Train Algorithm

The generation algorithm
1. works well for immature objects
2. works less well for mature objects — every time the generation they’re in is garbage collected, they get moved!

The train algorithm was designed to handle mature objects well.

Unlike the generational algorithm, the train algorithm never needs to collect the entire heap.

Train Algorithm — Basic Idea

Split memory into small blocks.
Perform a copy-collection-type garbage collection separately for each block.
We get shorter pauses since the size of the blocks is small.
Since we’re only collecting a part of the entire heap at a time, we need to use remembered sets (just like for generational collection) to handle references between blocks.

Combining Algorithms

Combine the train and generational algorithms:
1. Use the generational algorithm for immature objects;
2. When an object has survived a few collections, move it to a different head managed by the train algorithm,
Multi-Generation Cycles

• In the generational algorithm we must occasionally collect the entire heap, or we may miss cyclic garbage.

Train Algorithm

• Fixed sized partitions called **cars** — the size of one (or more) disk blocks.
• Cars are organized into **trains**.
• No limit on the number of trains.
• No limit on the number of cars per train.

Train Organization

• Trains are numbered 1, 2, 3, . . .
• Cars are numbered \( \langle \text{train} – \text{number} \rangle \cdot \langle \text{car} – \text{number} \rangle \).
• Trains and cars are ordered lexicographically, i.e. 1.1, 1.2, 1.3, . . . , 2.1, 2.2, 2.3, . . . , 3.1, 3.2, 3.3, . . .

Train Algorithm — Case 1

• Car 1.1 is collected. Unreachable objects, including cycles contained within the car, are identified.
• Reachable objects are moved to some other car.
• The car becomes empty and is removed from the train.
No root pointers point to train 1.
The remembered set for train 1 only has pointers from cars of the same train.
All cycles are contained in train 1 ⇒ Delete train 1!

- Each car's remembered set contains pointers to objects in higher numbered cars in the same train and higher-numbered trains.
- Each train's remembered sets contain pointers to objects in higher-numbered trains.

The goal is to move out of train one everything that's not cyclic garbage.
When a train is just cyclic garbage, we throw it away.
Create a new train every \( k \) object creations.
On \( o \leftarrow \text{new } T \) we could add \( o \) to
1. the last car of the last train, if there's room, or
2. a new last car of last train, or
3. the first car of a new last train.
Consider Car 1.1's remembered set and the roots.

Scan objects within the car.

Move reachable object \( o \) to another car \( c \):

- If the remembered set says \( o \) is referenced from some other (higher-numbered) train, move \( o \) to some car \( c \) in that train. Pick a car that references \( o \) (good for locality). If there's no room, add a new car.
- If no other train references \( o \), move \( c \) to another car within the same train. Prefer a car that references \( o \) — this will move cyclic structures to the same car. If there’s no room, move \( c \) to a new last car.

Remove car 1.1.

Eventually, all the cars in Train 1 will have been removed ⇒ remove Train 1.

Therefore, eventually, every train becomes the first train, and its cars get garbage collected.
Example 1 — Step 2
Remembered Set:
C
B
D
A
C
D
4.1
2 2.1
3.13
4

Example 1 — Step 3
Remembered Set:
DC B
D
A
4
3.13
4.1

Example 1 — Step 4
Remembered Set:
CD B A
4 4.1

Example 1 — Step 5
Remembered Set:
Example 2 — Step 1

Example 2 — Step 2

Example 2 — Step 3

Really Large Objects

- Since cars are fixed size (maybe the size of a memory page), there may be really large blocks that don’t fit.
- Use a special heap for such large objects.
Read Aho, Lam, Sethi, Ullman, Section 7.7.5

Incremental Garbage Collection: The Train Algorithm, Thomas Würthinger:
http://www.ssw.uni-linz.ac.at/General/Staff/TW/Wuerthinger05Train.pdf