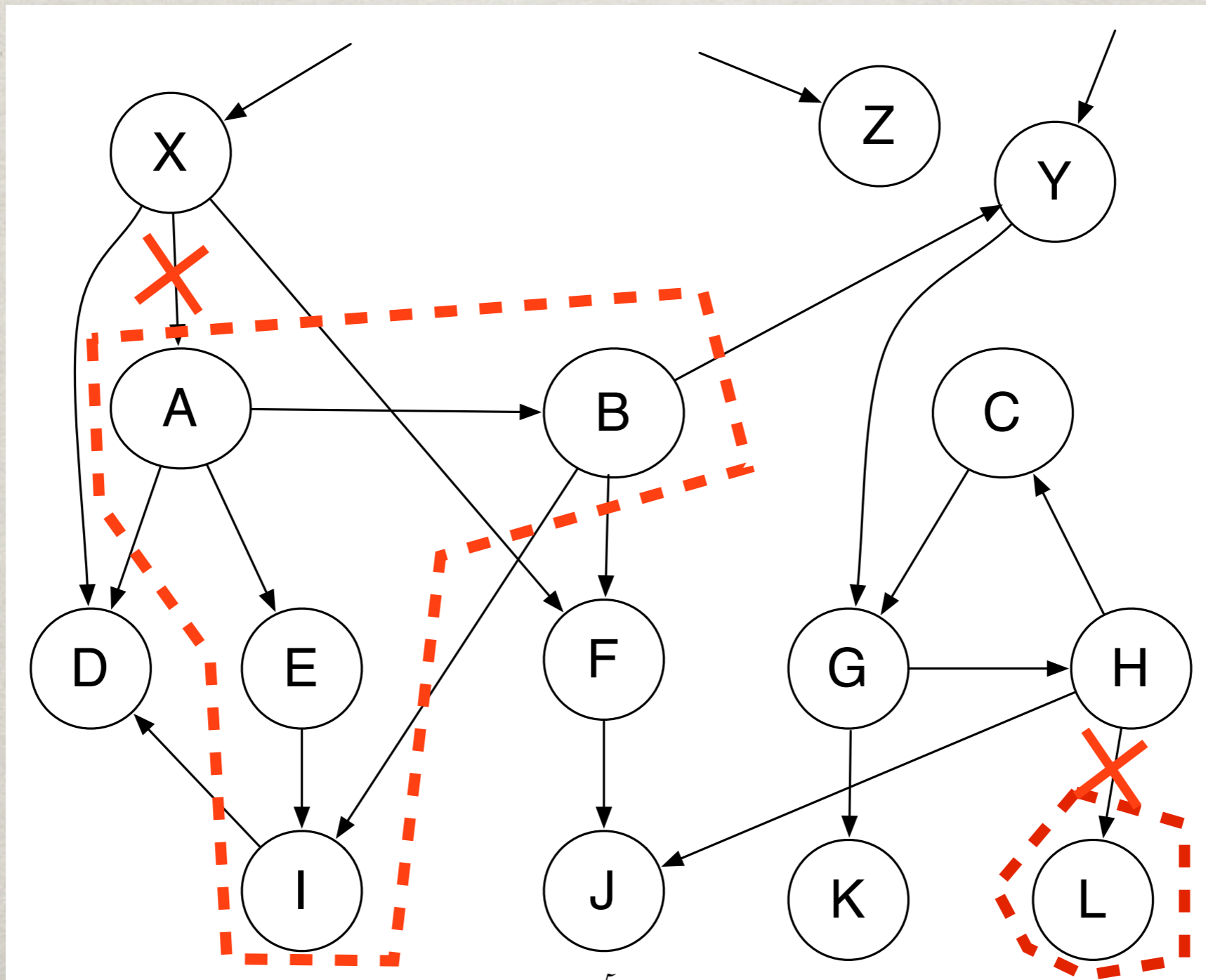
GARBAGE COLLECTOR

Only collects objects that nobody else points to.

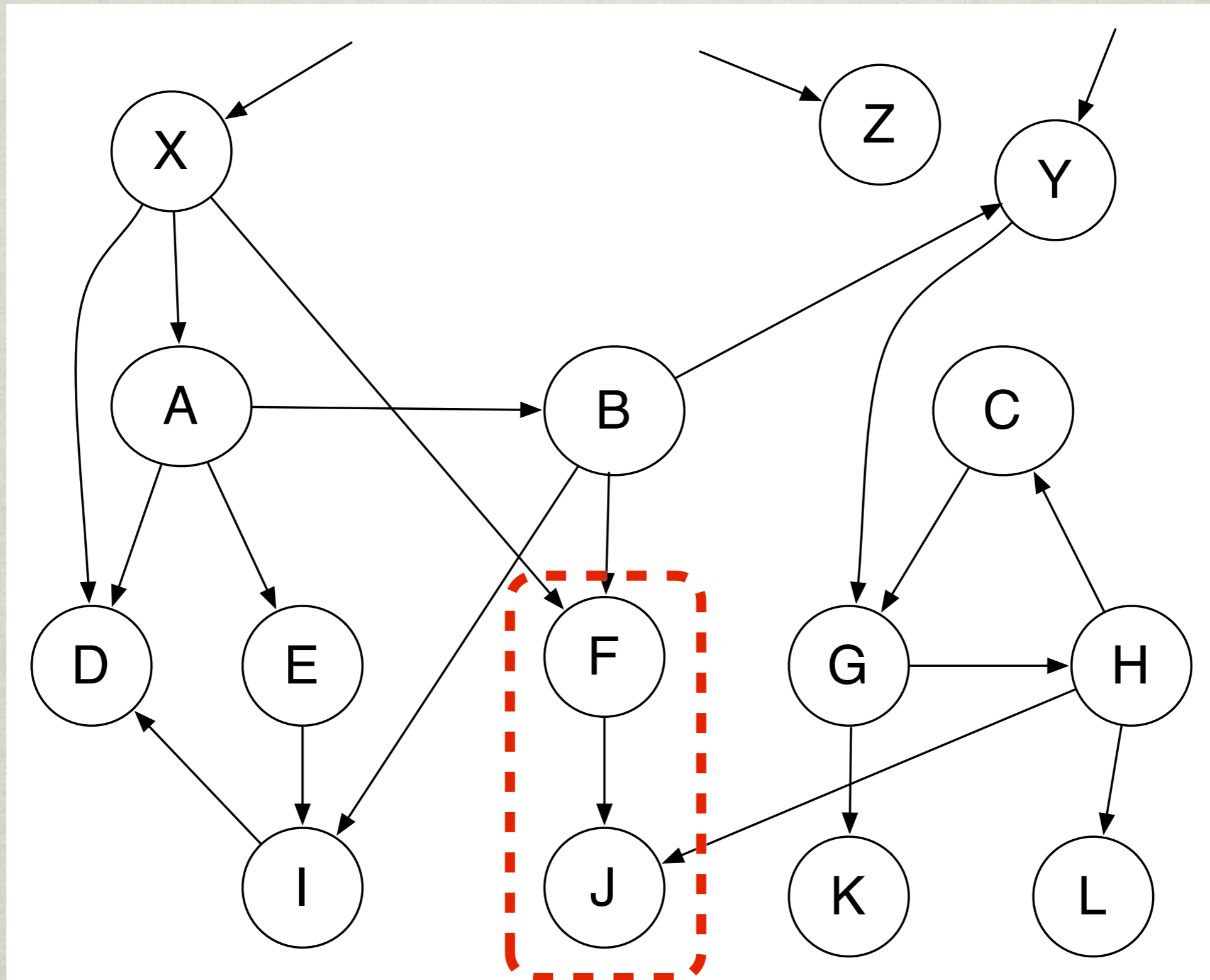


GARBAGE COLLECTOR

Only collects objects that nobody else points to.



But...what happens with referenced yet unused objects?



IDEA

- ✻ Swap out (not remove) unused objects to disk.
- ✻ Automatically load them back when needed.

There are related works...but no one solves all the
problems

MAIN CHALLENGES

- ✱ Not to use more memory than the one released by swapping.
- ✱ Low overhead penalty.
- ✱ Group objects in an smart way.

KEY ASPECTS

- ✻ Mark and trace unused/used objects at runtime.
- ✻ The usage of proxies.
- ✻ Group unused objects (**subgraphs**).

WHY WE NEED TO GROUP OBJECTS?

- ✻ Because if we replace each object by a proxy, we release little memory.
- ✻ We want to replace a whole group by one or a few proxies.

WHY SUBGRAPHS?

- ✱ Group of objects that are used (or not used) together.
- ✱ We need few proxies (for the roots) for several objects.

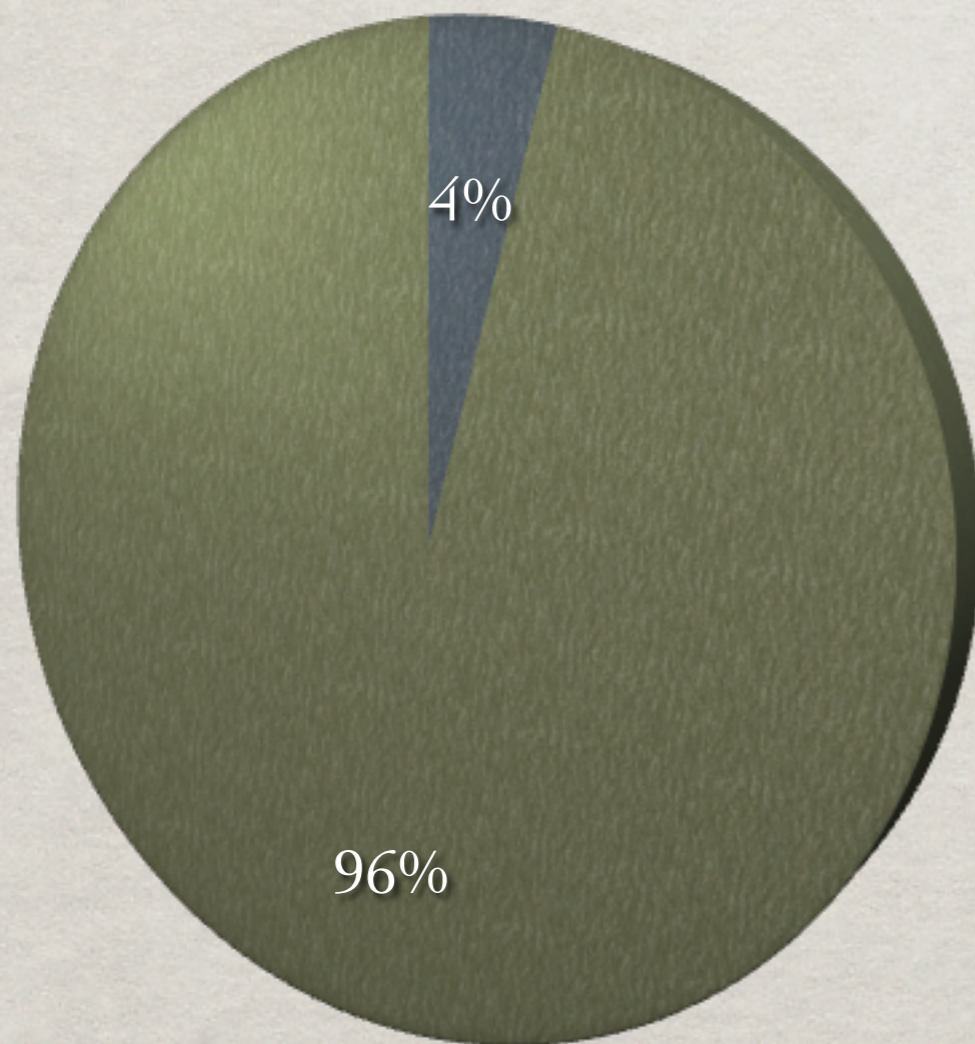
EXPERIMENTS DONE

- ✻ Modify Smalltalk VM to mark and trace objects usage.
- ✻ Visualize objects and memory usage.
- ✻ Take statistics from different scenarios.

DEPLOYED WEB APPLICATION EXAMPLE

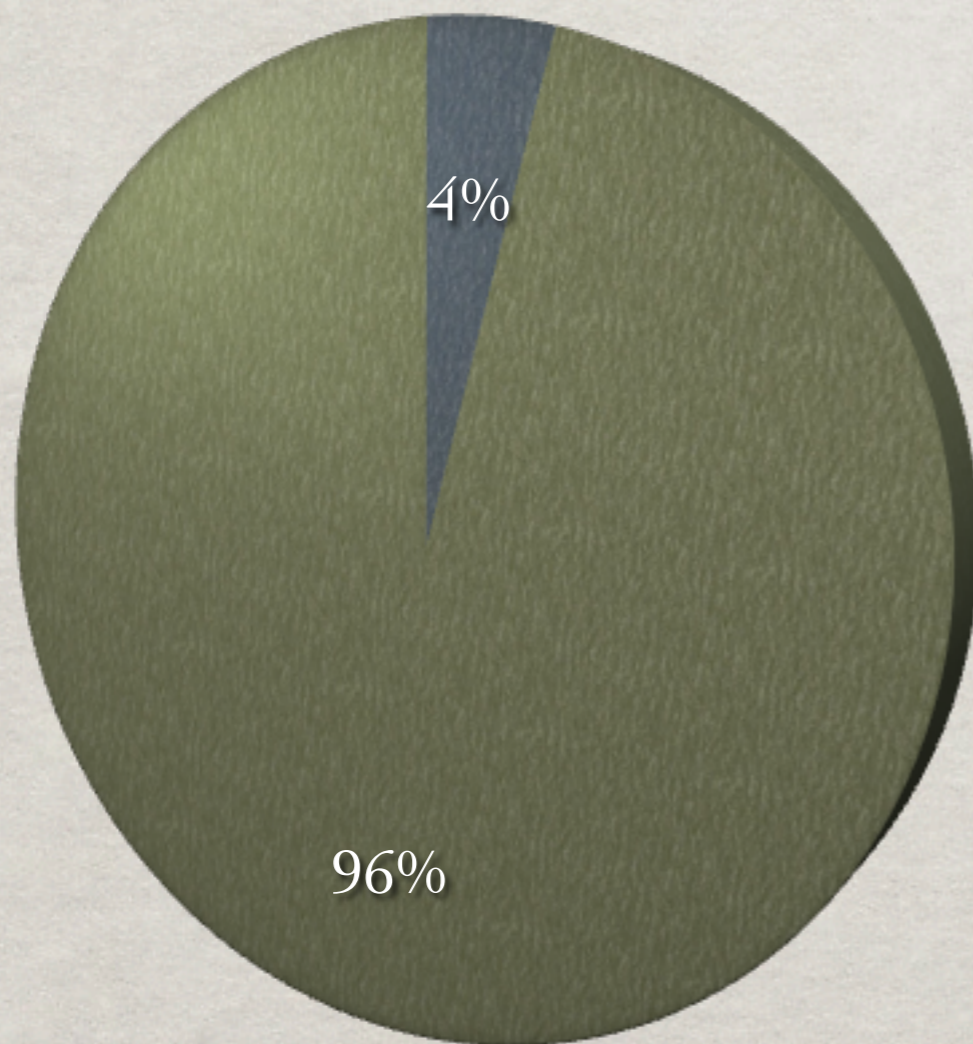
DEPLOYED WEB APPLICATION EXAMPLE

● Used ● Unused
Amount of objects

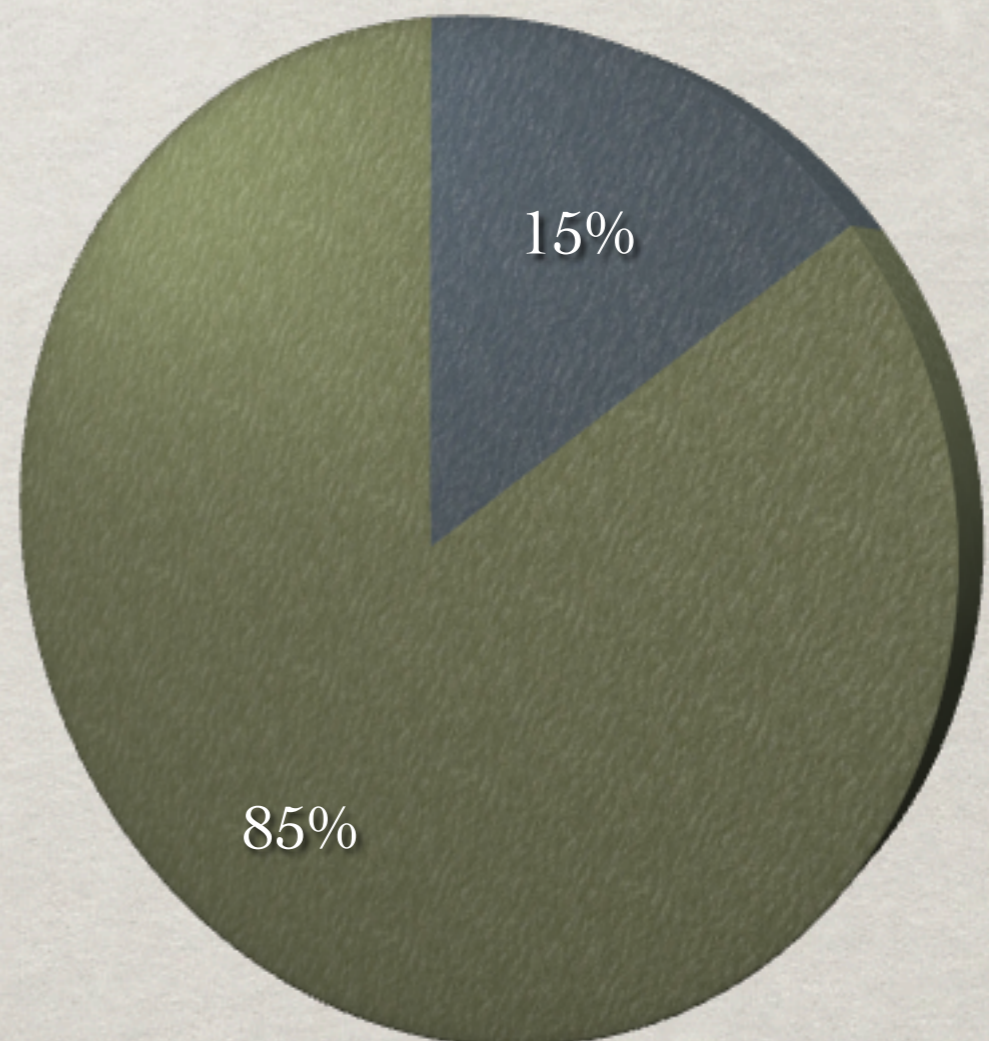


DEPLOYED WEB APPLICATION EXAMPLE

● Used ● Unused
Amount of objects



● Used ● Unused
Amount of memory



SWAPPING STEPS AND CHALLENGES

1. Identify sets of objects and serialize them.

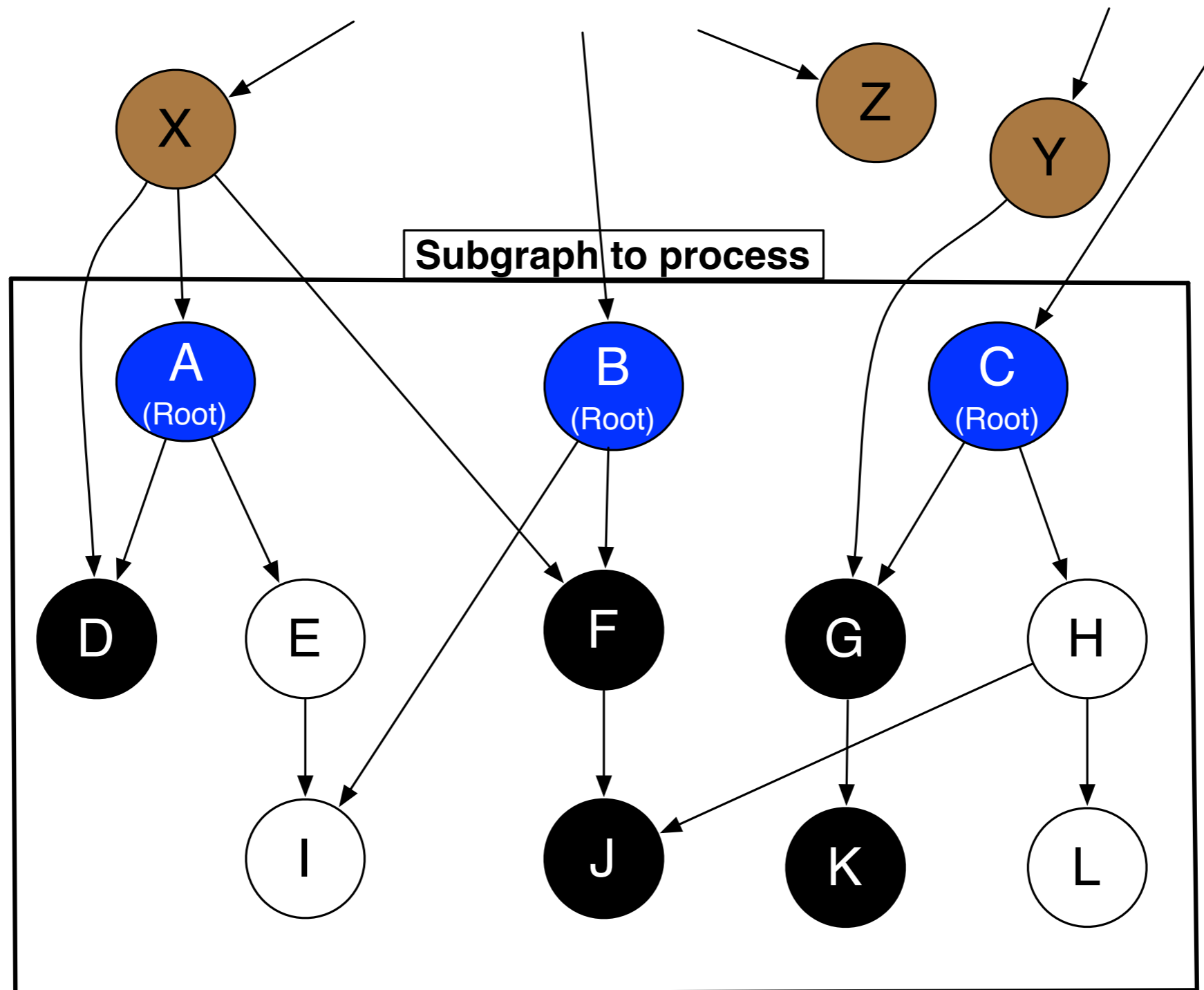
Problems: cycles, **speed**, etc.

2. Write the serialized objects into a file.

Problems: file format, encoding, **speed**, etc.

3. Load the objects from a file. Problems: class reshape, avoid duplicates, **speed**, etc.

SUBGRAPHS



MORE PROBLEMS

- Should shared objects be included or not?

- GC moves objects.

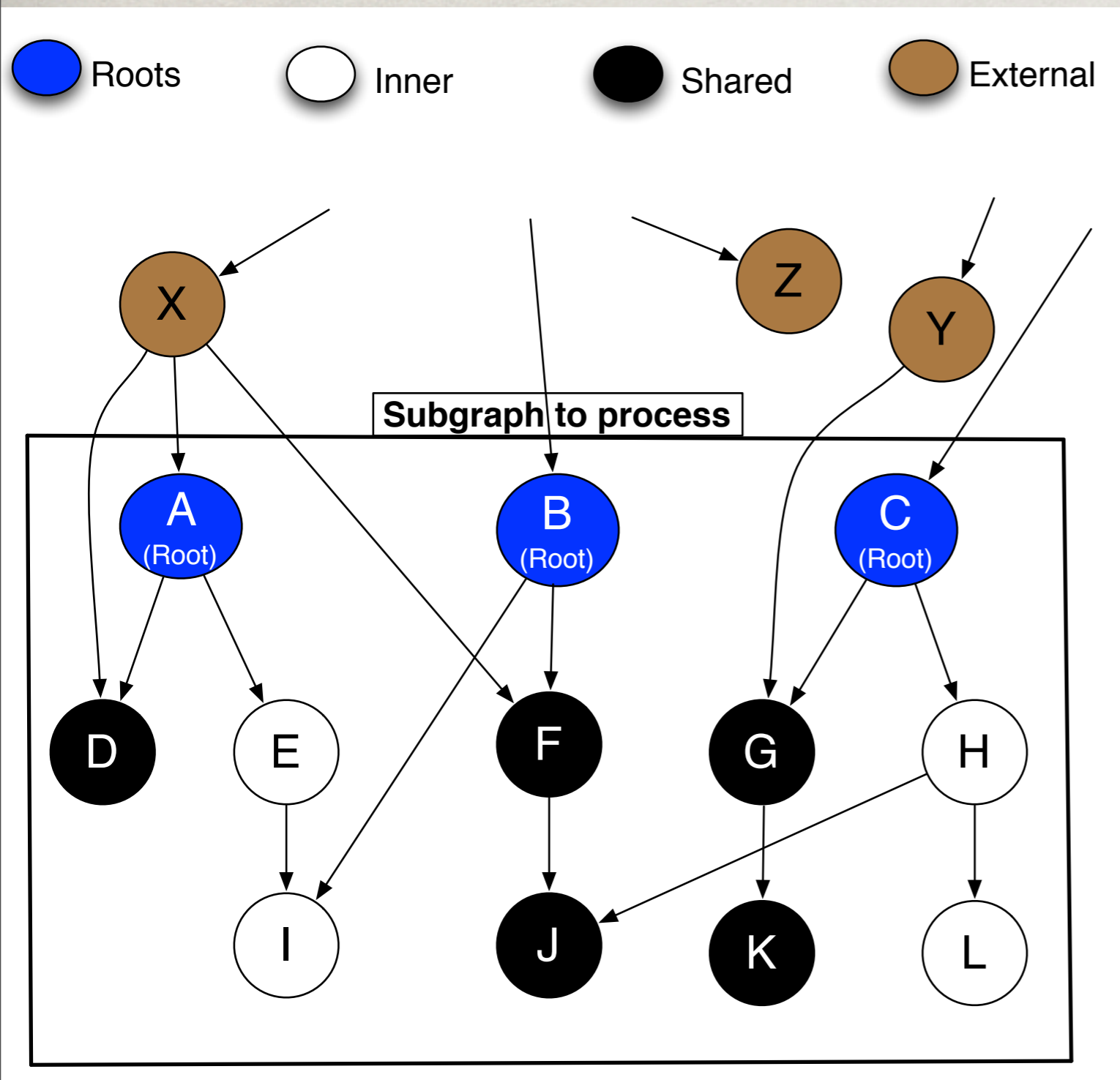
- Pointers update.

- Class changes.

- Recreate and reinitialize objects.

- Code executed after loading.

- Special objects.

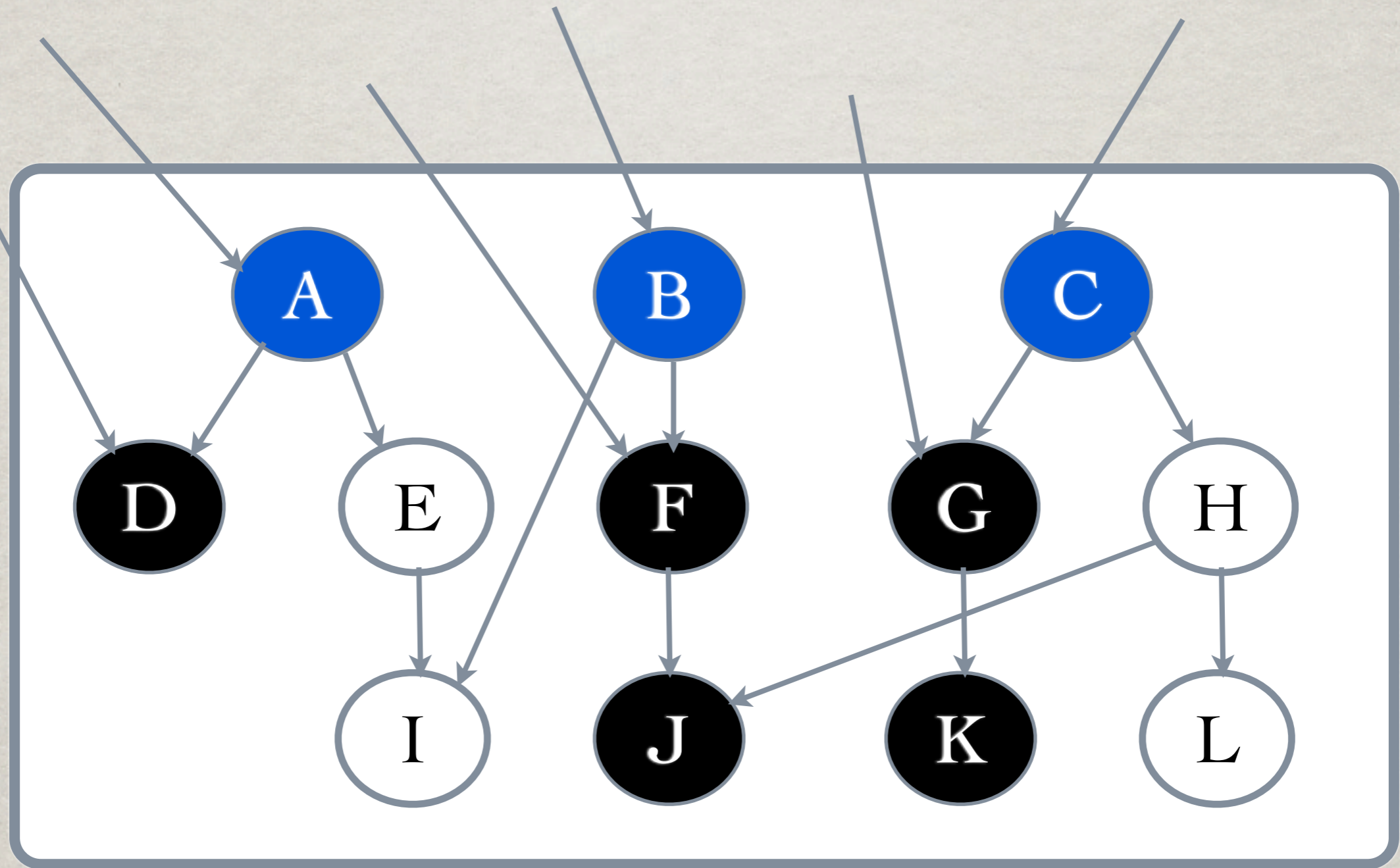


IMAGESEGMENT

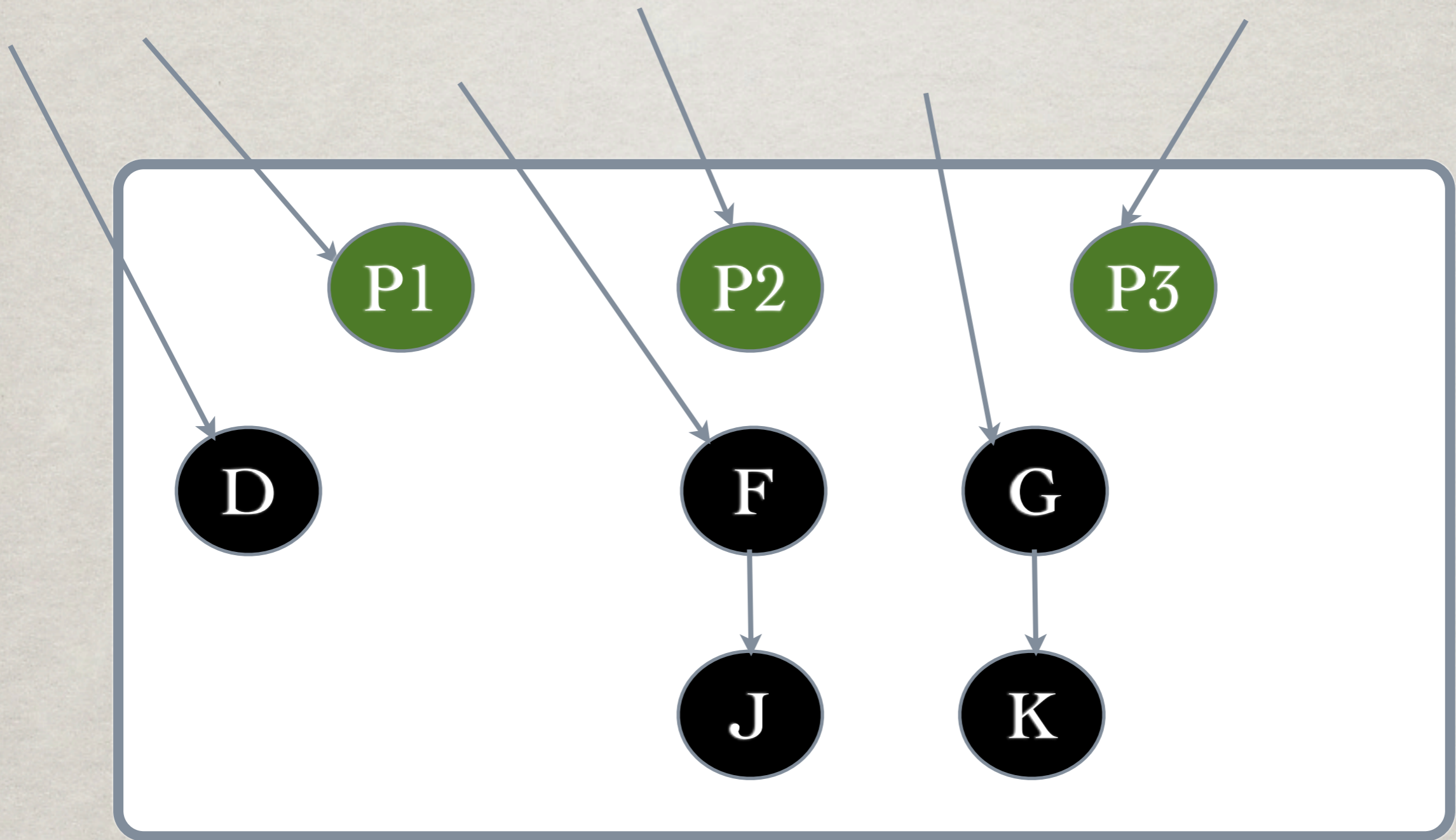
IMAGESEGMENT BASIS

- ✻ Only write/swap roots and inner objects.
- ✻ Shared objects are NOT swapped.
- ✻ Keep an array in memory for the shared objects.
- ✻ Update object pointers to point to a relative address inside the arrays (offset).
- ✻ Roots are replaced by proxies.
- ✻ Uses GC facilities to detect shared objects.

THE GOAL

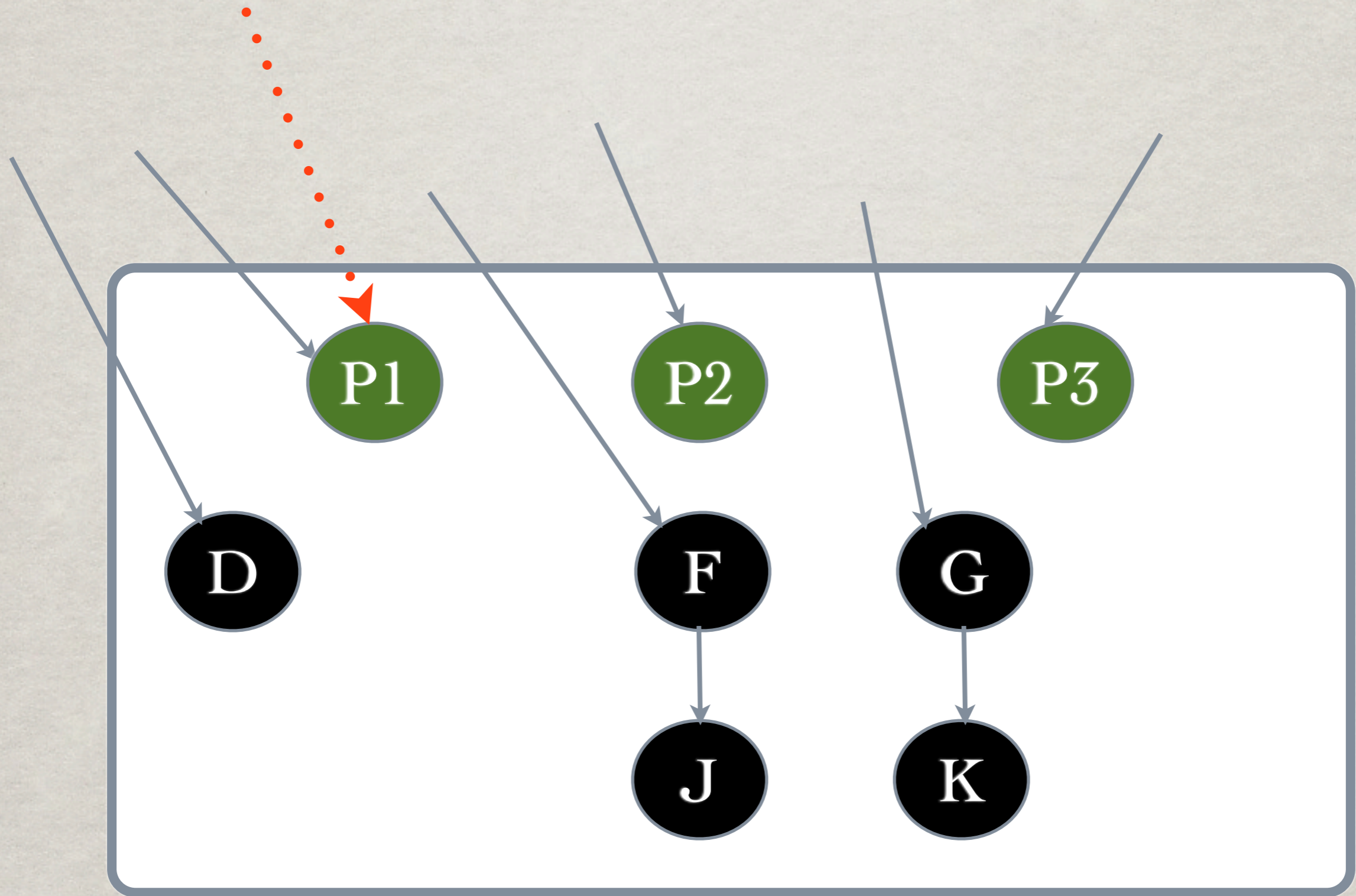


THE GOAL

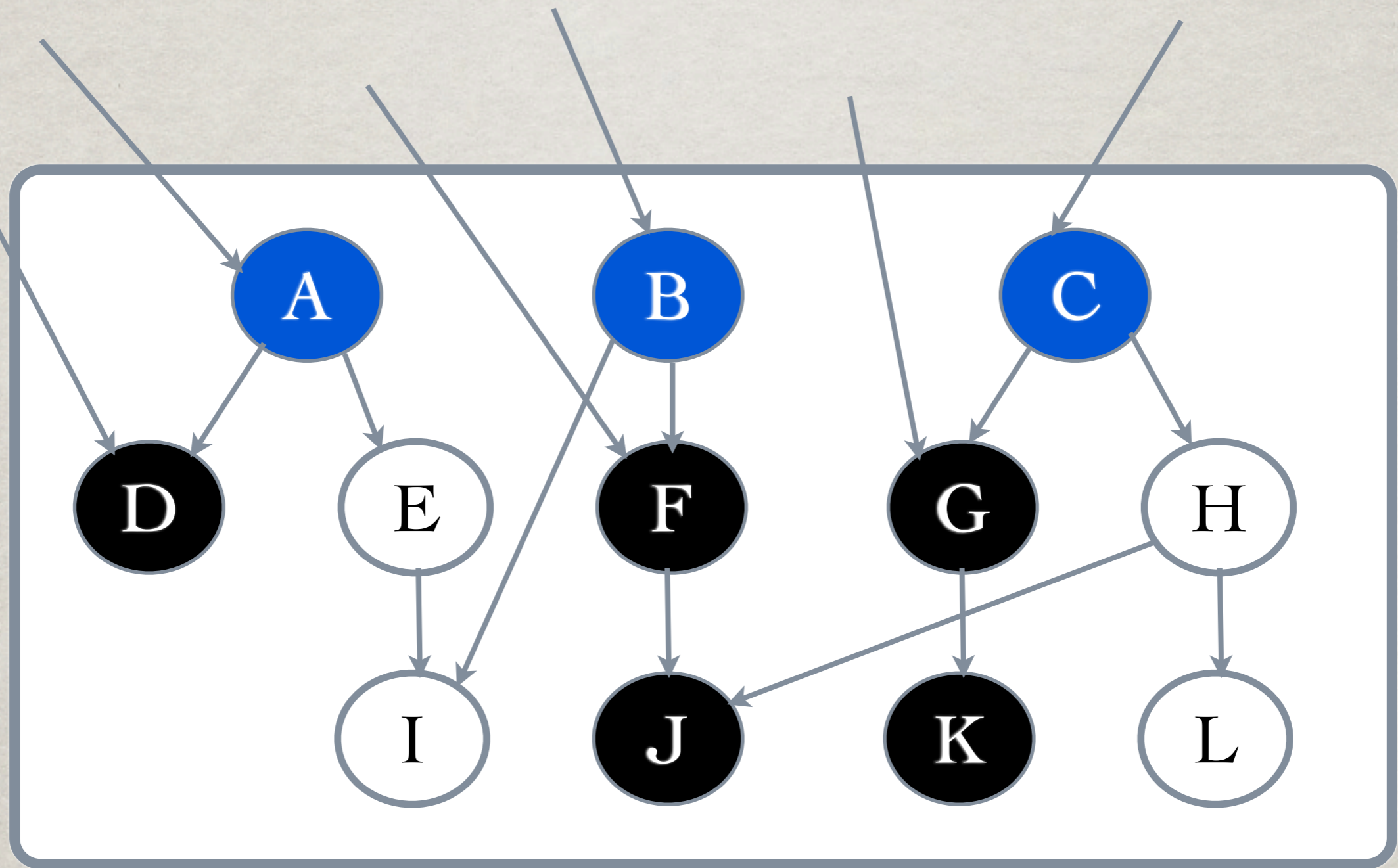


THE GOAL

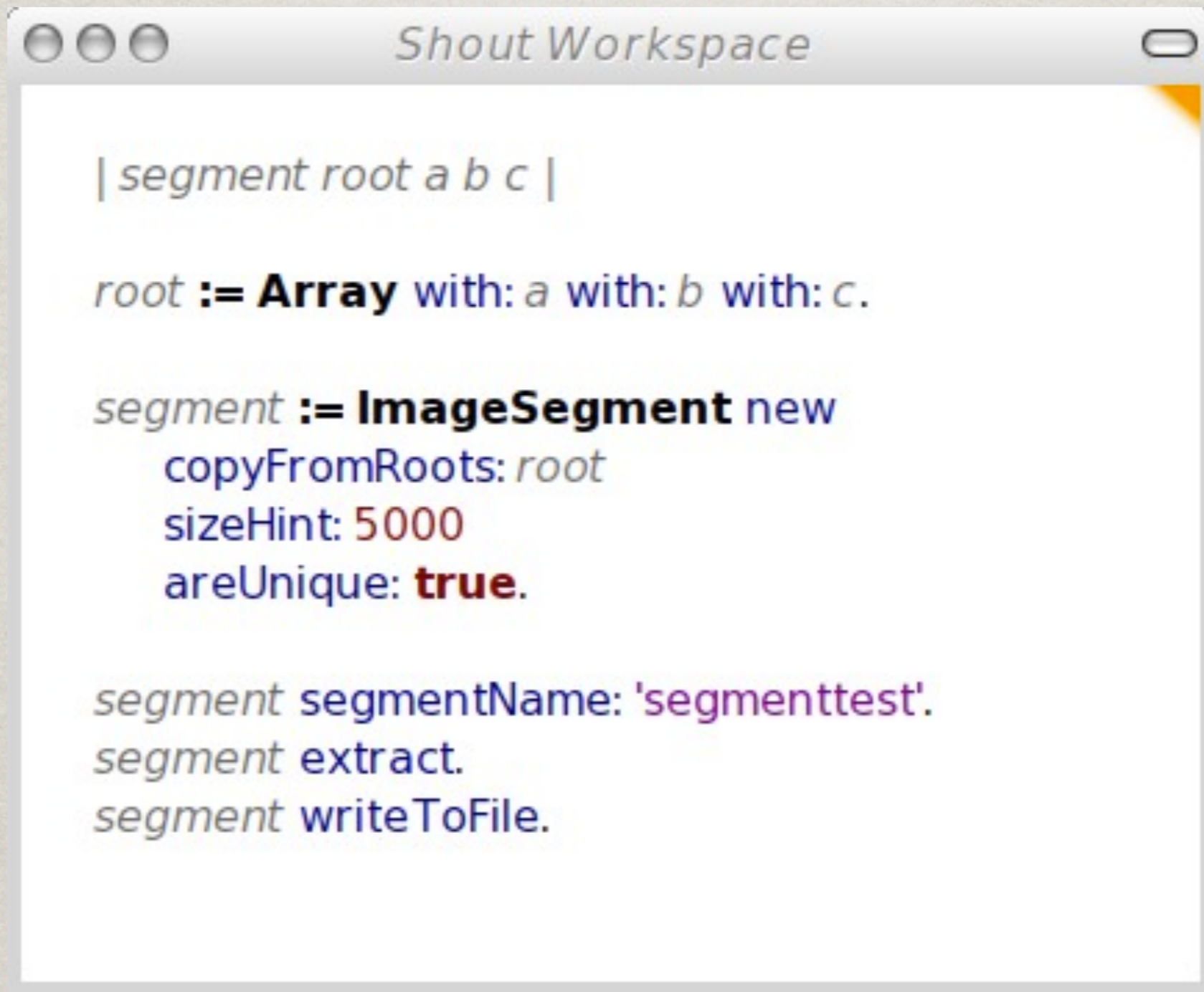
aMessage



THE GOAL

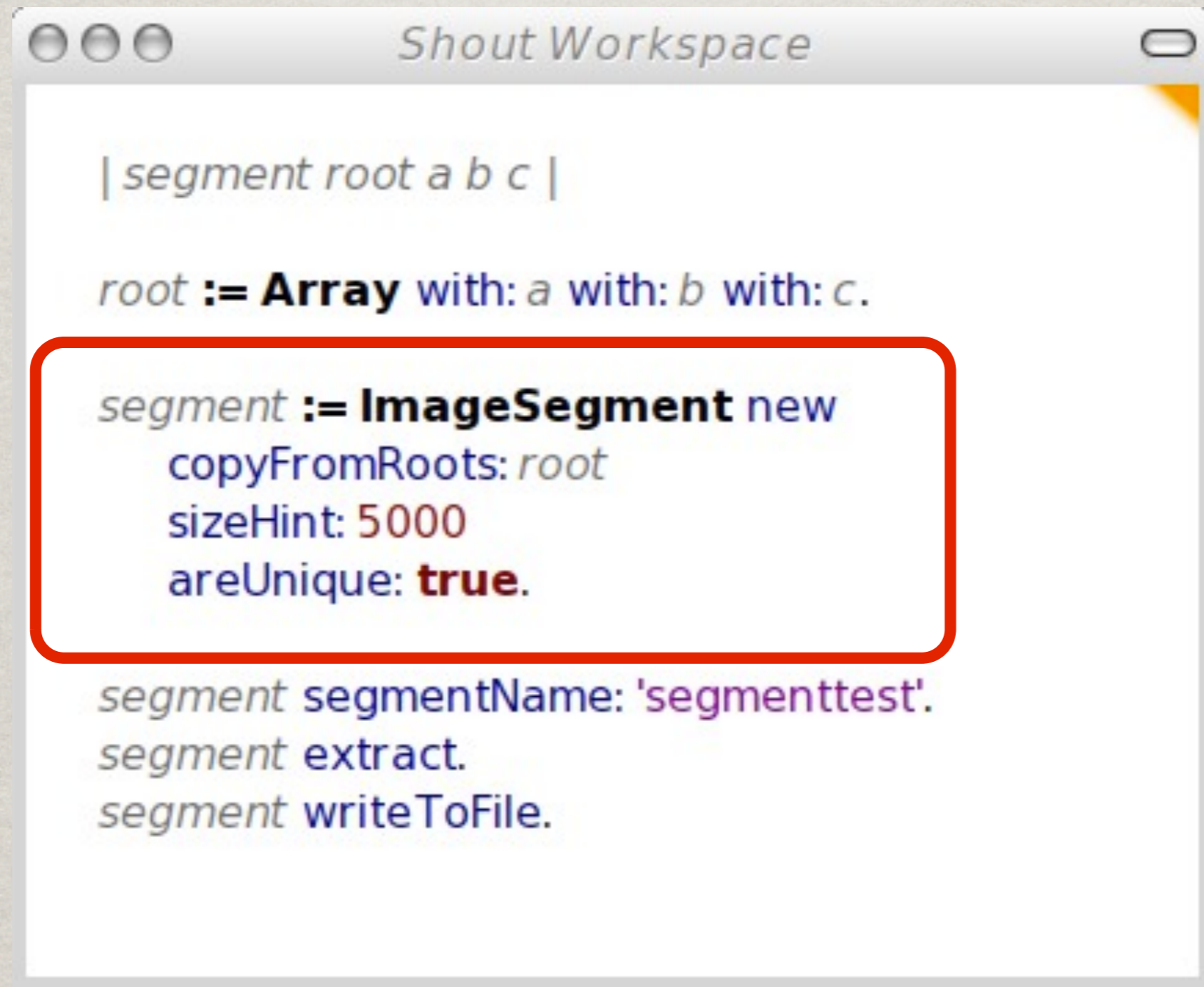


SIMPLE EXAMPLE

A screenshot of a window titled "Shout Workspace". The window has a standard macOS-style title bar with three buttons on the left and a close button on the right. The content area is white and contains several lines of code. The code is color-coded: "segment" is purple, "root" is blue, "Array" is bold black, "ImageSegment" is bold black, "new" is blue, "copyFromRoots:" is blue, "root" is blue, "sizeHint:" is blue, "5000" is red, "areUnique:" is blue, "true" is red, "segmentName:" is blue, "'segmenttest'" is purple, "extract" is blue, and "writeToFile" is blue. The code is as follows:

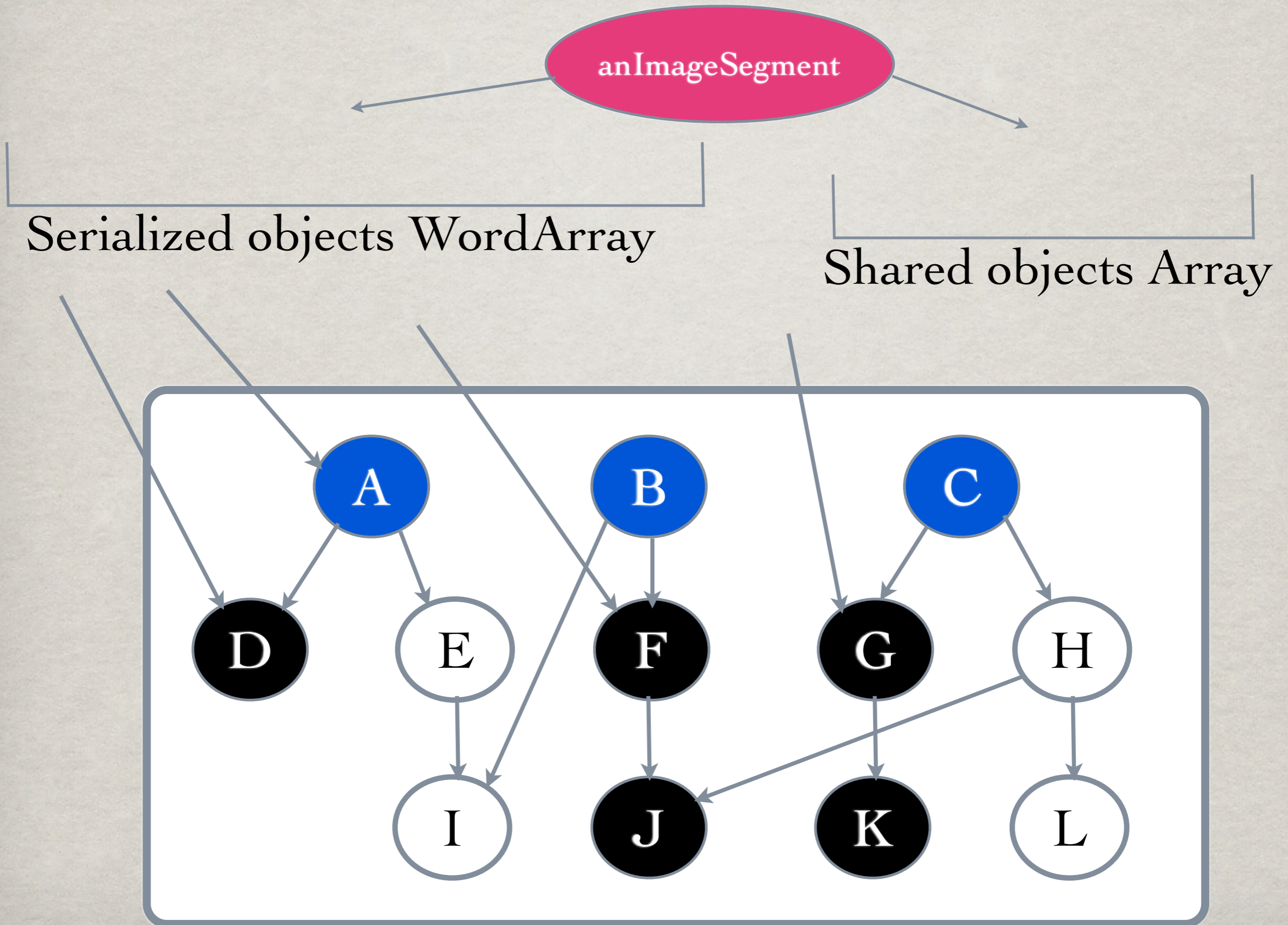
```
| segment root a b c |  
  
root := Array with: a with: b with: c.  
  
segment := ImageSegment new  
  copyFromRoots: root  
  sizeHint: 5000  
  areUnique: true.  
  
segment segmentName: 'segmenttest'.  
segment extract.  
segment writeToFile.
```

SUBGRAPH TRAVERSE

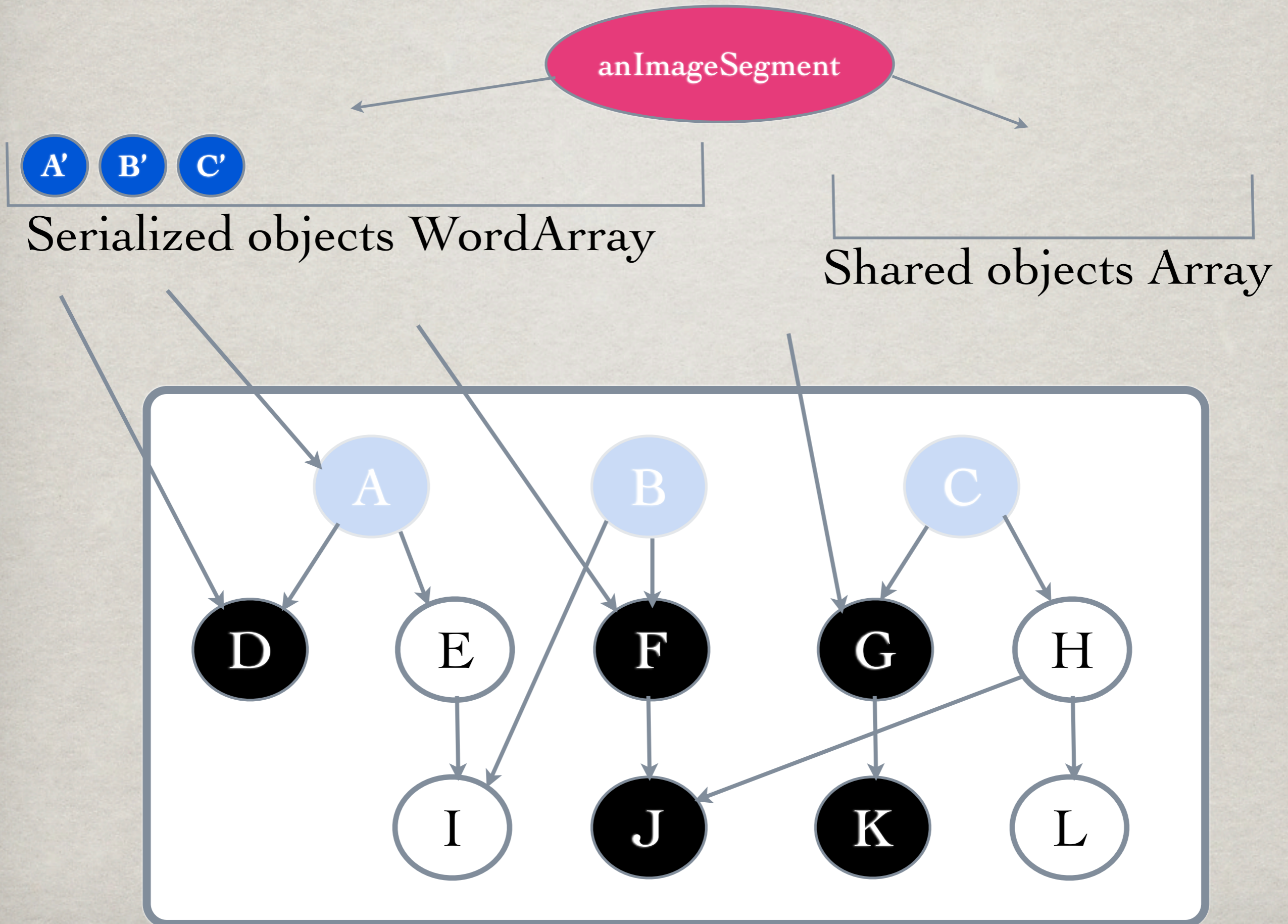


```
| segment root a b c |  
  
root := Array with: a with: b with: c.  
  
segment := ImageSegment new  
  copyFromRoots: root  
  sizeHint: 5000  
  areUnique: true.  
  
segment segmentName: 'segmenttest'.  
segment extract.  
segment writeToFile.
```

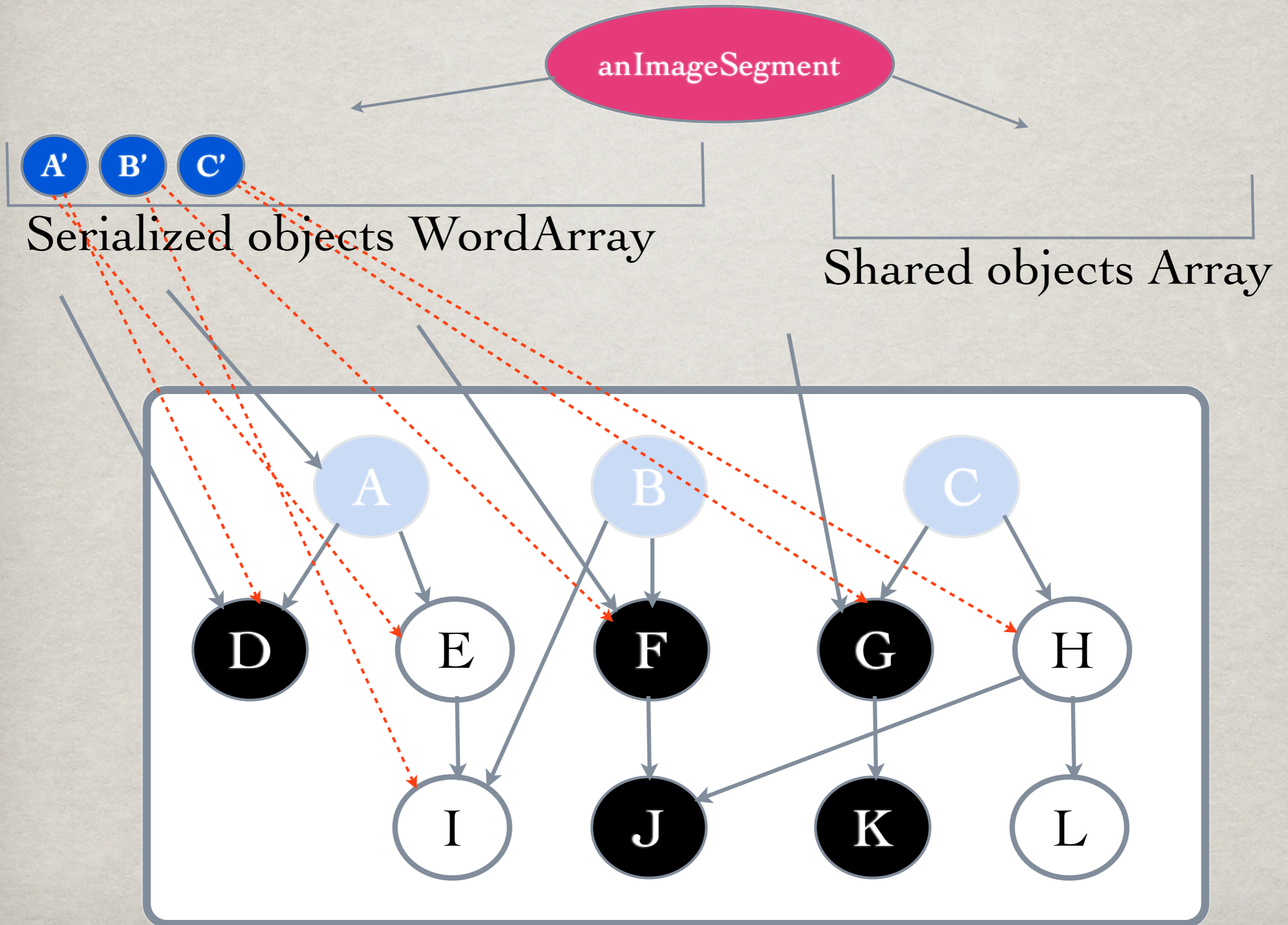
SUBGRAPH TRAVERSE



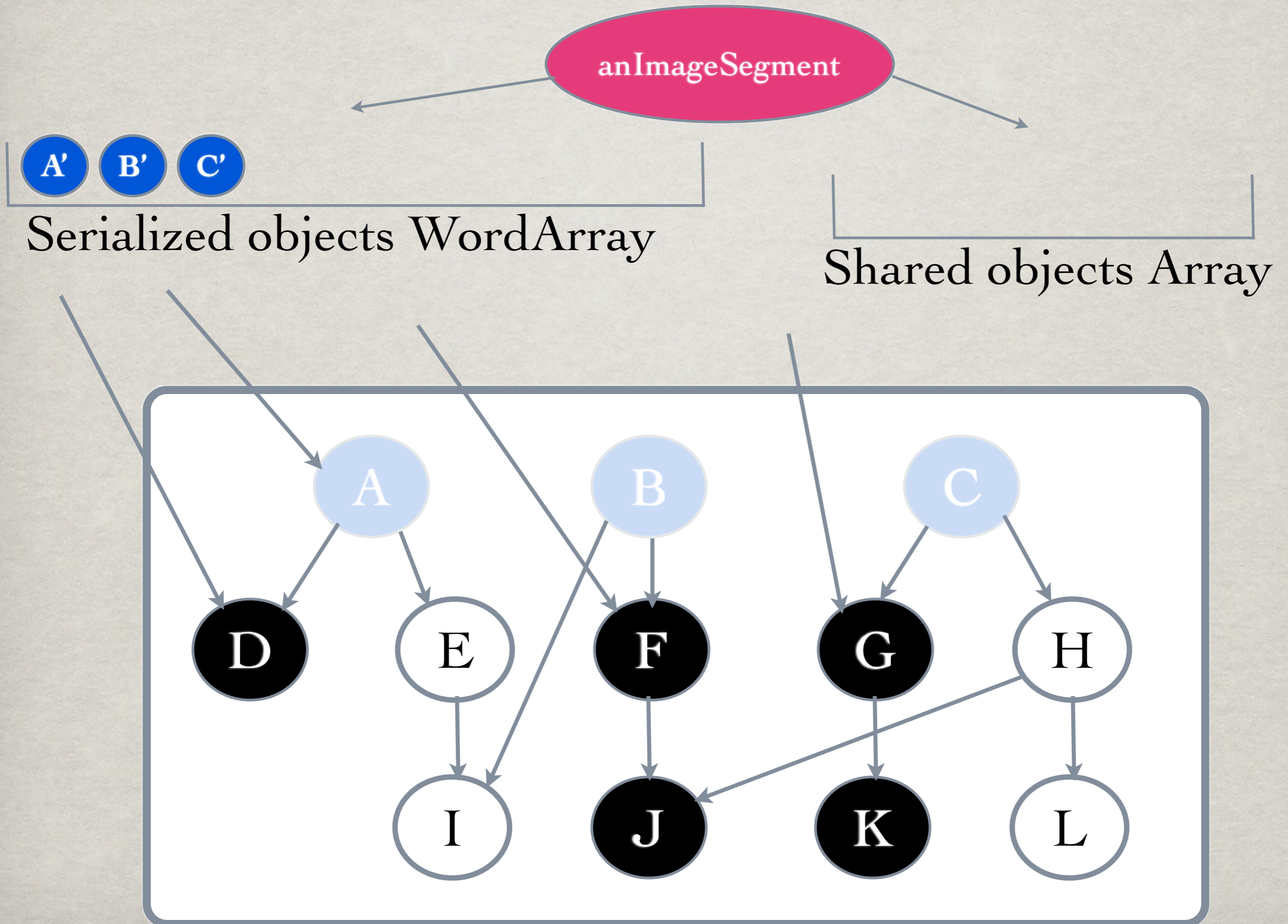
SUBGRAPH TRAVERSE



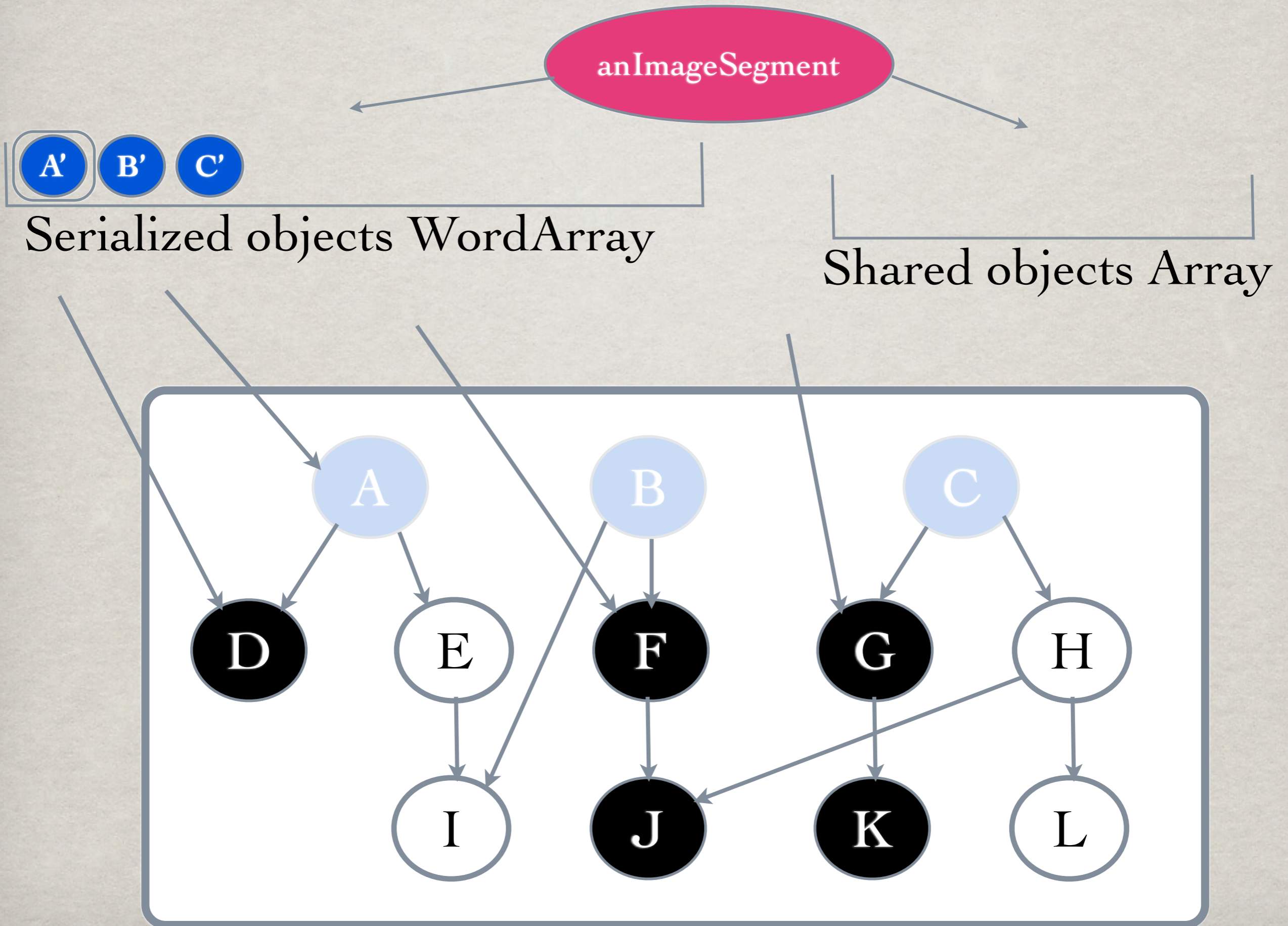
SUBGRAPH TRAVERSE



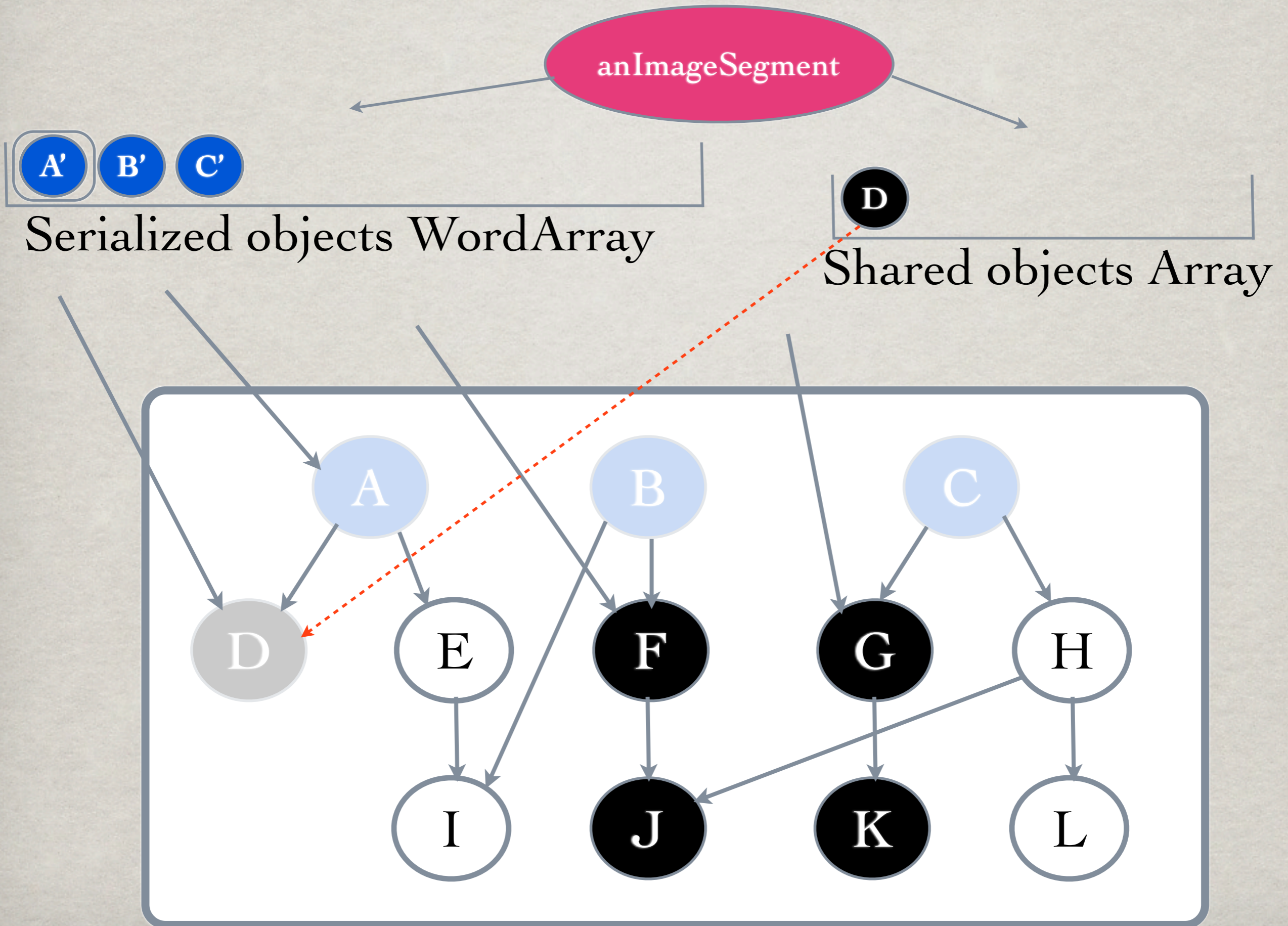
SUBGRAPH TRAVERSE



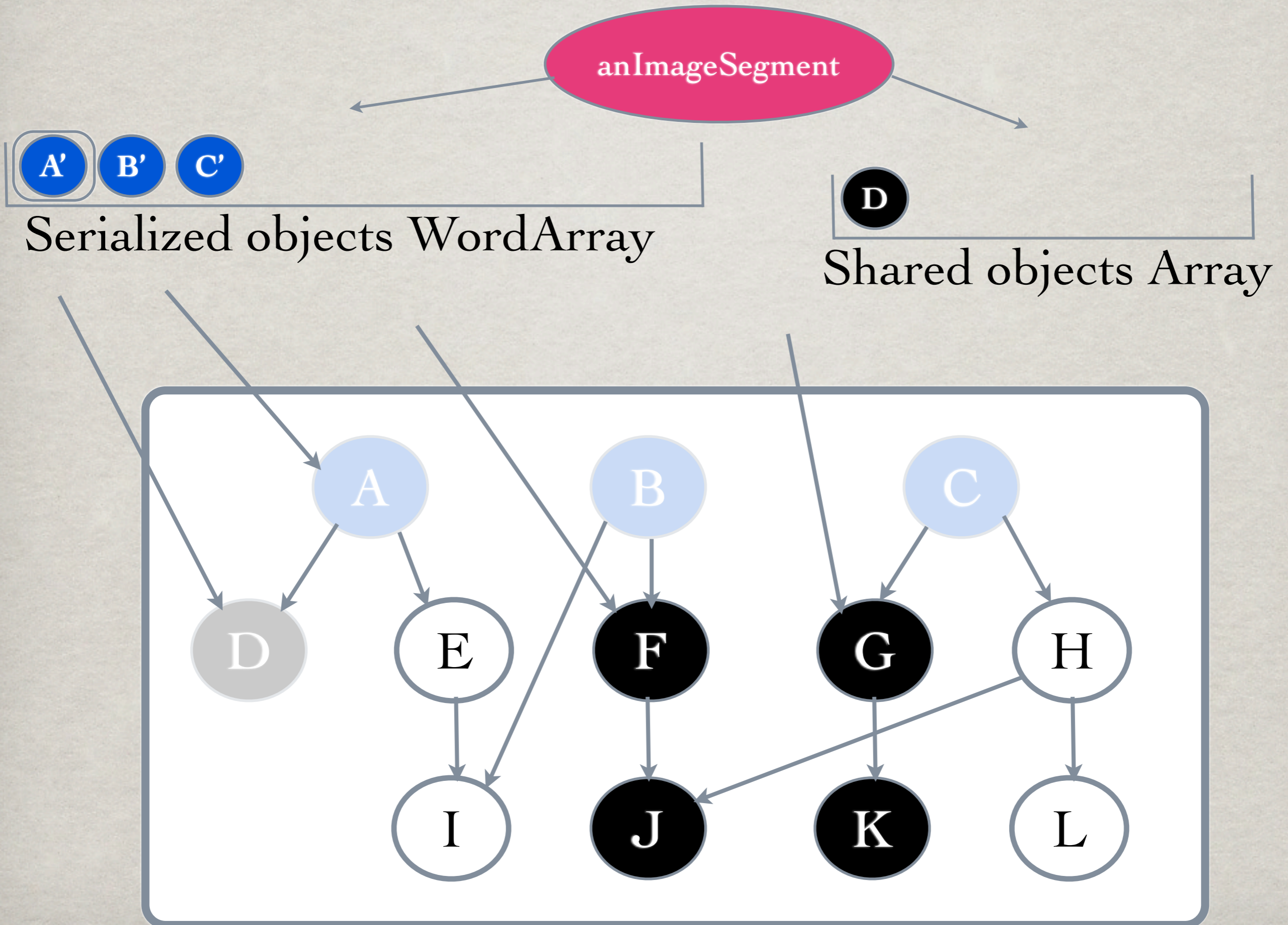
SUBGRAPH TRAVERSE



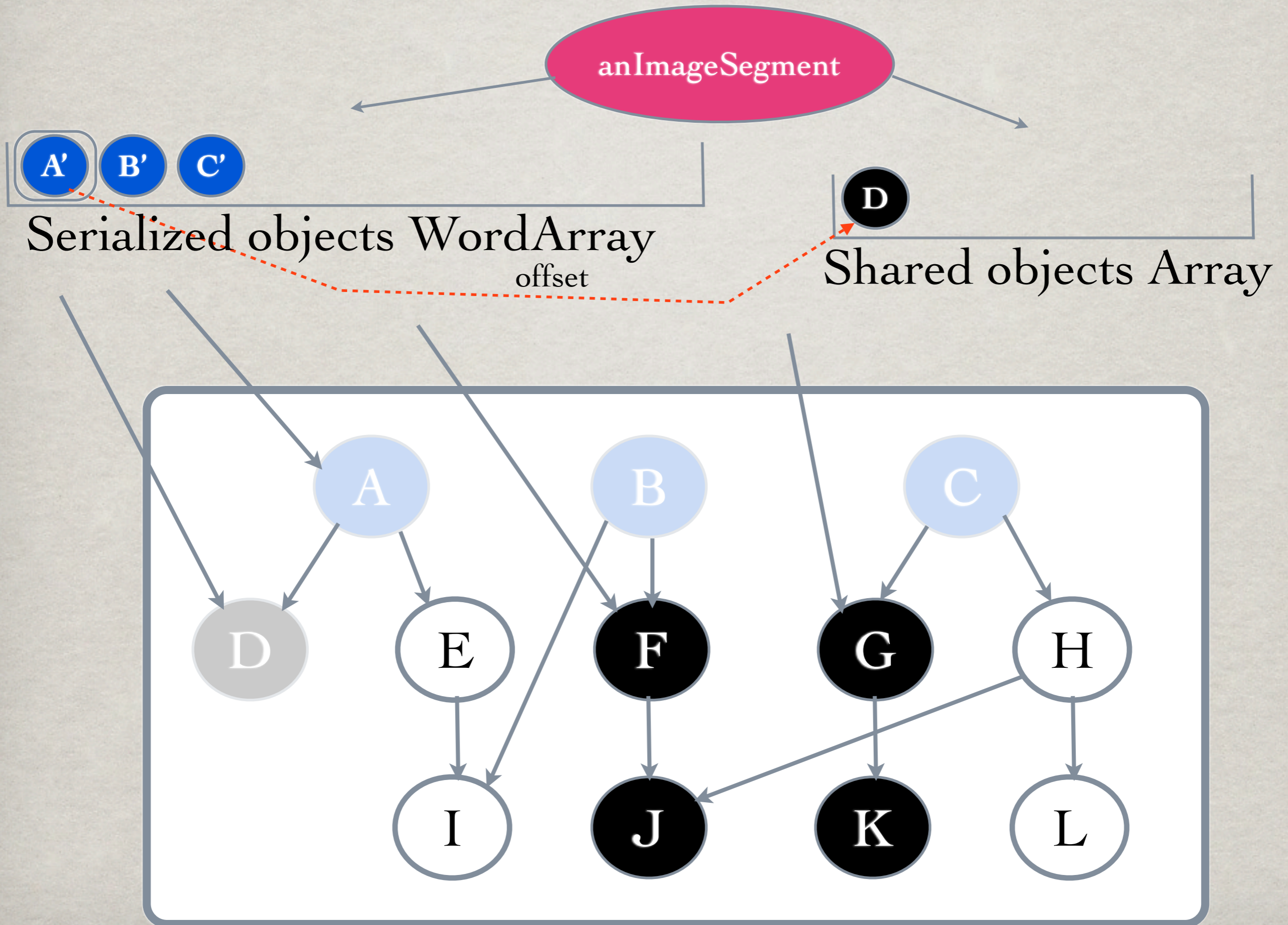
SUBGRAPH TRAVERSE



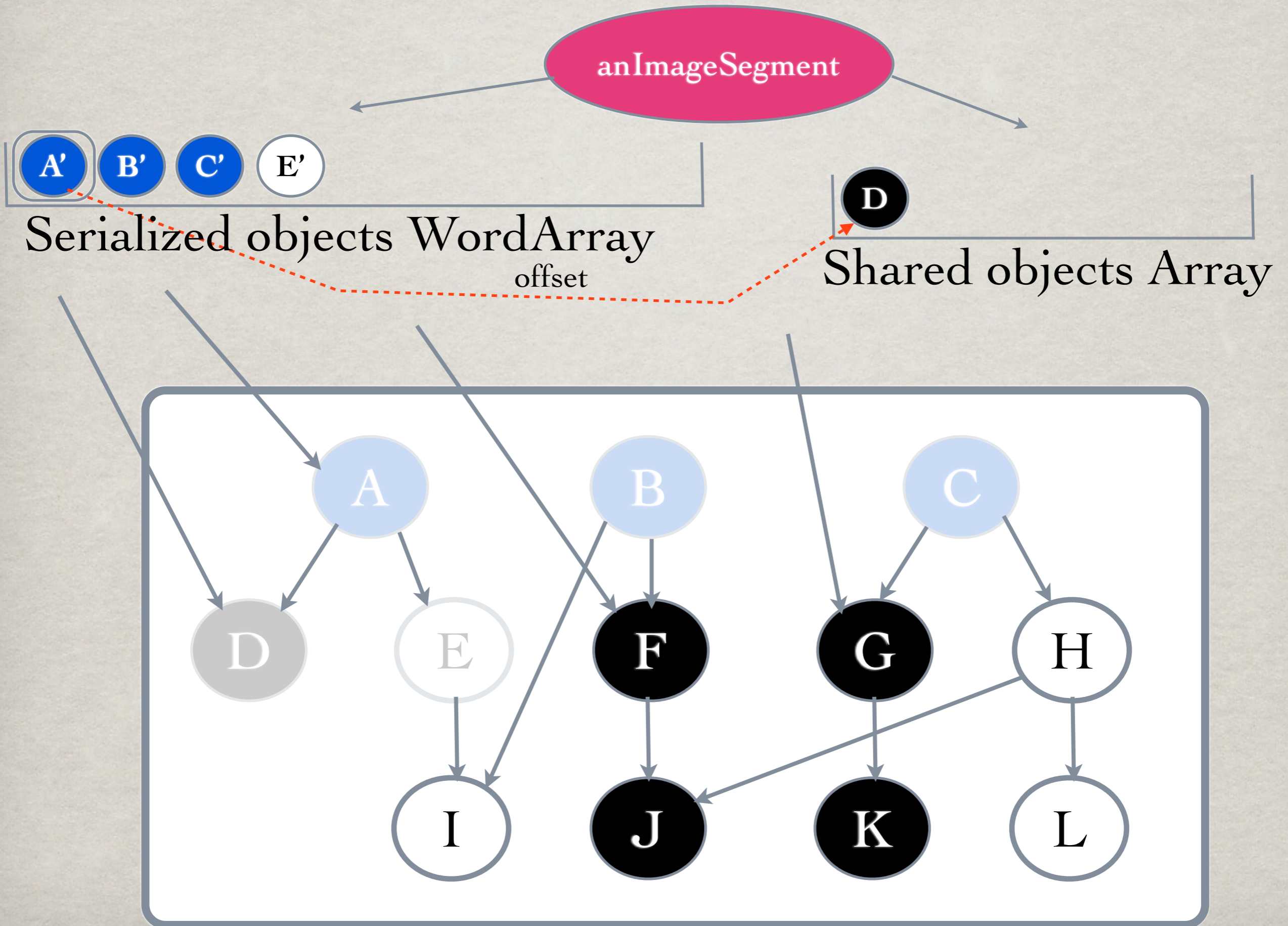
SUBGRAPH TRAVERSE



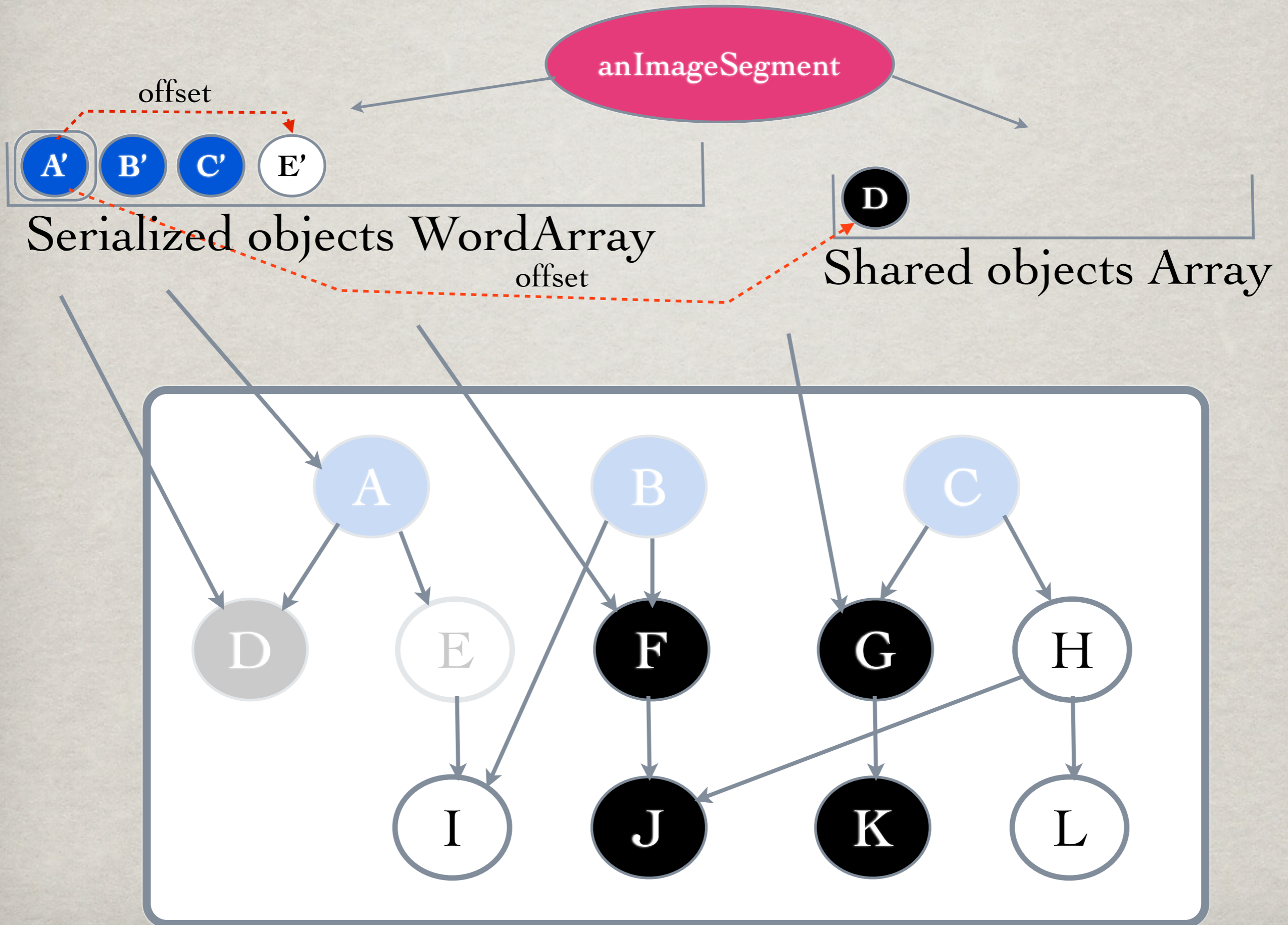
SUBGRAPH TRAVERSE



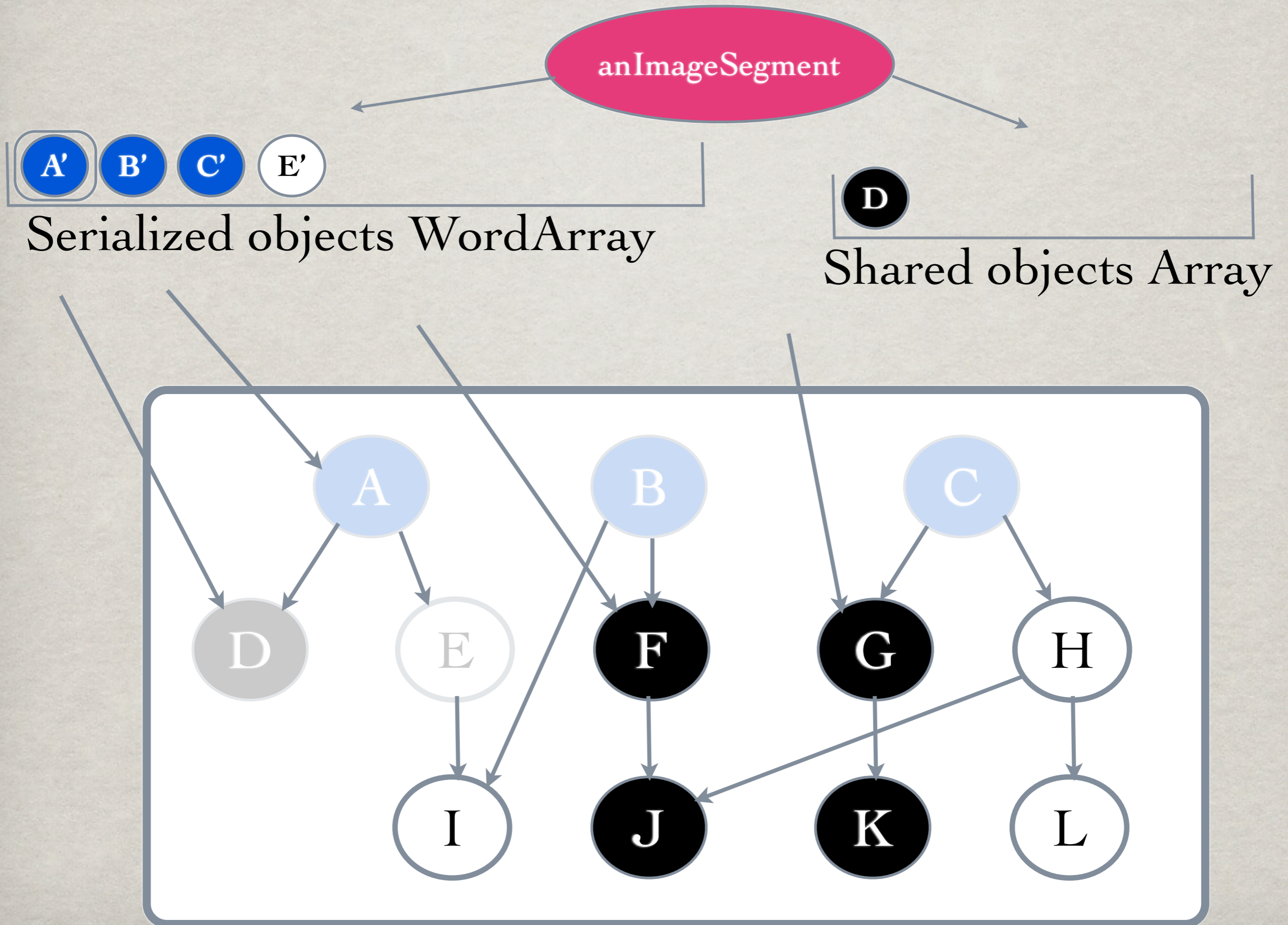
SUBGRAPH TRAVERSE



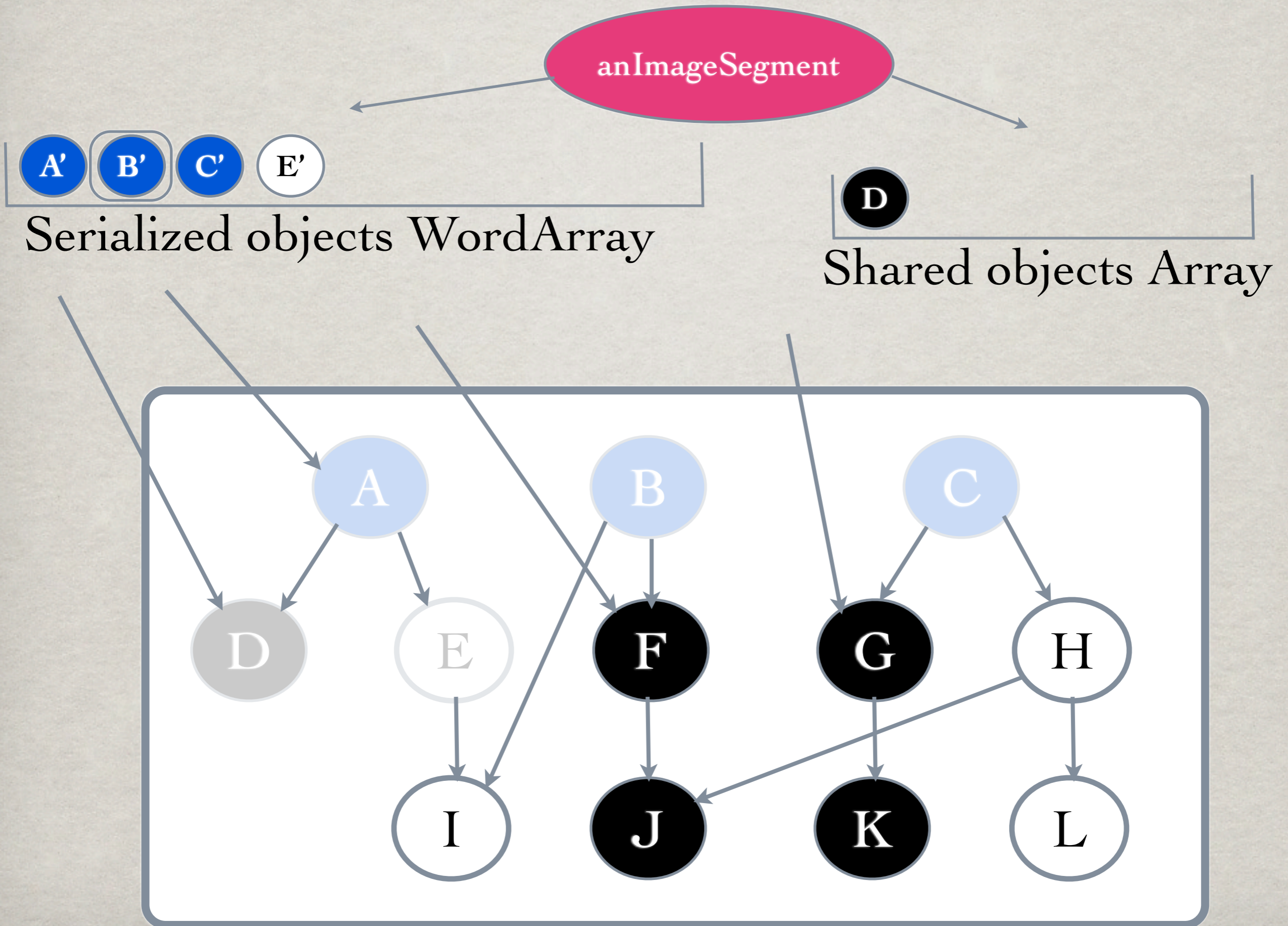
SUBGRAPH TRAVERSE



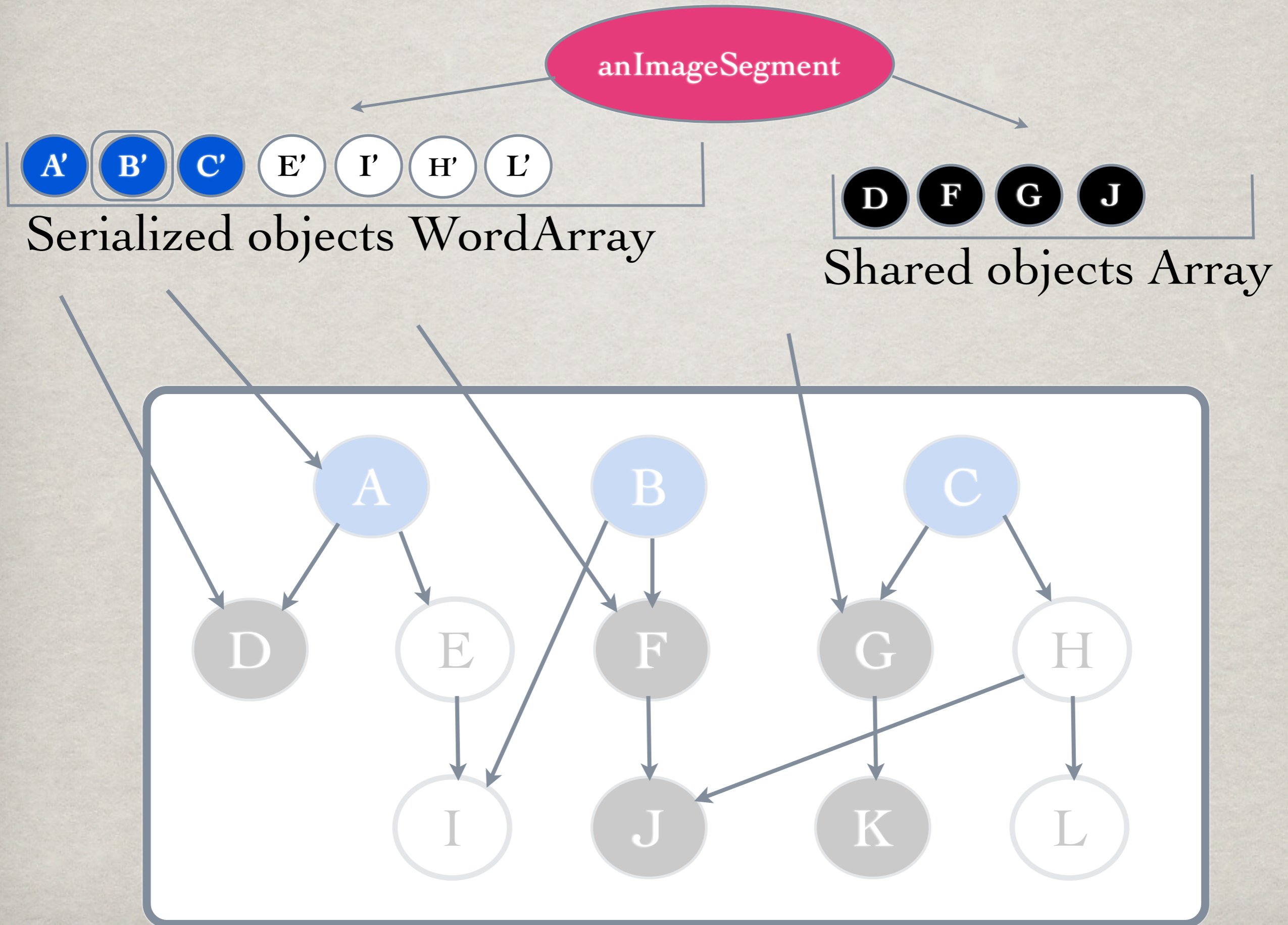
SUBGRAPH TRAVERSE

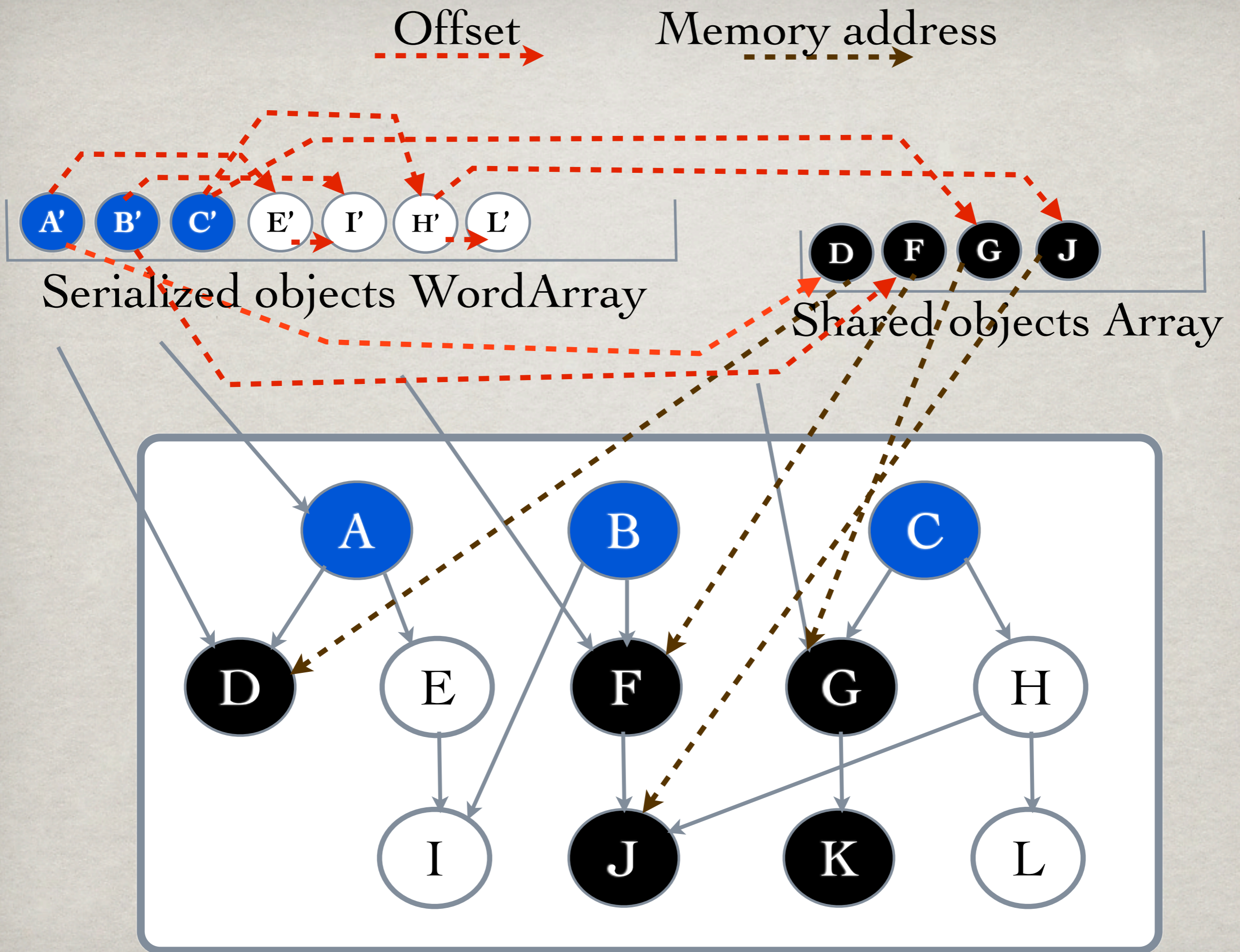


SUBGRAPH TRAVERSE

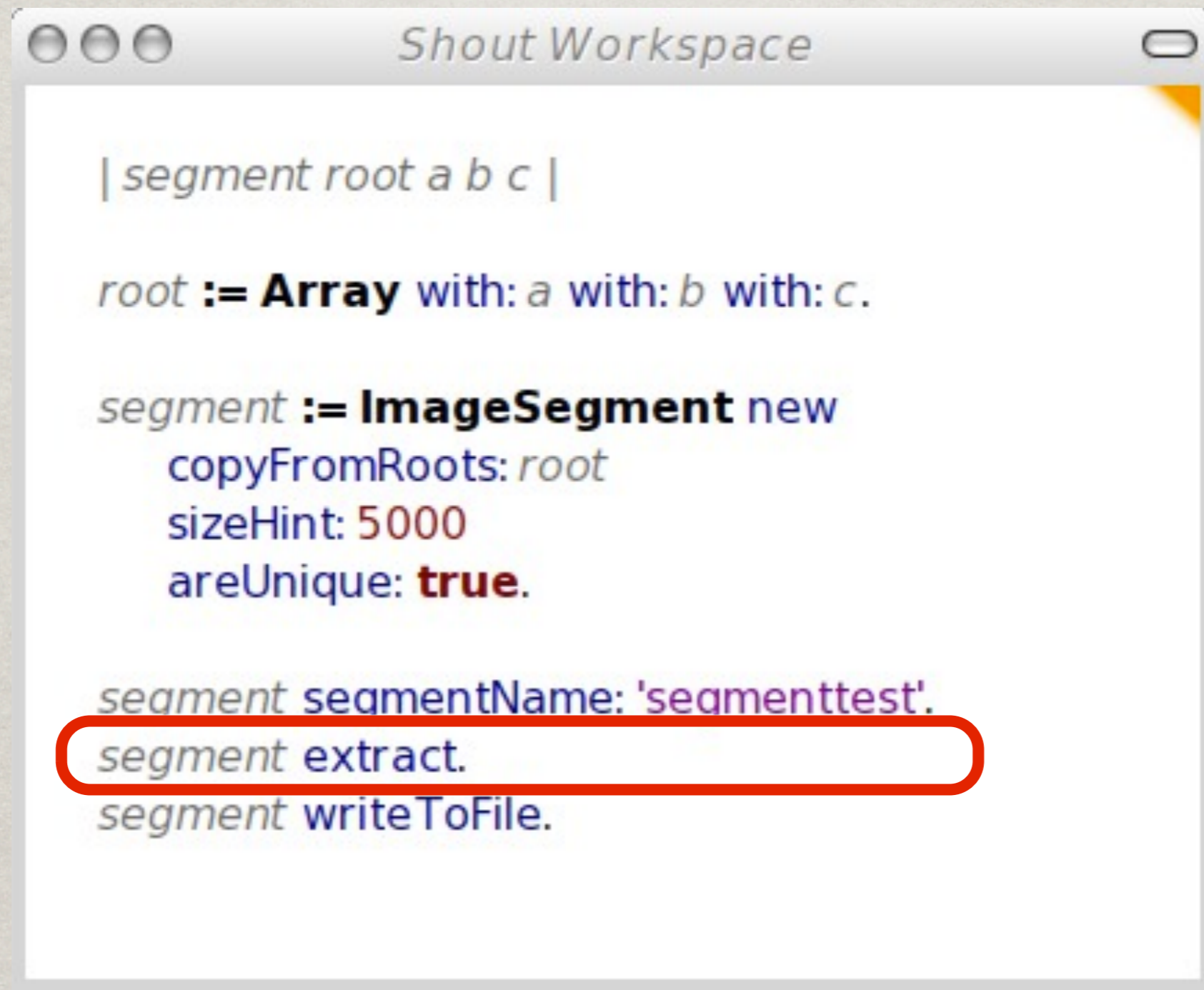


SUBGRAPH TRAVERSE





ROOTS REPLACE

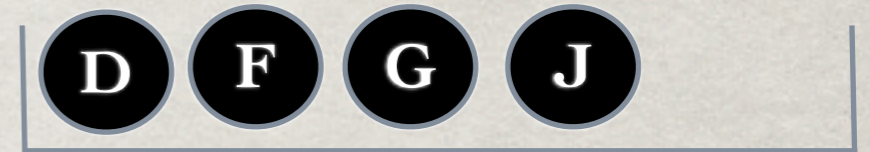


```
| segment root a b c |  
  
root := Array with: a with: b with: c.  
  
segment := ImageSegment new  
    copyFromRoots: root  
    sizeHint: 5000  
    areUnique: true.  
  
segment segmentName: 'segmenttest'.  
segment extract.  
segment writeToFile.
```

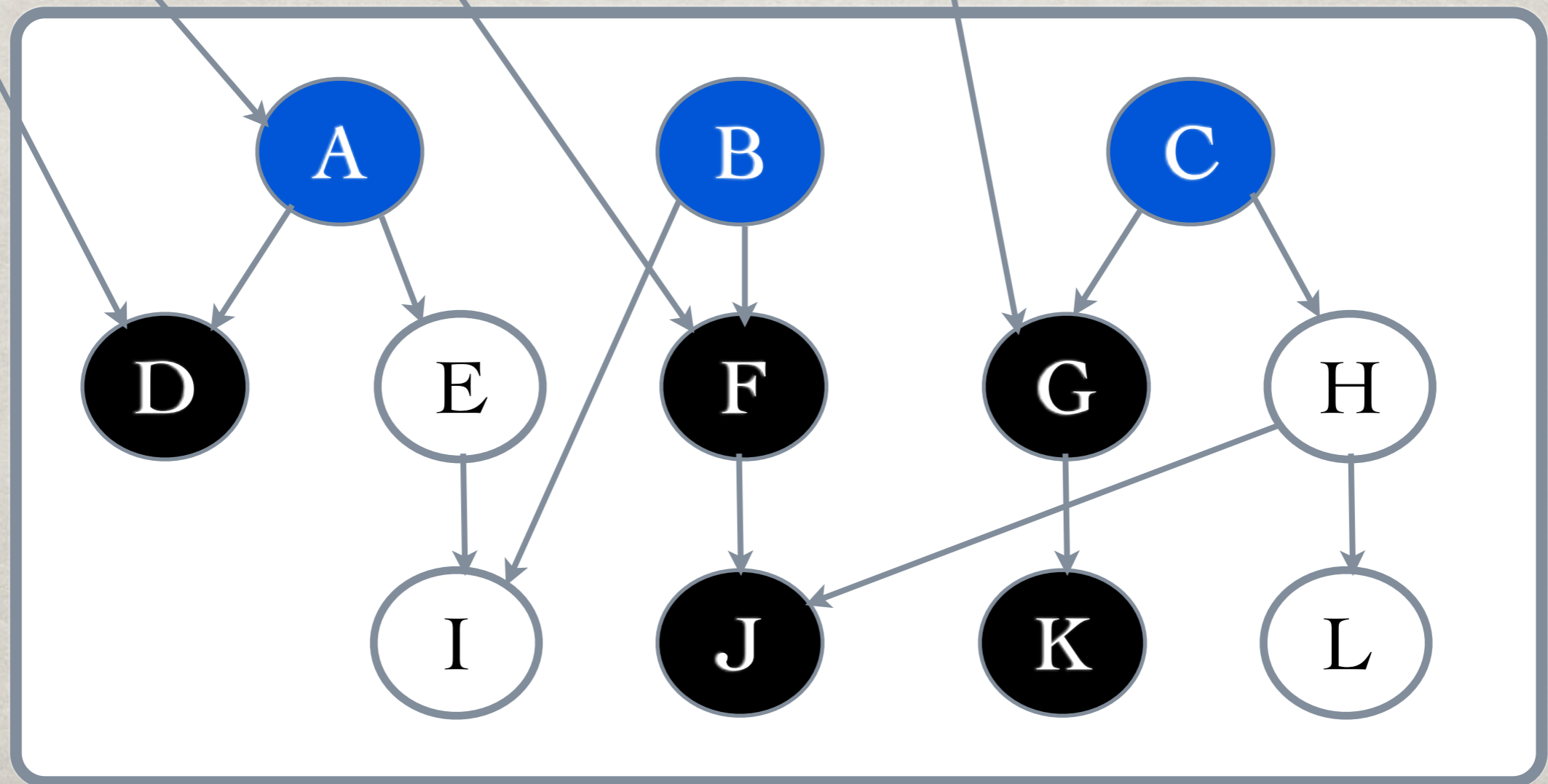
anImageSegment

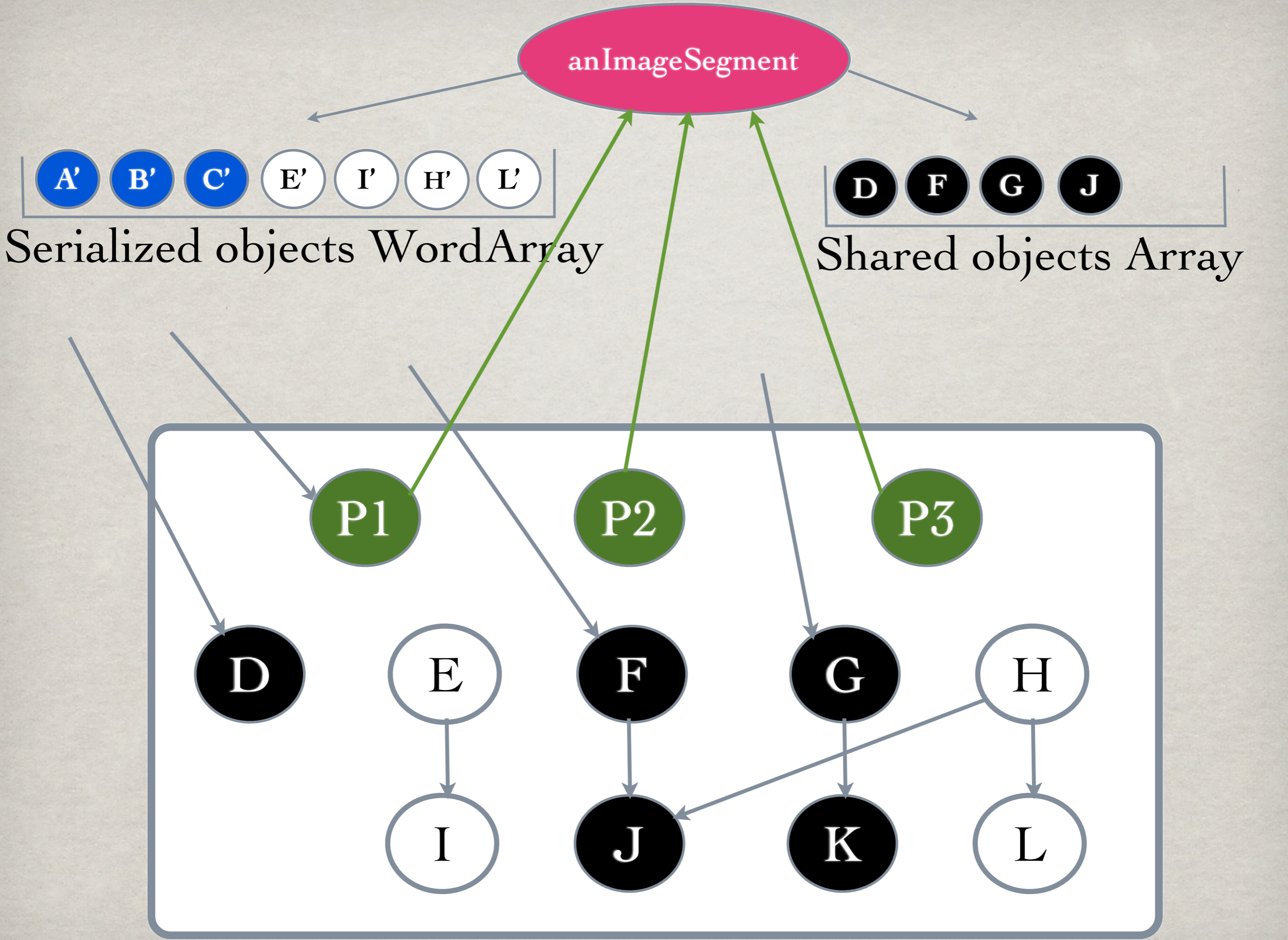


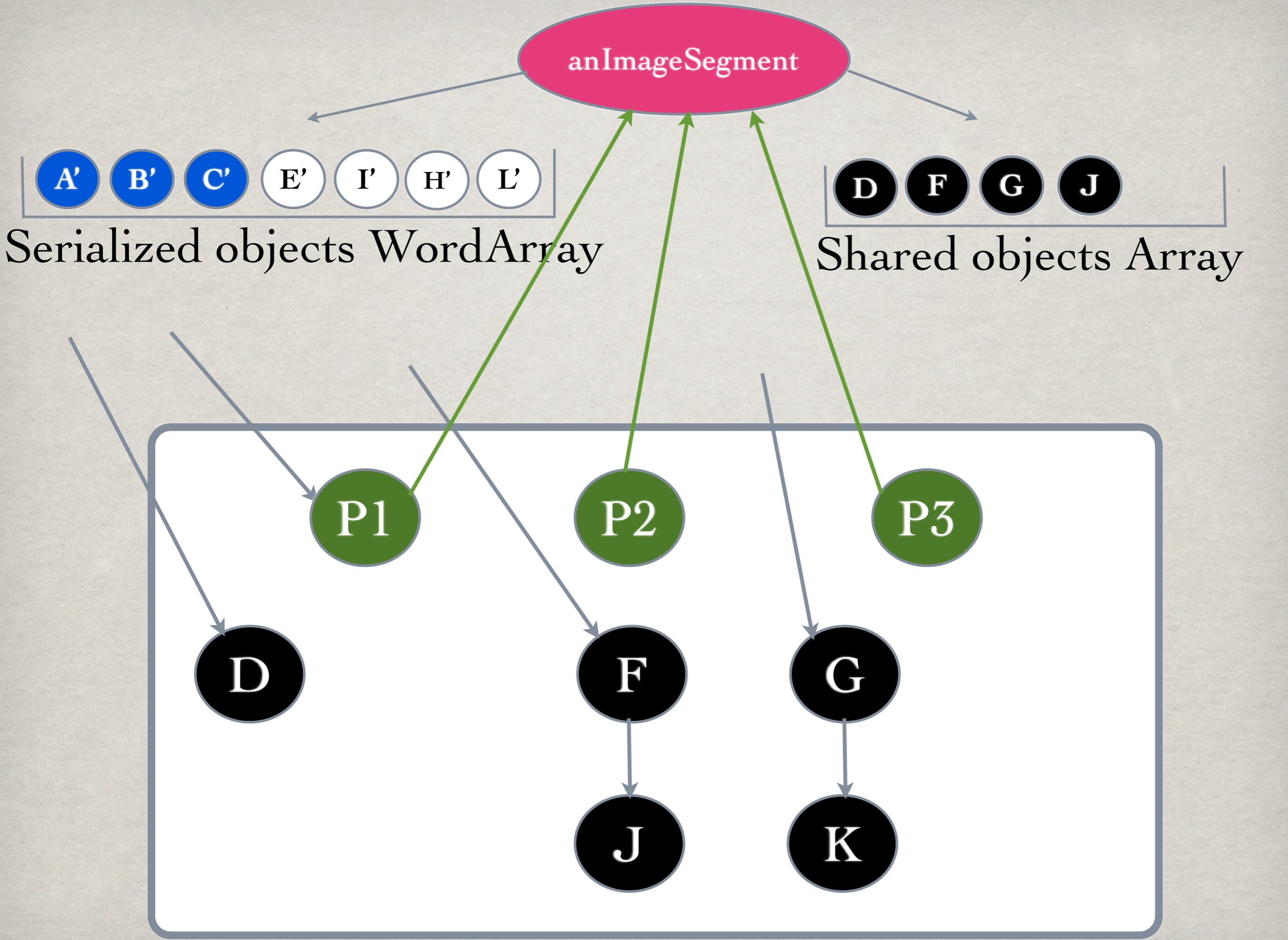
Serialized objects WordArray



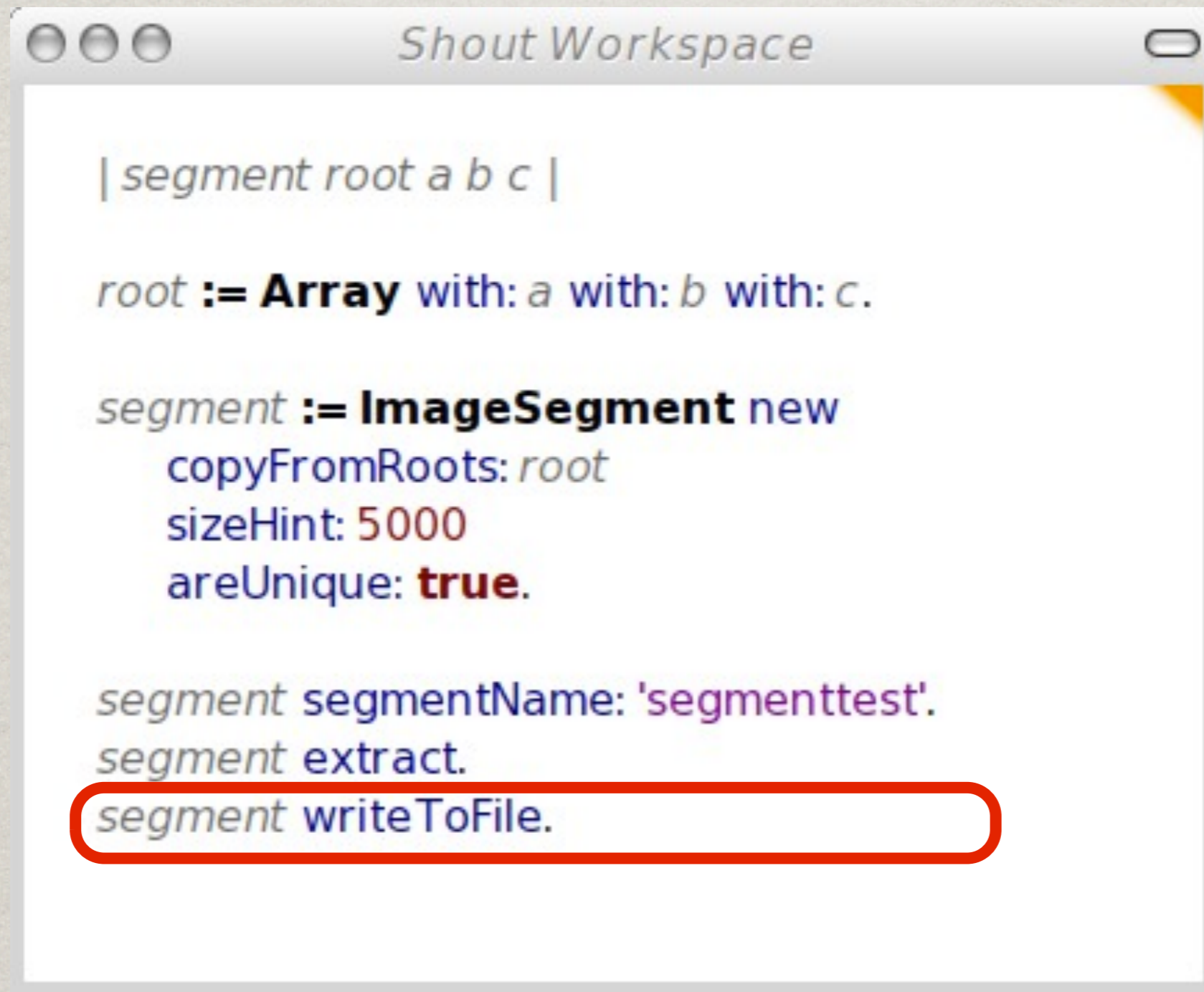
Shared objects Array





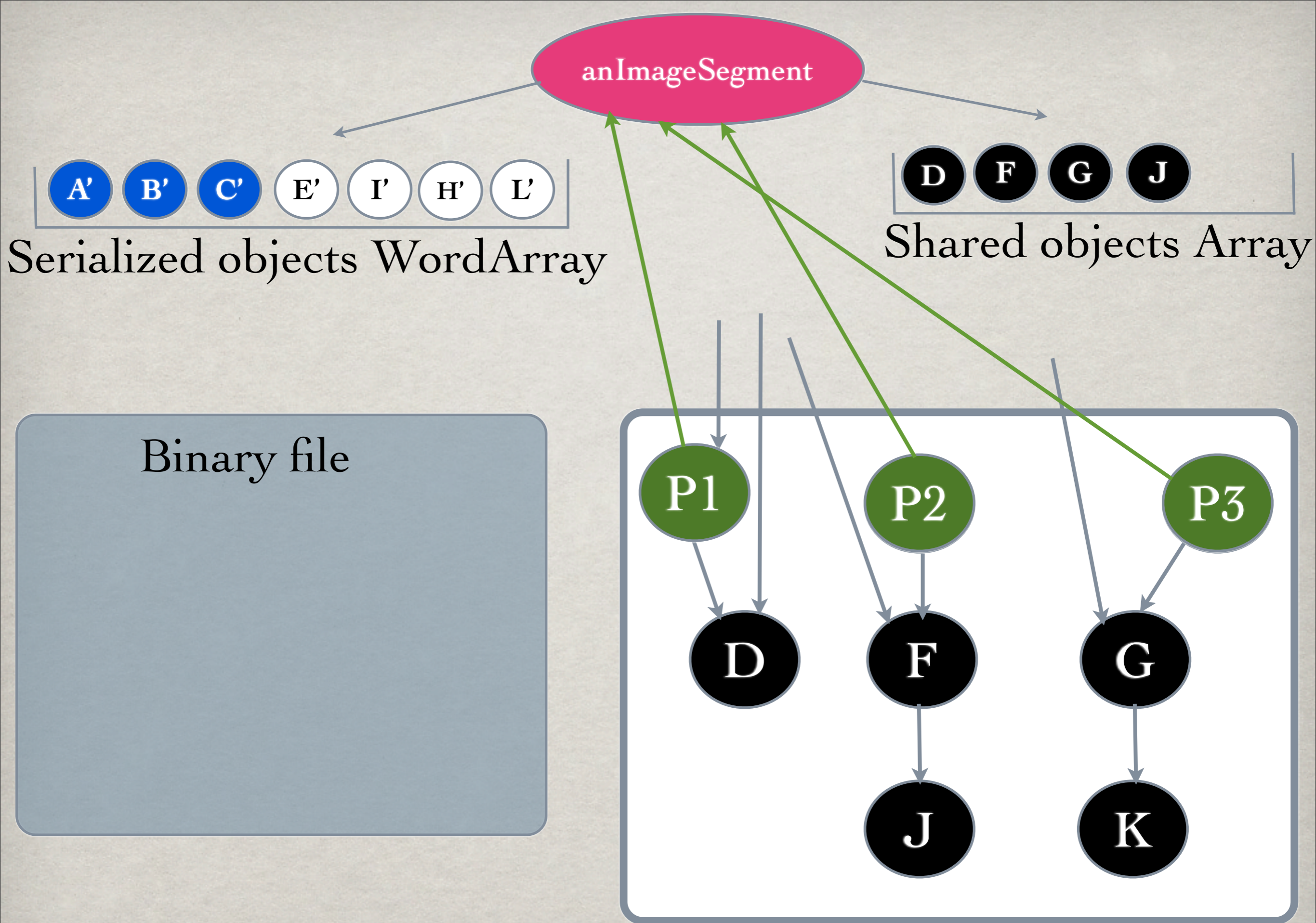


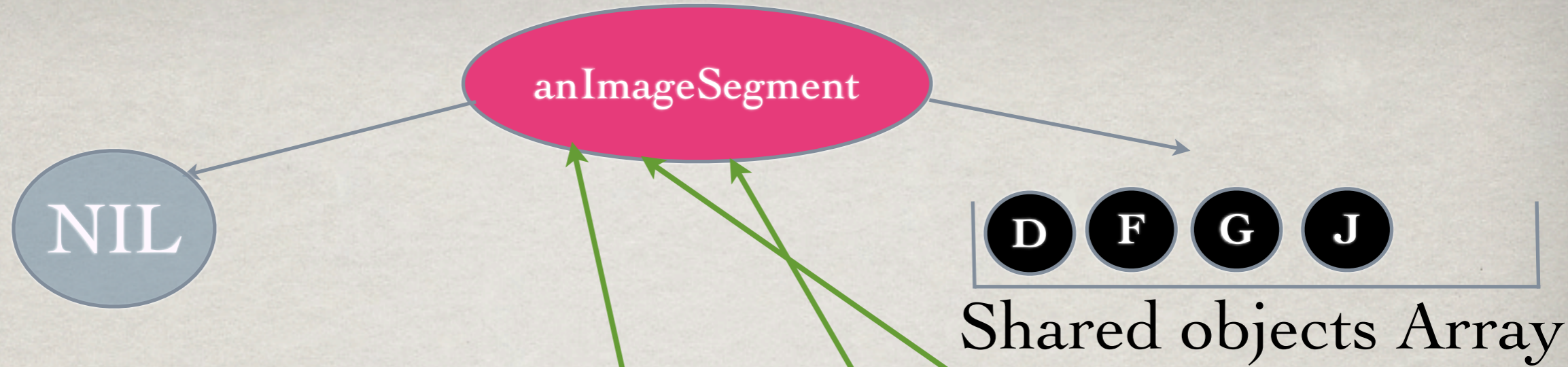
WRITE TO FILE



A screenshot of a window titled "Shout Workspace". The window contains several lines of code. The last line, `segment writeToFile.`, is highlighted with a red rounded rectangle.

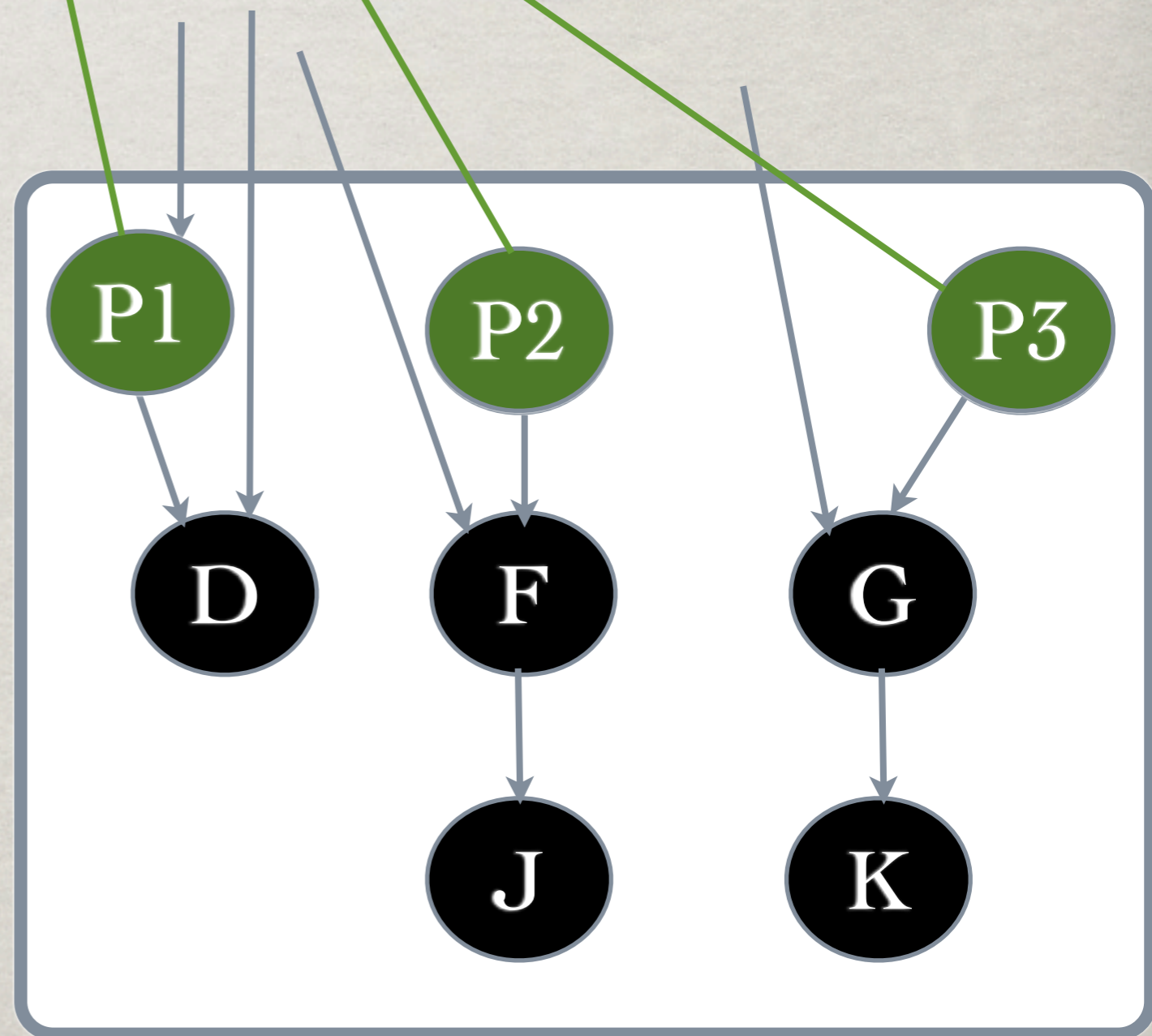
```
| segment root a b c |  
  
root := Array with: a with: b with: c.  
  
segment := ImageSegment new  
  copyFromRoots: root  
  sizeHint: 5000  
  areUnique: true.  
  
segment segmentName: 'segmenttest'.  
segment extract.  
segment writeToFile.
```





Binary file

A' **B'** **C'** **E'** **I'** **H'** **L'**



IMAGESEGMENT CONCLUSIONS

- ✓ Good speed.
- ✓ Graph traverse is done in VM side.
- ✓ Good use of GC facilities.
- ✗ You have to be aware of shared objects.
- ✗ Bad granularity level.
- ✗ Implicit needed information in object graphs.

Thanks!

Mariano Martinez Peck
marianopeck@gmail.com

