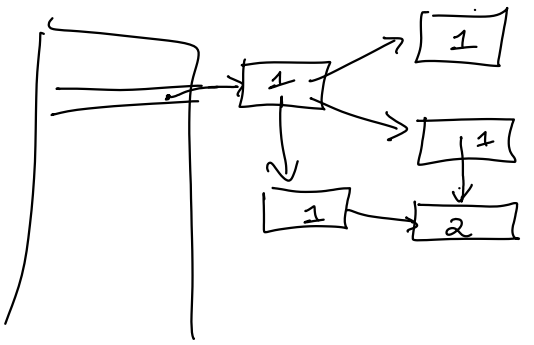
Malloc

- FBR in a linked cyclic list
- on alloc, find FBR and use its memory
- on dealloc, create FBR
- FBR stored in "free" memory

Suppose I use malloc (C++98)

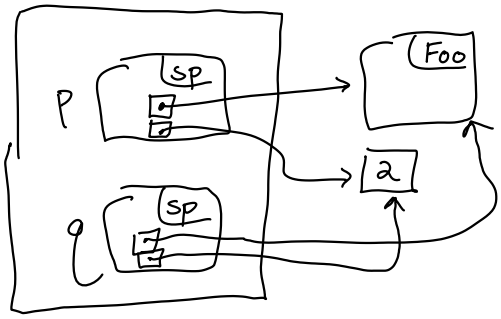
- C++ has "copy constructor" `Foo x = y;` (also a Foo)



In manual memory management in C++, I delete an object when it can no longer be reached.

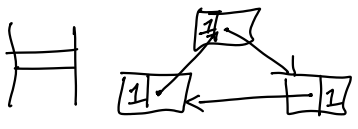
Reference counting: track # of refs to an object; delete object when # reaches 0

C++11: `#include <memory>`




```
shared_ptr<Foo> p(new Foo(1,2,3));
shared_ptr<Foo> q = p;
        ⋮
return 0;
```

Problem Case



Ref Counting

- ⊕ Library implemented
- ⊖ A little overhead
- ⊕ More eagerly frees memory
- ⊖ Can't amortize on effort
- ⊖ Fails on cycles

Mark / Compact

- Recognize pointers
- Move memory
- Stop everything and GC
- Invariant: all heap objects are adjacent
- Know how to read heap objects (size)

Alternative #1

- Subdivide heap
- Run GC on divisions of heap

⊕ Linear walks on heap (forward, compact), time is divided by constant

⊖ Fragmentation across subdivisions (mitigations possible)

> **"Generational Hypothesis"**
>
> Most allocations are short-lived
> ⟹ Generational heaps

Alternative #2 (based on #1)

- Subdivide heap
- On GC on a division, kept objects get "promoted": moved to the next subdivision