How to collect garbage when heap memory is running out

Ruoyun Tang

Abstract

When the written program is running, it's unavoidable that the heap memory would be allocated for certain data. However, there's no infinite capacity for heap memory, which means we may get the result of OutOfMemory Error. So where to find more memory space? The answer lies in reusing the garbage in the heap memory. Therefore, this paper focused on the situation where heap memory was running out, and collected some online materials about relevant algorithms to solve this problem, mainly Mark and Sweep Algorithm and Reference Counting Algorithm.

Keywords: Heap memory, Garbage Collection Algorithm, Mark and Sweep Algorithm, Reference Counting Algorithm

1 Introduction

Due to the limited capacity of heap memory, programmers will probably encounter the situation where the memory is running out, while not all the allocated memory is being used. In this circumstance, the most effective solution is to release the garbage in the memory and reuse it.[5]

Finding no corresponding function to automate this process, programmers need to finish this work themselves, which is burdensome and will possibly cause many extra problems. The reasons are as follows. First, the garbage may scatter in different places, leading to too many "new" and "delete" statements. Secondly, if the programmer doesn't manage the task well, there's high risk for memory leakage.

For the all the reasons above, it is necessary for us to find some algorithms to deal with the garbage when heap memory is running out.

This paper is generally divided into three part. We would first walk through the principle, advantages and disadvantages of the Mark and Sweep Algorithm, then the Reference Counting Algorithm in the same pattern and finally a brief conclusion.

2 Mark and Sweep Algorithm

Mark and Sweep Algorithm is an algorithm raised by J.McCarthy in his dissertation in 1960.[4] He ushered in a new domain in computer science, and first apply this algorithm to the programming language LISP, which was also his brain child.

In this section, I will first introduce the principle of Mark and Sweep Algorithm, and then briefly analyze the advantages and disadvantages of it.

2.1 The Principle of Mark and Sweep Algorithm

The algorithm includes two phases, mark phase and sweep phase.

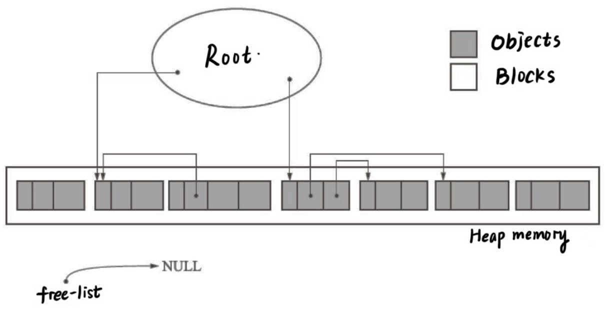

Figure 1. Before the garbage collection process

We can assume a particular situation in heap memory which is shown in Figure 1. In the mark phase, the garbage collector will traverse from the root node, marking those reachable objects (the objects that can be reached following the arrow tips in Figure 1)or those objects users may refer to, usually by depth-first search approach. What should be noticed is that each time we only start from one root node, so if there's more than one root, we should repeat this phase for several times.

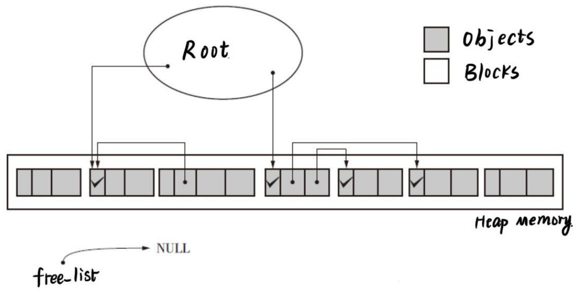

Figure 2. After Mark Phase

As can be seen in Figure 2, after the mark phase, the headers of objects have been ticked or not, representing the current state of objects. Then the sweep phase will start.

In the sweep phase,the collector will traverse the whole heap memory again, linearly, finding out those objects that are not marked and "clear" them out. This "clear" process means that the header of the object will be included to a free_list which is used to store the unallocated memory space, and you can comprehend it visually in Figure 3. After this phase ends, all the mark bit of the objects in heap memory will be reset, preparing for the next time of running this garbage collection algorithm.[1]

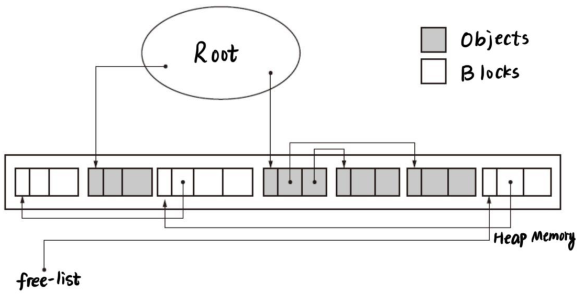

Figure 3. After Sweep Phase

2.2 The Advantages and Disadvantages of Mark and Sweep Algorithm

Obviously, this algorithm is easy to actualize for programmers, but the workload of computer soars up, leading to low efficiency and some potential problems.

First is the problem termed "Fragmentation", meaning that the free memory, especially that that has been "cleared" in the former garbage collecting process, scatter separately in the heap memory. Therefore, if the user wants to create a relatively large object, we probably would not find enough available and contiguous units for him/her, resulting in Out-OfMemory Error.

Besides, the whole program must suspend for garbage collection algorithm, adding with its low efficiency, users might not feel pleasant when this algorithm is applied to a programming language even it save part of their work to some extent.

3 Reference Counting Algorithm

This is an algorithm raised by George E. Collins in his dissertation in 1960. [3] And Harold McBeth pointed out that this algorithm couldn't manage the problem of circular reference in 1963 in "A Letter to The Editor". This is the algorithm which is still widely applied to programming languages nowadays.

In this section, the principle of Reference Counting Algorithm will be first introduced, and following is a brief analysis of the advantages and disadvantages of the algorithm.

3.1 The Principle of Reference Counting Algorithm

In this algorithm, the object will be equipped with a counter for the reference of the object. When the object is referred to by other nodes, the counter would add one, while if that reference is canceled, the counter would minor one, and it will be collected when the counter equals to zero. This 'collected' process is the same as that in the former algorithm, which means that the header of this object will be included into the free _ list storing unallocated objects. What is special is that we would traverse from the object that would be collected soon, to all its child nodes and make those objects' counters each minors one.

The garbage collecting process only works when a new objected is created, or the pointer is renewed, because only when these times the reference of objects will change. This is an immediate process, which makes the time for suspending the whole program much shorter than the former algorithm. Moreover, it can be sure to say that the heap memory is exactly running out if the allocating process failed in this algorithm.[2]

3.2 The Advantages and Disadvantages of Reference Counting Algorithm

This is evidently an algorithm with simple principle and easy to actualize. However, it also has many disadvantages.

The most obvious problem of this algorithm is circular reference, meaning that if A's pointer points to B and B's pointer points to A, while both two of them are no longer useful in the program, they would not be collected because both of their collector doesn't equals to zero.

Besides,the added counter of each object will occupy much spaces, leading to the low efficiency of memory usage. Aside from that, the workload of processing all the collectors is also heavy for the computer because the program would probably renew the pointer, especially the pointers of root nodes.

4 Conclusion

From the two algorithms above, we can figure out clearly that there are two crucial problems to think through when creating an garbage collection algorithm. The first problem is how to find the garbage, by counting the number of times the object referred to, or by judging whether this object can be reached from the root node. Another problem is how to prepare for the reallocating problem of the collected garbage.

The two algorithms mentioned above can solve the problem of garbage collecting to some extent, but there's still much problems existed in them, which needs much more improvements. In the future, garbage collection algorithms are expeceted to be more efficient, occupy less space and decrease the time of suspending the program.

References

[1] Mark-and-Sweep: Garbage Collection Algorithm.
[2] cheniie . Reference Counting Algorithm.
[3] George E.Collins. A Method for Overlapping and Erasure of List. 1960.
[4] John McCarthy. Recursive Functions of Symbolic Expressions and Their Computation by Machine,Part I. *CACM*, April 1960.
[5] Ritchie _Zeng. Recursion and Memory Management.