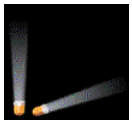
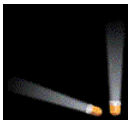
Simulated Pointers

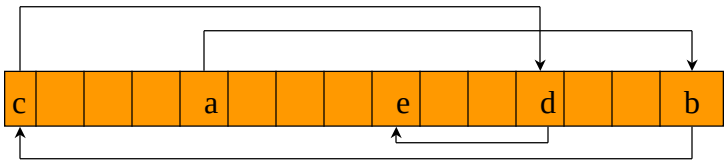




Limitations Of Java Pointers

- May be used for internal data structures only.
- Data structure backup requires serialization and deserialization.
- No arithmetic.

Simulated-Pointer Memory Layout



Data structure memory is an array, and each array position has an element field (type Object) and a next field (type int).

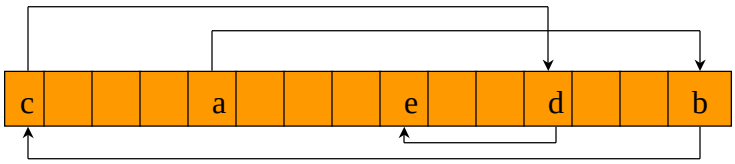
Node Representation

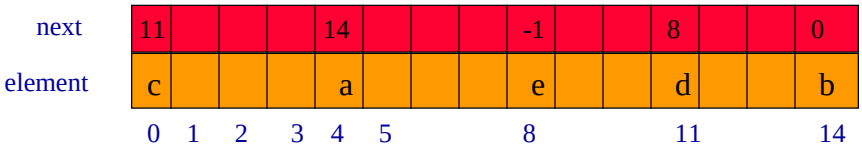
```
package dataStructures;

class SimulatedNode
{
  // package visible data members
  Object element;
  int next;

   // constructors not shown
}
```

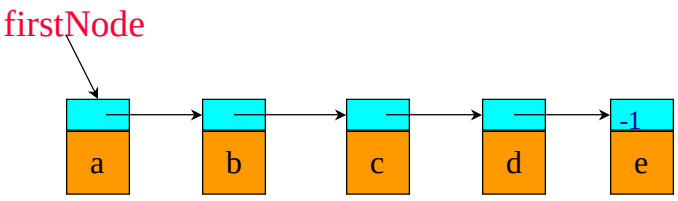
How It All Looks





firstNode = 4

Still Drawn The Same



Memory Management

Linked system (Java or simulated pointer) requires:

- a way to keep track of the memory that is not in use (i.e., a storage pool)
- way to allocate a node

  Java has the method new
- way to free a node that is no longer in use

  Java uses garbage collection

Garbage Collection

The system determines which nodes/memory are not in use and returns these nodes (this memory) to the pool of free storage.

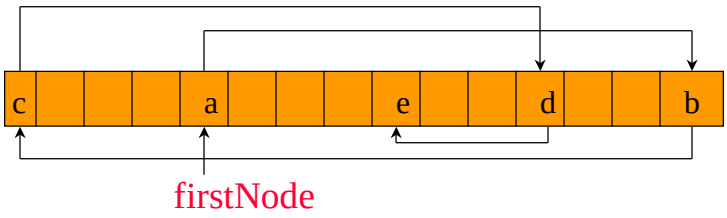
This is done in two or three steps:

Mark nodes that are in use.

Compact free space (optional).

Move free nodes to storage pool.

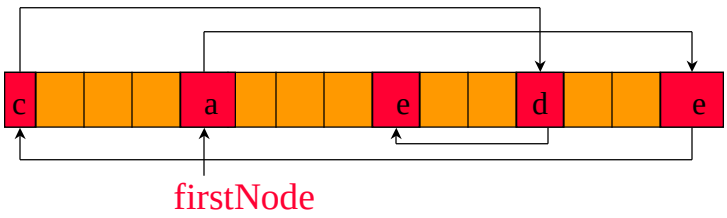
Marking



Unmark all nodes (set all mark bits to false).

Start at each program variable that contains a reference, follow all pointers, mark nodes that are reached.

Marking



Start at firstNode and mark all nodes reachable from firstNode.

Repeat for all reference variables.

Compaction

Move all marked nodes (i.e., nodes in use) to one end of memory, updating all pointers as necessary.

Put Free Memory In Storage Pool

Scan memory for unmarked nodes (if no compaction done), otherwise put single free block (unless no free memory) into pool.

Advantages Of Garbage Collection

- Programmer doesn't have to worry about freeing nodes as they become free.
- However, for garbage collection to be effective, we must set reference variables to null when the object being referenced is no longer needed.

Advantages Of Garbage Collection

- Applications may run faster when run on computers that have more memory.

Disadvantage Of Garbage Collection

- Garbage collection time is linear in memory size (not in amount of free memory).

Alternative To Garbage Collection

Provide a method to free/deallocate a node.

e.g., delete method of C++

Now free nodes are always in storage pool.

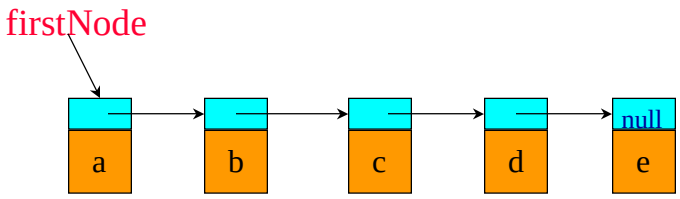
Advantage Of Alternative

- Time to free nodes is proportional to number of nodes being freed and not to total memory size.

Disadvantages Of Alternative

- User must write methods to free data structure nodes.
- Time is spent freeing nodes that may not be reused.
- Application run time does not improve with increase in memory size.

Storage Pool Organization When All Nodes Have Same Size



- Maintain a chain of free nodes
- Allocate from front of chain
- Add node that is freed to chain front

Simulated-Pointer Memory Management

```
/** memory management for simulated pointer classes */
package dataStructures;
import utilities.*;
public class SimulatedSpace1
{
  // data members
  private int firstNode;
  SimulatedNode [] node;  // package visible

  // constructor and other methods come here
}
```

Constructor

```
public SimulatedSpace1(int numberOfNodes)
 {
   node = new SimulatedNode [numberOfNodes];

   // create nodes and link into a chain
   for (int i = 0; i < numberOfNodes - 1; i++)
     node[i] = new SimulatedNode(i + 1);

   // last node of array and chain
   node[numberOfNodes - 1] = new SimulatedNode(-1);
   // firstNode has the default initial value 0
 }
```

Allocate A Node

```
public int allocateNode(Object element, int next)
  {// Allocate a free node and set its fields.
    if (firstNode == -1)
    {   // double number of nodes, code omitted
    }

    int i = firstNode;              // allocate first node
    firstNode = node[i].next;  // firstNode points to next free node
    node[i].element = element;
    node[i].next = next;
    return i;
  }
```

Free A Node

```
public void deallocateNode(int i)
 {// Free node i.
   // make i first node on free space list
   node[i].next = firstNode;
   firstNode = i;

   // remove element reference so that space can be garbage
   //  collected
   node[i].element = null;
 }
```

Simulated Pointers

- Can allocate a chain of nodes without having to relink.
- Can free a chain of nodes in O(1) time when first and last nodes of chain are known.

Simulated Pointers

- Don’t use unless you see a clear advantage to using simulated pointers over Java references.