Difference between Cheney's algorithm and Mark compact algorithm

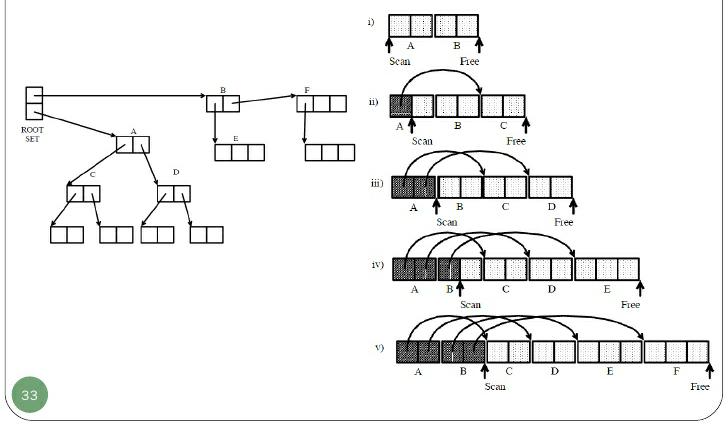
|  |  |
| --- | --- |
| **Cheney's Algorithm** | **Mark-compact algorithm** |
| Traverse data breadth first, copying objects from from-space to to-space | Starting from roots, mark all reachable objects by using a depth-first-search pointer traversal |
| Twice the heap size | Heap fragmentation |
| Need forwarding pointer | Need recursion or work queue |
| Requires copying | Needs liveness flag |
| Caches are messed up | Must be able to find all dead objects |

Cheney's algorithm(Copying garbage collection) is better than mark-sweep for two reasons.

1. it compacts memory, and hence avoids any fragmentation.
2. its running time is proportional to the amount of live memory, **not** the size of the heap.

**Cheney's Algorithm:**

Cheney's Algorithm uses a very simple allocation algorithm and has no need for stack. More importantly, its run time depends on the number of live objects, not on the heap size. Forwarding pointers can be done in breadth-first-search, using Cheney's Algorithm. Compared to depth-first-search, breadth-first-search has better locality of reference:

:

**ALGORITHM EXPLANATION**:

* Start at the roots & traverse the reachable data
* Copy reachable data from the active heap (from-space) to the other

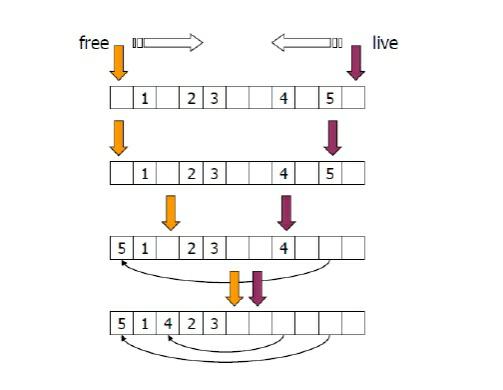
heap (to-space)

* Dead objects are left behind in from space
* Heaps switch roles

**Mark – compact algorithm:**

Garbages are squeezed to the end of the memory. First, reachable objects are marked, then a compacting step relocates the reachable marked objects towards the beginning of the heap area.The live objects in the heap in the same fashion as the [mark-sweep algorithm](https://en.m.wikipedia.org/wiki/Mark-sweep_algorithm), the heap will often be [fragmented](https://en.m.wikipedia.org/wiki/Fragmentation_(computer)). Two types of algorithm

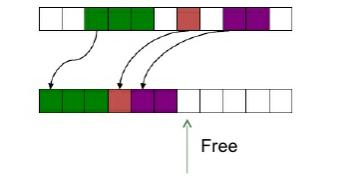
1. **Two finger algorithm**

****

**ALGORITHM EXPLANATION**:

* free- goes over heap from bottom searching from free or empty objects
* Live- goes over heap from top downwards searching for live objects.
* When object is moved a pointer to its new location is left at its old location.

**2**.**Lisp2 algorithm**

****

**ALGORITHM EXPLANATION:**

* keep new address in an additional object field.
* Modify the pointer
* Two pointer (free and live) run from bottom. Live object are moved to free space keeping their original order.