Memory Management

Program

Language runtime (Pointers, memory allocation, garbage collection)

Operating system (Virtual memory, page tables, swapping, ...)

Hardware (Memory controller, Direct Memory Access (DMA), Translation Lookaside Buffer (TLB), ...)

```
let f n =
  let l = Base.List.range 0 n in
  fun i -> List.nth l i
```

This call to f allocates a list l

```
─( 16:13:20 )─< command 5 >──────────────────────────────{ counter: 0 }─
utop # let g = f 10;;
val g : int -> int = <fun>
─( 16:13:44 )─< command 6 >──────────────────────────────{ counter: 0 }─
utop # g 3;;
- : int = 3
─( 16:13:57 )─< command 7 >──────────────────────────────{ counter: 0 }─
utop # g 5;;
- : int = 5
─( 16:14:13 )─< command 8 >──────────────────────────────{ counter: 0 }─
utop # g 7;;
- : int = 7
```

All calls Refer to the same l.

```
let g2 = f 100

g2 73
```

Creating g_2 does not invalidate g

There must be two independent l-s in memory.

Jobs of the language runtime

- Memory management
- What things are in memory?
- What memory is used?
- What memory is available?

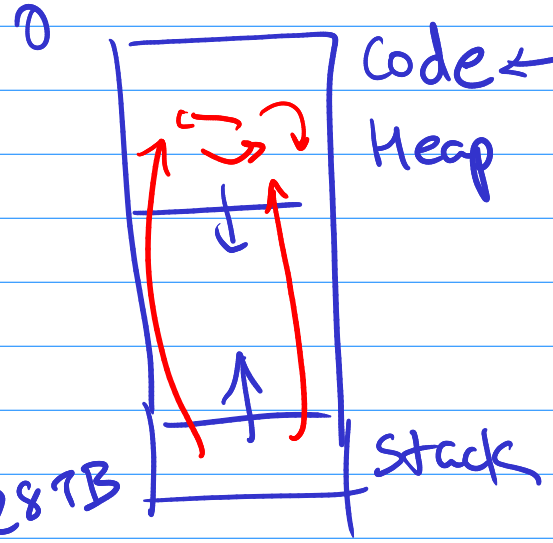
```
let rl = ref (Base.List.range 0 10)

rl := (List.map Int.succ (Base.List.range 0 200))
```

Old list not reachable any more.

Memory can be reclaimed!

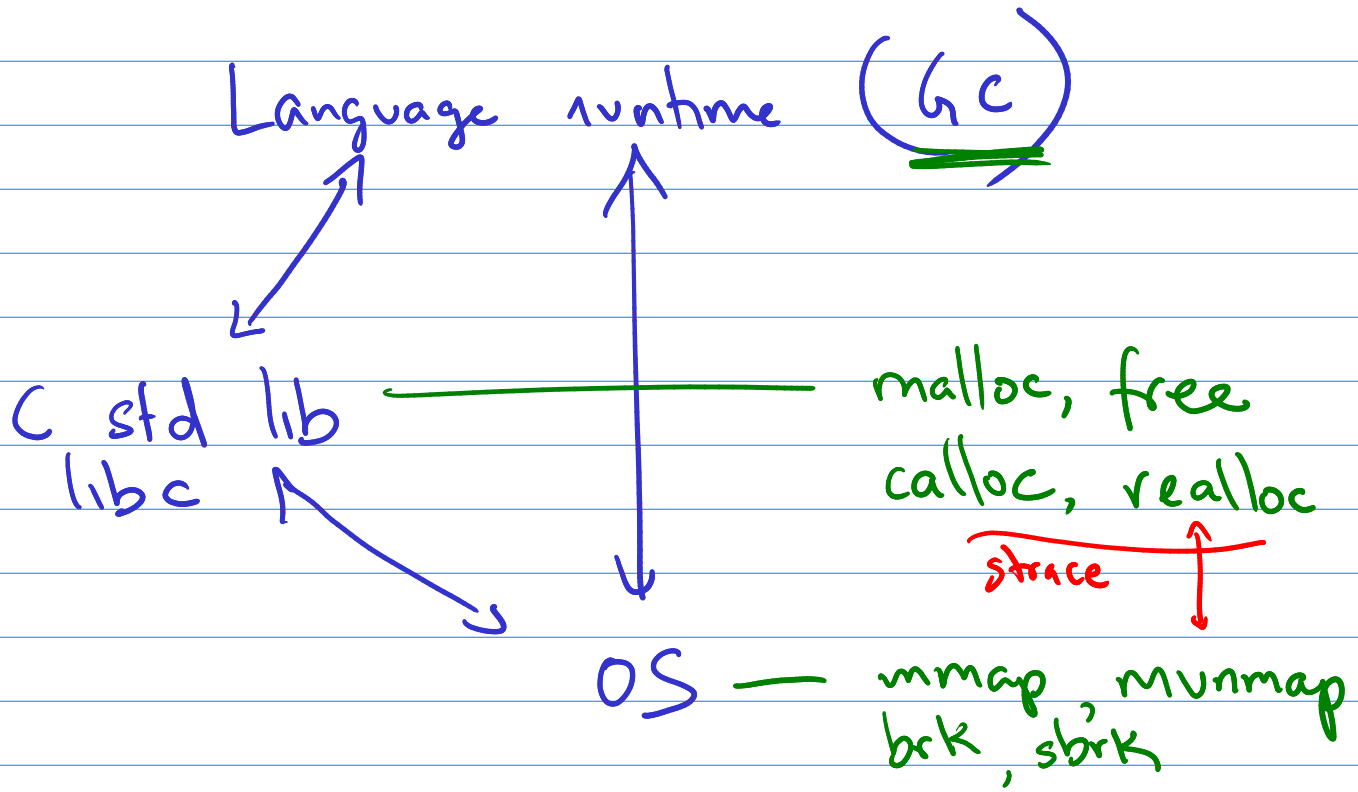
Virtual address space

2^{64}



$2^{47} = 128TB$

All currently accessable values must be reachable from the stack.



Memory management / Garbage collection

1. Where to allocate an object?
2. When is it safe to deallocate an object?
3. When to actually deallocate?

Proposal: An object is safe to deallocate if nothing points to it.

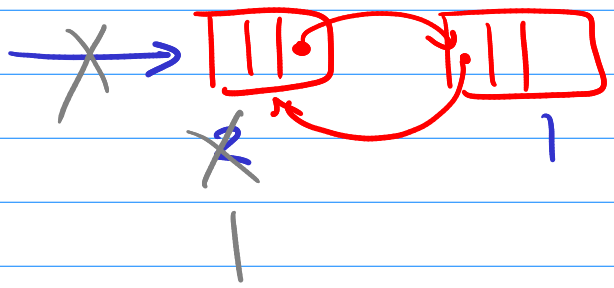
(Ref-counting) → Python

→ C++ auto ptrs

+ simple to implement (careful about concurrency!)

+ fast

− but it can leak memory!



"problem of circular references"

Proposal 2 : Mark-&-sweep garbage collector

"Pauseless GCs"

Step 0 : Pause the program "Stop-the-world GCs"

Phase 1 : Start from the "root set"

- Follow the chain of pointers
- Mark all visited objects as "live"

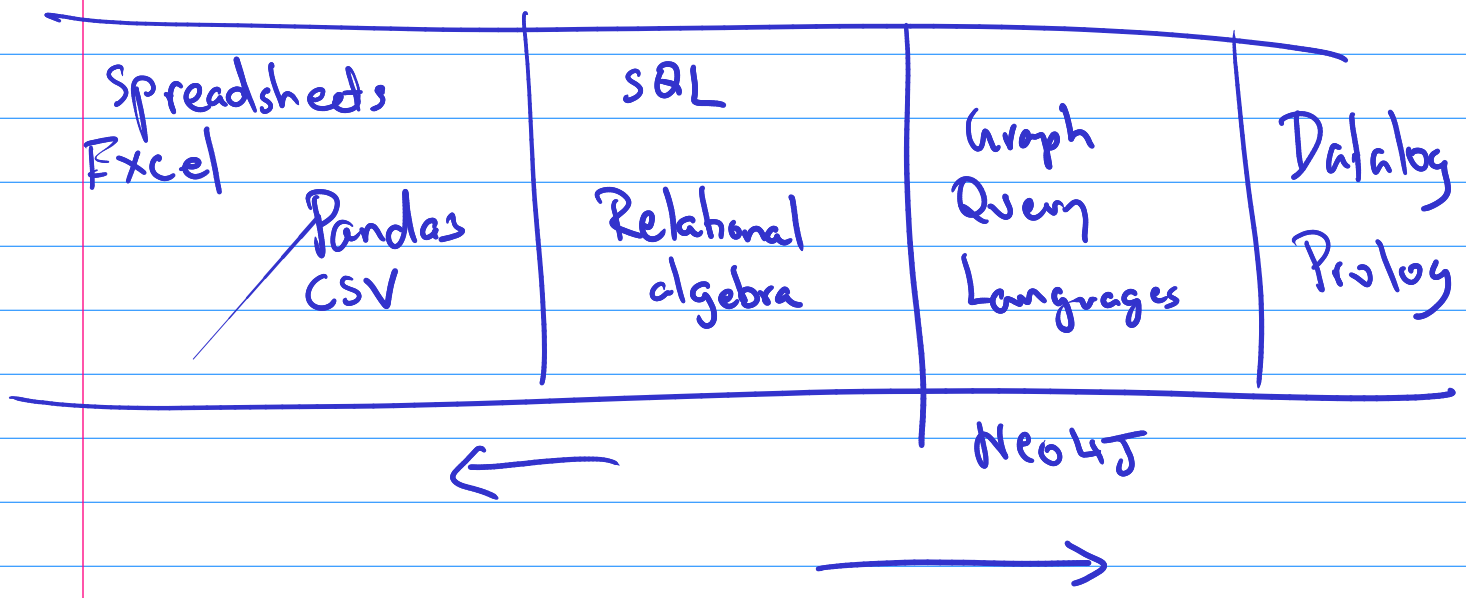
Phase 2 : Go over all currently allocated objects

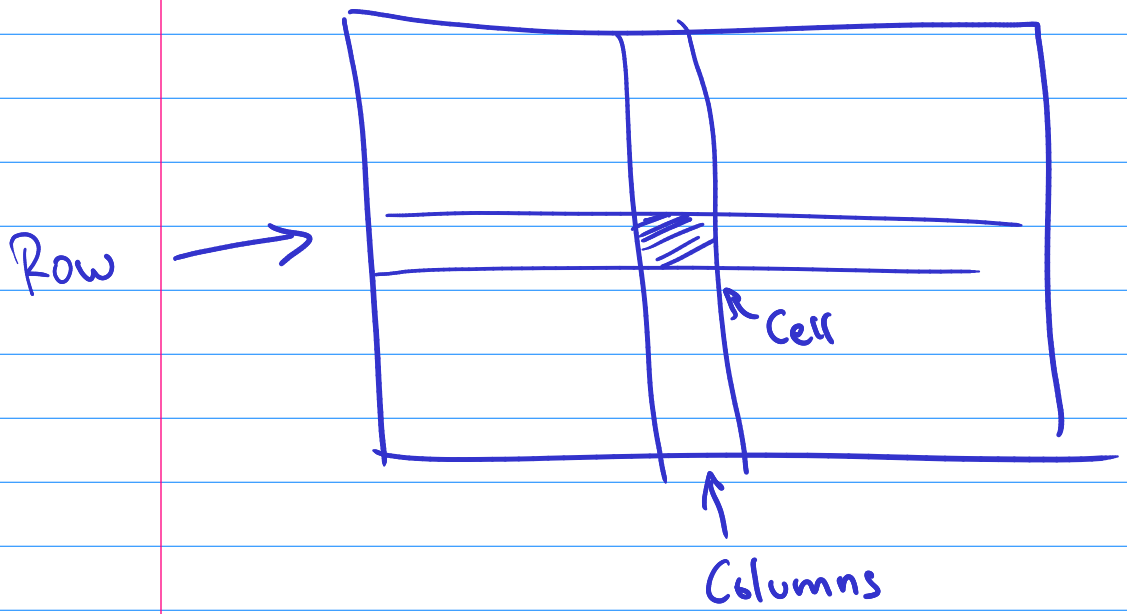
- If not live, then mark them as candidates for deallocation.

"Generational garbage collection"

Newly created objects more likely to become unreachable than old objects.

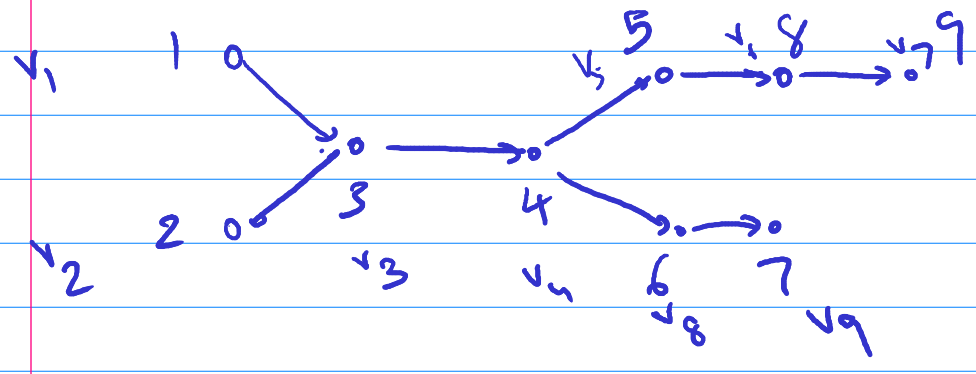
Unit 3 : Programming With Relations





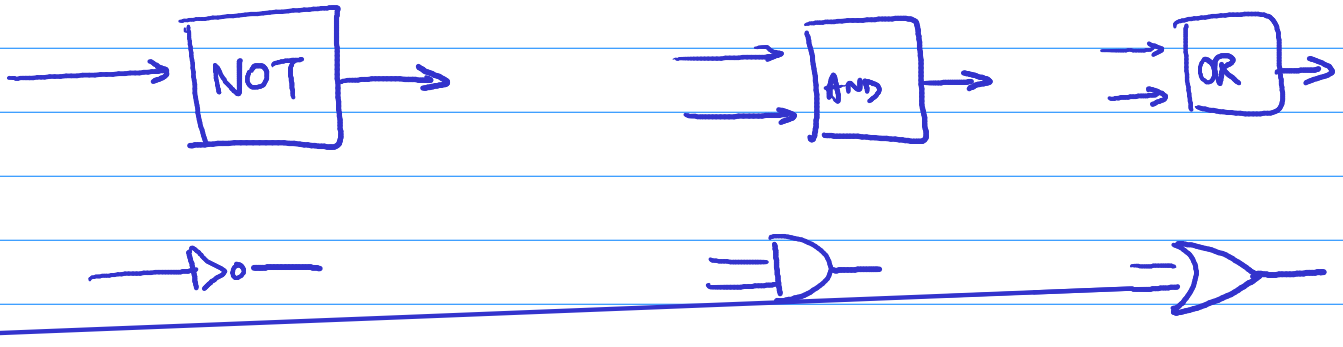
Topological Sort

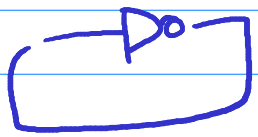
- Check if a graph is a DAG (Directed Acyclic Graph)
- Emit a proof that it is a DAG

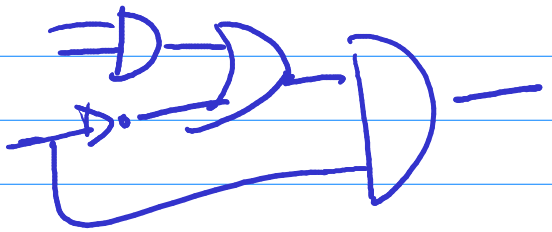


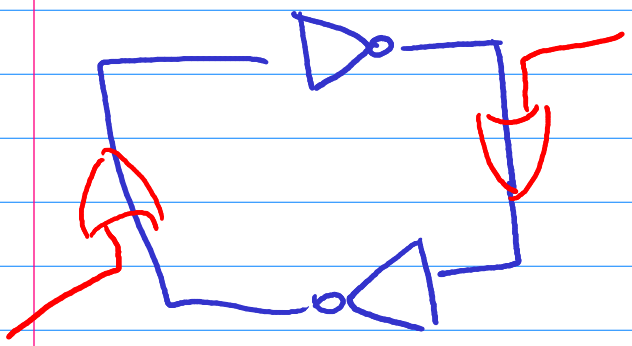
$0 < v_1$
$0 < v_2 \dots$
$v_1 < v_3$
$v_2 < v_3$
$v_3 < v_4$
$v_4 < v_5$
$v_5 < v_6$
$\vdots$

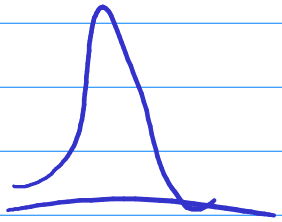
$\min(v_1 + v_2 + \dots + v_9)$


NOT
AND
OR









DAGs ⟶ Spreadsheets /
Makefiles / Build systems
Combinatorial circuits

Q1. Is there / what is the equivalent of pointers?

Q2: What is the counterpart of a dictionary?

Q3: Can we do simple "real" computations in Excel.