COMPILER DESIGN

→GARBAGE COLLECTION

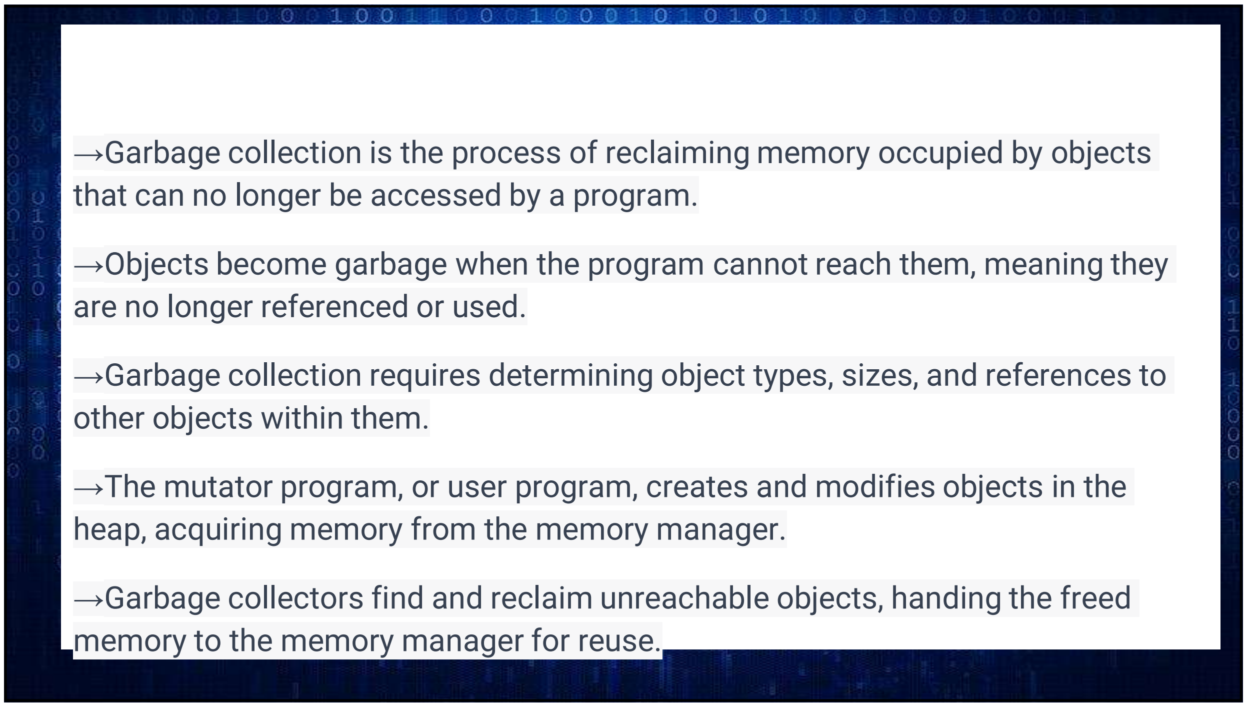
1. Garbage collection is the automatic process of deallocating memory for data that is no longer needed or accessible by a program.

1. It removes the burden of manual memory management from programmers in high-level programming languages.

1. Garbage collection has been around since the early implementation of Lisp in 1958 and is now a standard feature in languages like Java, Perl, ML, Modula-3, Prolog, and Smalltalk.

1. The concept of "reachability" is crucial in garbage collection, referring to whether an object can still be accessed or used by the program.

1. There are different methods of garbage collection, including reference counting, which relies on the idea that objects without references can be deallocated, and trace-based collectors that trace and mark live objects for memory reclamation.



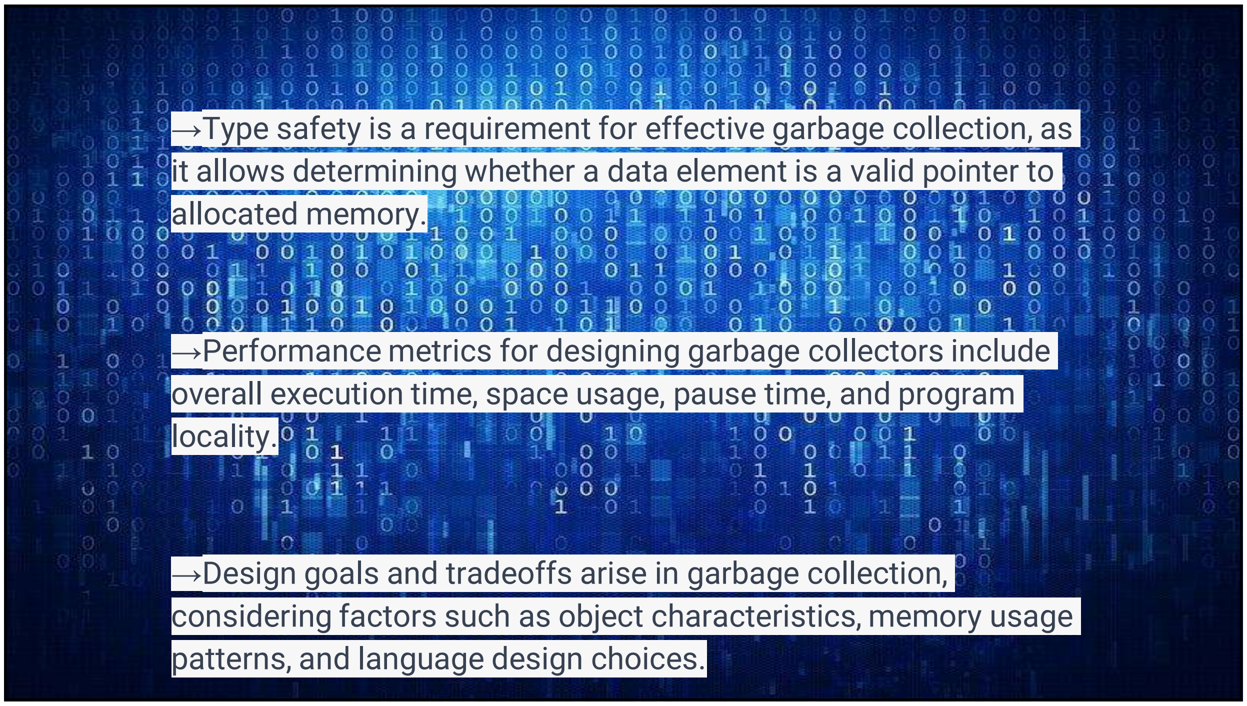
→Garbage collection is the process of reclaiming memory occupied by objects that can no longer be accessed by a program.

→Objects become garbage when the program cannot reach them, meaning they are no longer referenced or used.

→Garbage collection requires determining object types, sizes, and references to other objects within them.

→The mutator program, or user program, creates and modifies objects in the heap, acquiring memory from the memory manager.

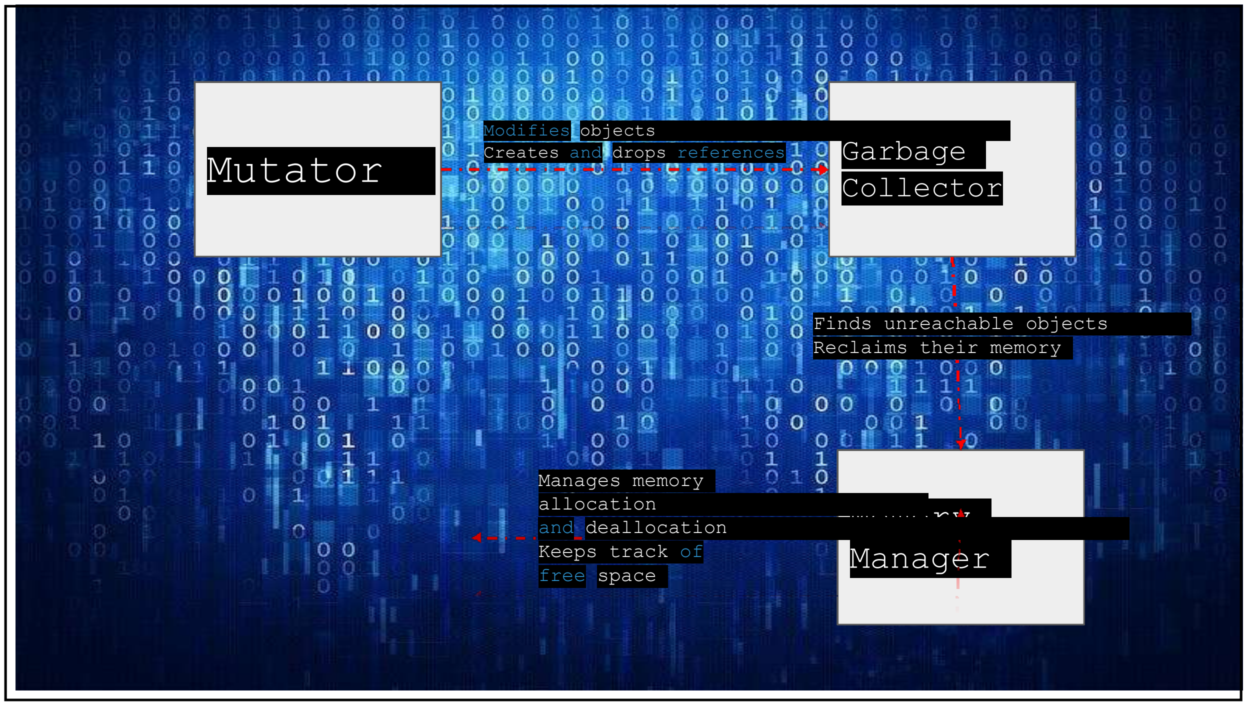
→Garbage collectors find and reclaim unreachable objects, handing the freed memory to the memory manager for reuse.

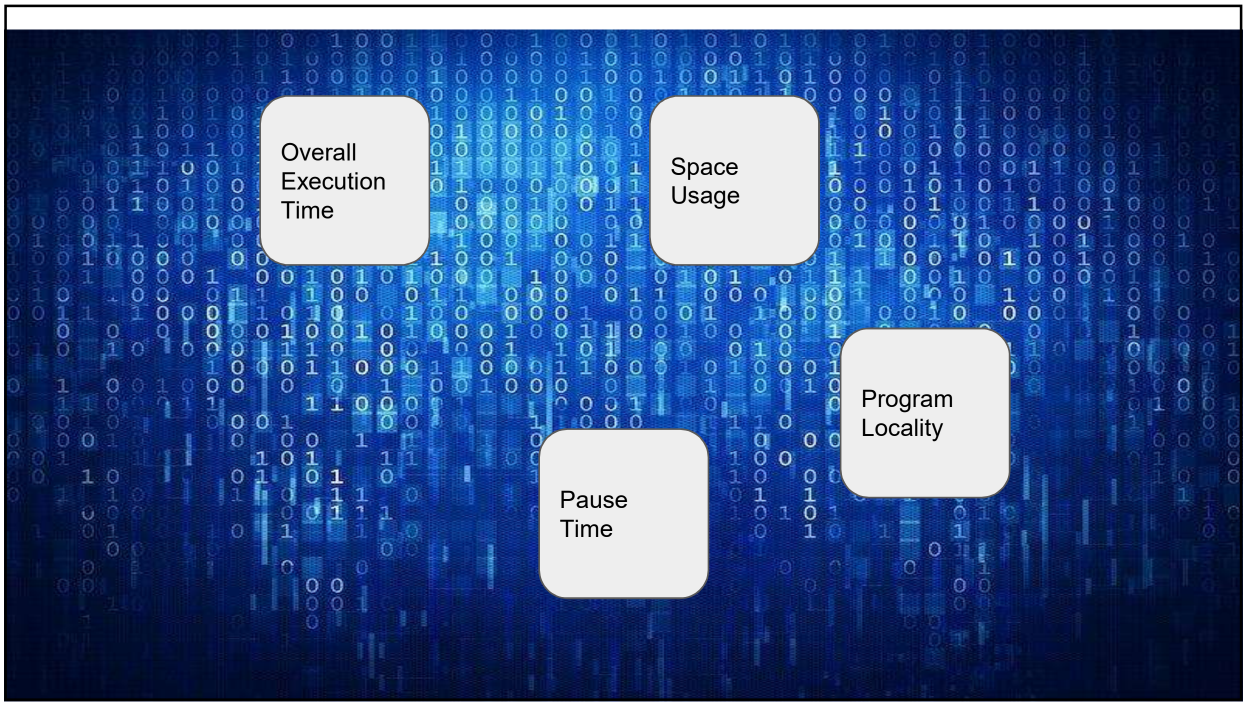


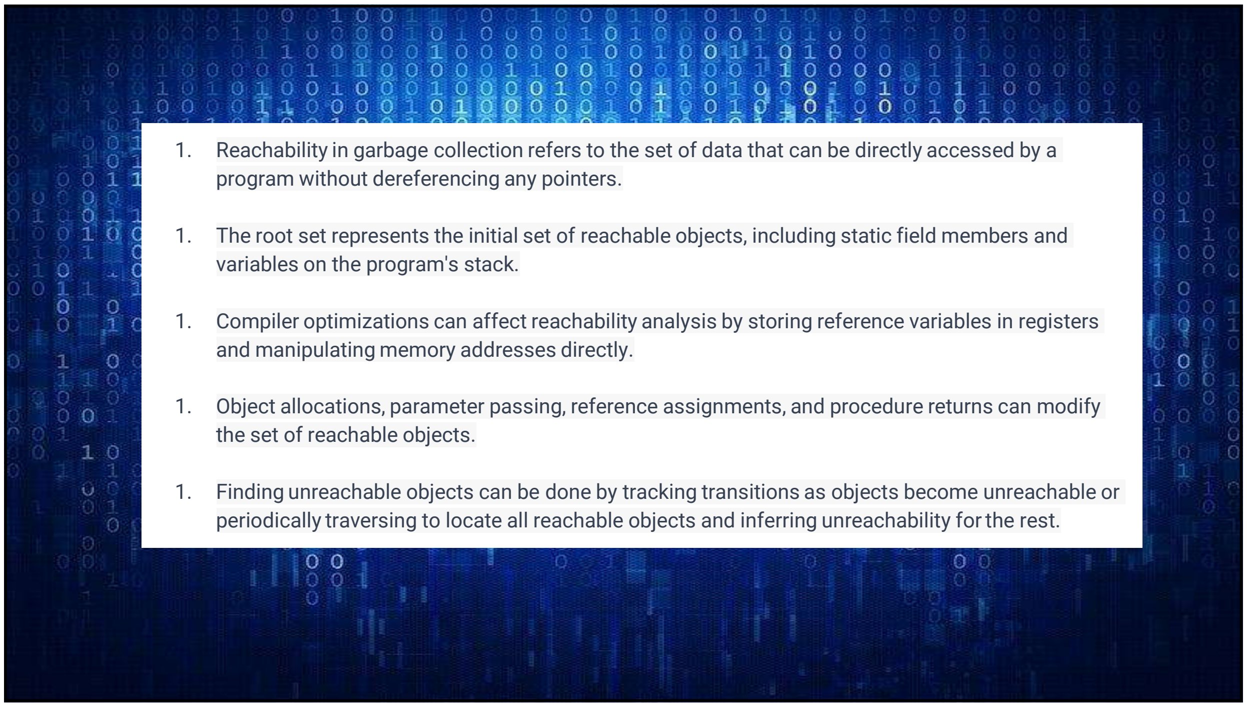
→Type safety is a requirement for effective garbage collection, as it allows determining whether a data element is a valid pointer to allocated memory.

→Performance metrics for designing garbage collectors include overall execution time, space usage, pause time, and program locality.

→Design goals and tradeoffs arise in garbage collection, considering factors such as object characteristics, memory usage patterns, and language design choices.


Mutator
Modifies objects
Creates and drops references
Garbage Collector
Finds unreachable objects
Reclaims their memory
Manages memory allocation and deallocation
Keeps track of free space
[illegible] Manager


Overall Execution Time
Space Usage
Program Locality
Pause Time



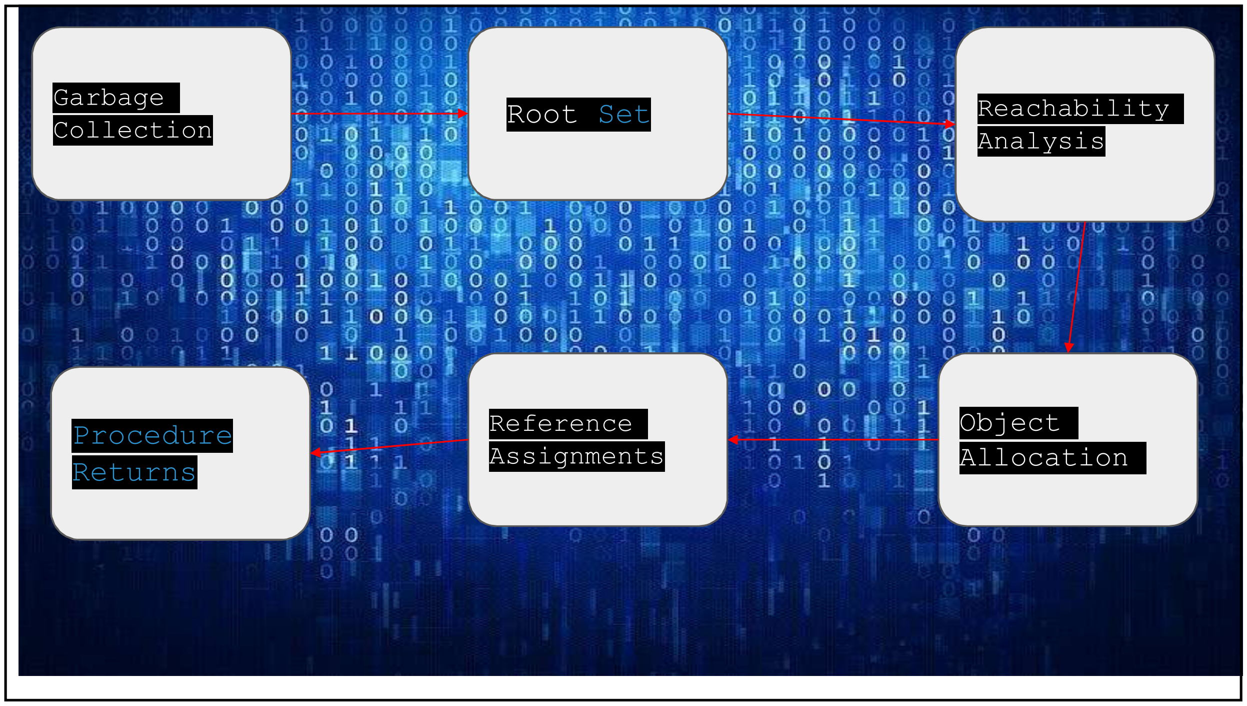
1. Reachability in garbage collection refers to the set of data that can be directly accessed by a program without dereferencing any pointers.

1. The root set represents the initial set of reachable objects, including static field members and variables on the program's stack.

1. Compiler optimizations can affect reachability analysis by storing reference variables in registers and manipulating memory addresses directly.

1. Object allocations, parameter passing, reference assignments, and procedure returns can modify the set of reachable objects.

1. Finding unreachable objects can be done by tracking transitions as objects become unreachable or periodically traversing to locate all reachable objects and inferring unreachability for the rest.



Reference Counting Garbage Collectors

1. Object Allocation. The reference count of the new object is set to 1.

2. Parameter Passing. The reference count of each object passed into a procedure is incremented.

3. Reference Assignments. For statement u = v, where u and v are references, the reference count of the object referred to by v goes up by one, and the count for the old object referred to by u goes down by one.

4. Procedure Returns. As a procedure exits, all the references held by the local variables of that procedure activation record must also be decremented. If several local variables hold references to the same object, tha t object's count must be decremented once for each such reference.

5. Transitive Loss of Reachability. Whenever the reference count of an object becomes zero, we must also decrement the count of each object pointed to by a reference within the object

Reference Counting Garbage Collectors

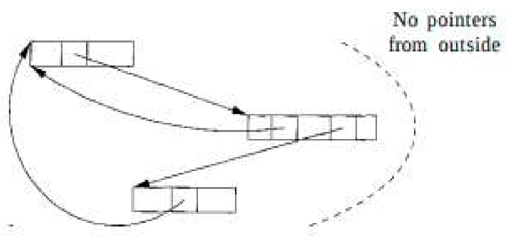


Figure 7.18: An unreachable, cyclic data structure

Reference Counting Garbage Collectors

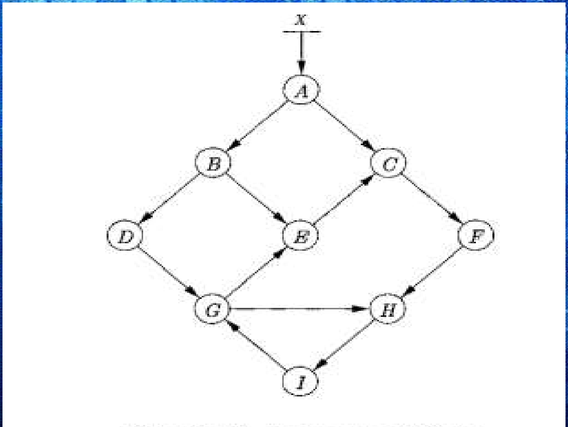


Figure 7.19: A network of objects