Programming Languages HW 3

Jiraphon Yenphraphai (jy3694)

July 24, 2022

1 Garbage Collection

1.1 The heap after mark() step

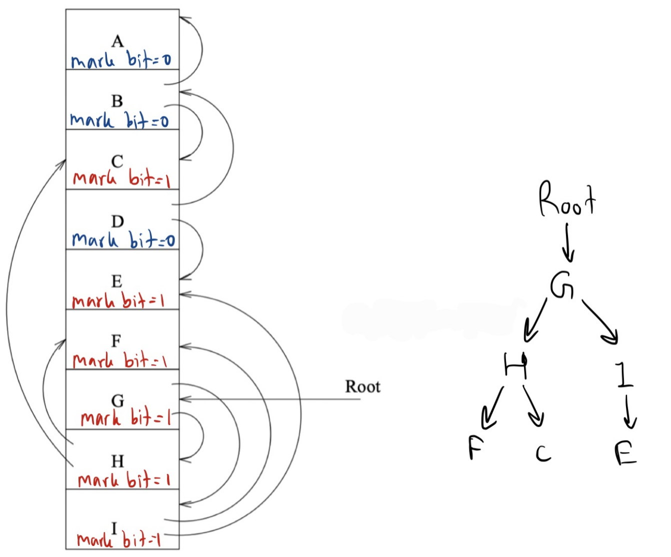

Figure 1: Left figure shows the heap after mark() step (Left figure), Right figure shows the graph during traverse

Fig. 1 shows the heap after mark() step has been completed. During this step, the collector traverses the heap and sets the mark bit of each object encountered to be 1. The object that is not reachable from the root will have mark bit 0. Note that the mark bit is shown in the fig. 1. If we traverse the heap from the root, we will get the graph shown on the right figure of Fig. 1.

1.2 The heap after the garbage collection is complete

After the mark phase, garbage collection will call sweep() to clear an object whose mark bit is 0 and insert it to free list. The object, whose mark bit is 1, will still be in the heap and the mark bit is set to 0 for the next garbage collection run, as shown in fig. 2. The free list is a linked list whose data in each element is the removed object's address and its size. In this case, object A, B, and D are removed, thus, their addresses and sizes are on the free list as in fig. 2.

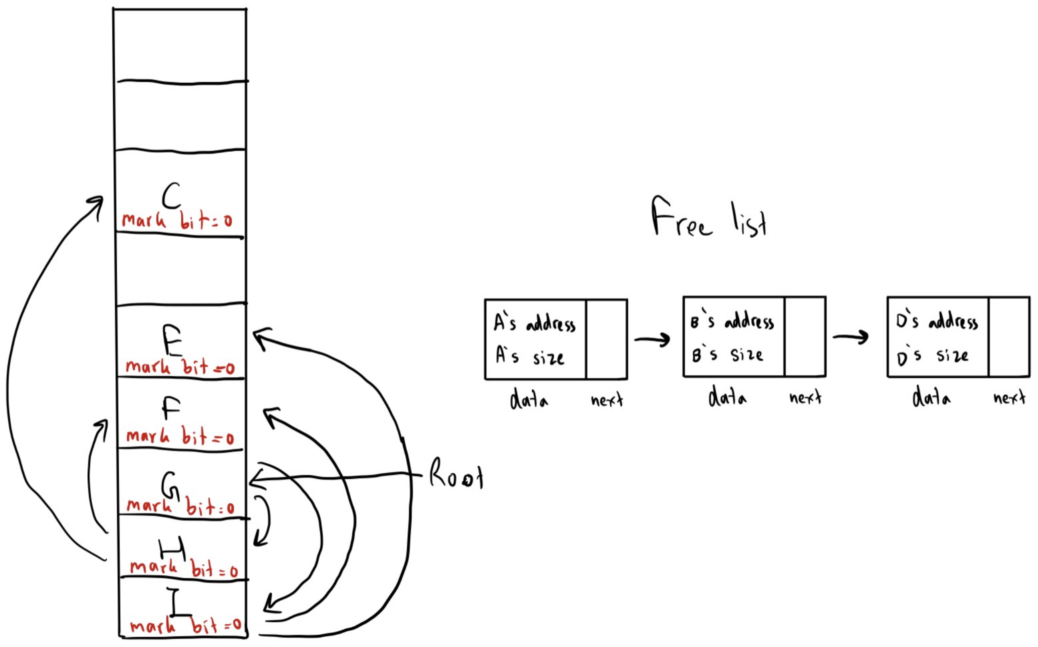

Figure 2: Left figure shows the heap after the garbage collection is complete, Right figure shows the free list

2 Garbage Collection - 2

2.1 2.1

In this situation, both Mark/Sweep and Copying GC can be used as garbage collectors.

Let's first consider the mark() step. Because we ignore the tracing through root pointer running time, this step would result in $\mathcal{O}(1)$. During sweep() step, we have to traverse through every object in the heap regardless of the survival rate. Assume that there are N objects in the heap and sweeping an object costs M units of time. Thus, the time complexity is MN.

Now, let's consider copying GC. Traverse step will copy on the object that is alive. According to the question, only $0.08N$ objects are live data. These objects will be copied with time complexity $20M$. The total running time is $1.6MN$. We can see that in this case, Mark/Sweep is the fastest.

2.2 2.2

Again in this situation, both Mark/Sweep and Copying GC can be used as garbage collectors.

The Mark/Sweep consumes the same time complexity as the above question as it is independent of the survival rate. Thus, the Mark/Sweep takes MN.

Likewise, copying GC can use the same reasoning as the above question. There will be $0.15N$ live objects. These objects will be copied with time complexity $5M$. The total running time is $0.75MN$. We can see that in this case, Copying GC is the fastest.

2.3 2.3

Copying GC partitions the heap into 2 spaces: FROM and TO. As there is no additional heap space is available, we cannot use copying GC. In this situation, only Mark/Sweep can be used.

3 Memory Allocation

3.1 3.1

Assume that the free list available in this case is $[4, 3, 2, 1]$ and the sequence of requests is $[1, 2, 3, 4]$. Note that the element of free list is the available size.

First-fit method allocates 1,2,3,4. The free list is $[4, 3, 2, 1] \rightarrow^1 [3, 3, 2, 1] \rightarrow^2 [1, 3, 2, 1] \rightarrow^3 [1, 0, 2, 1] \rightarrow^4 ERROR$. First-fit method cannot allocate the last sequence 4.

Best-fit method allocates 1,2,3,4. The free list is $[4,3,2,1] \rightarrow^1 [4,3,2,0] \rightarrow^2 [4,3,0,0] \rightarrow^3 [4,0,0,0] \rightarrow^4 [0,0,0,0]$. Best-fit method can allocate all sequence.

Worst-fit method allocates 1,2,3,4. The free list is $[4,3,2,1] \rightarrow^1 [3,3,2,1] \rightarrow^2 [1,3,2,1] \rightarrow^3 [1,0,2,1] \rightarrow^4$ $ERROR$. Worst-fit method cannot allocate the last sequence 4.

3.2 3.2

Assume that the free list available in this case is $[12,10,9,15]$ and the sequence of requests is $[3,8,13,10,8]$.

First-fit method allocates 3,8,13,10,8. The free list is $[12,10,9,15] \rightarrow^3 [9,10,9,15] \rightarrow^8 [1,10,9,15] \rightarrow^{13} [1,10,9,2] \rightarrow^{10} [1,0,9,2] \rightarrow^8 [1,0,1,2]$. First-fit method can allocate all sequence.

Best-fit method allocates 3,8,13,10,8. The free list is $[12,10,9,15] \rightarrow^3 [12,10,6,15] \rightarrow^8 [12,2,6,15] \rightarrow^{13} [12,2,6,2] \rightarrow^{10} [2,2,6,2] \rightarrow^8$ $ERROR$. Best-fit method cannot allocate the last sequence 8.

Worst-fit method allocates 3,8,13,10,8. The free list is $[12,10,9,15] \rightarrow^3 [12,10,9,12] \rightarrow^8 [4,10,9,12] \rightarrow^{13}$ $ERROR$. Best-fit method cannot allocate sequence 13, 10 and 8.

3.3 3.3

Assume that the free list available in this case is $[9,6,7,10,3]$ and the sequence of requests is $[3,2,5,7,5,6]$.

First-fit method allocates 3,2,5,7,5,6. The free list is $[9,6,7,10,3] \rightarrow^3 [6,6,7,10,3] \rightarrow^2 [4,6,7,10,3] \rightarrow^5 [4,1,7,10,3] \rightarrow^7 [4,1,0,10,3] \rightarrow^5 [4,1,0,5,3] \rightarrow^6$ $ERROR$. First-fit method cannot allocate the last sequence 6.

Best-fit method allocates 3,2,5,7,5,6. The free list is $[9,6,7,10,3] \rightarrow^3 [9,6,7,10,0] \rightarrow^2 [9,4,7,10,0] \rightarrow^5 [9,4,2,10,0] \rightarrow^7 [2,4,2,10,0] \rightarrow^5 [2,4,2,5,0] \rightarrow^6$ $ERROR$. Best-fit method cannot allocate the last sequence 6.

Worst-fit method allocates 3,2,5,7,5,6. The free list is $[9,6,7,10,3] \rightarrow^3 [9,6,7,7,3] \rightarrow^2 [7,6,7,7,3] \rightarrow^5 [2,6,7,7,3] \rightarrow^7 [2,6,0,7,3] \rightarrow^5 [2,6,0,2,3] \rightarrow^6 [2,0,0,2,3]$. Worst-fit method can allocate all sequence.

3.4 3.4

Assume that the free list available in this case is $[7,10,8,5]$ and the sequence of requests is $[5,3,3,5,5,5]$.

First-fit method allocates 5,3,3,5,5,5. The free list is $[7,10,8,5] \rightarrow^5 [2,10,8,5] \rightarrow^3 [2,7,8,5] \rightarrow^3 [2,4,8,5] \rightarrow^5 [2,4,3,5] \rightarrow^5 [2,4,3,0] \rightarrow^5$ $ERROR$. First-fit method cannot allocate the last sequence 5.

Best-fit method allocates 5,3,3,5,5,5. The free list is $[7,10,8,5] \rightarrow^5 [7,10,8,0] \rightarrow^3 [4,10,8,0] \rightarrow^3 [1,10,8,0] \rightarrow^5 [1,10,3,0] \rightarrow^5 [1,5,3,0] \rightarrow^5 [1,0,3,0]$. Best-fit method can allocate all sequence.

Worst-fit method allocates 5,3,3,5,5,5. The free list is $[7,10,8,5] \rightarrow^5 [7,5,8,5] \rightarrow^3 [7,5,5,5] \rightarrow^3 [4,5,5,5] \rightarrow^5 [4,0,5,5] \rightarrow^5 [4,0,0,5] \rightarrow^5 [4,0,0,0]$. Worst-fit method can allocate all sequence.