```
x = new X
y = new Y
x.y = y; // increment reference count of y
y.x = x; // increment reference count of x
```

```
new Foo; // creates an object on the heap

// in a function
Foo f; // creates an object on the stack

// in global scope
Foo f; // creates an object in the global data section of the program
```

Root set
- the objects that could be referenced by the code without following any pointers/by only 1 pointer
e.g.
- global variables
- local variables (e.g. on the stack)

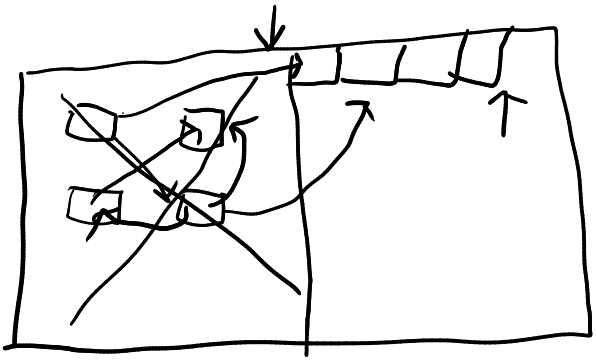
```
// all objects are implicitly pointers
void java_fn(Foo foo) {
	c_function(foo);
}

// C code
Foo* global;
void c_function(Foo* foo) {
	printf("%d\n", global->bar);
	foreign_code_keeping_pointer(foo);
	foreign_code_release_pointer(global);
	global = foo;
}
```

Reachable:
- root set is reachable
- any object we can reference by following pointers from the root set is reachable

```
signal_stop_and_wait();
unscanned = [];
for o in compute_root_set() {
	push(unscanned, o);
}
while len(unscanned) > 0 {
	o = pop(unscanned);
	for o' referenced by o {
		if !is_marked(o') {
			push(unscanned, o');
			mark(o');
		}
	}
}
for o in all_objects() {
	if !is_marked(o) {
		free(o);
	}
}
```

```
bool is_marked(char* bitmap, int index) {
	return (bitmap[i / 8] & (1 << (i % 8))) != 0;
}
```



Copying collector pseudocode: page 481 of textbook