Fine-grain Memory Object Representation in Symbolic Execution

Martin Nowack (m.nowack@imperial.ac.uk)

Imperial College London

SOFTWARE RELIABILITY GROUP

2nd International

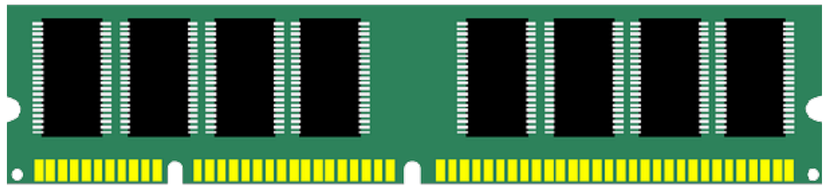
Workshop

Programs

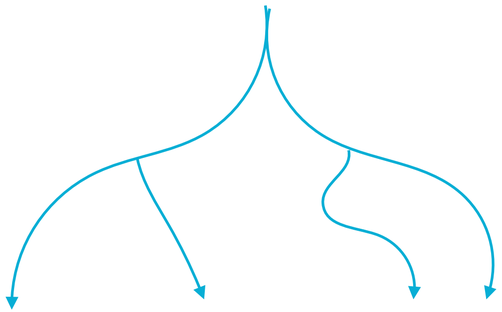
```
char * a = malloc(1024);
int32 i = 10;

a[i]++;
if (i != 12345)
{
  a[i-2] = a[i] * 2;
} else {
  a[i+2] = a[i] - 2;
}
```

Memory Representation

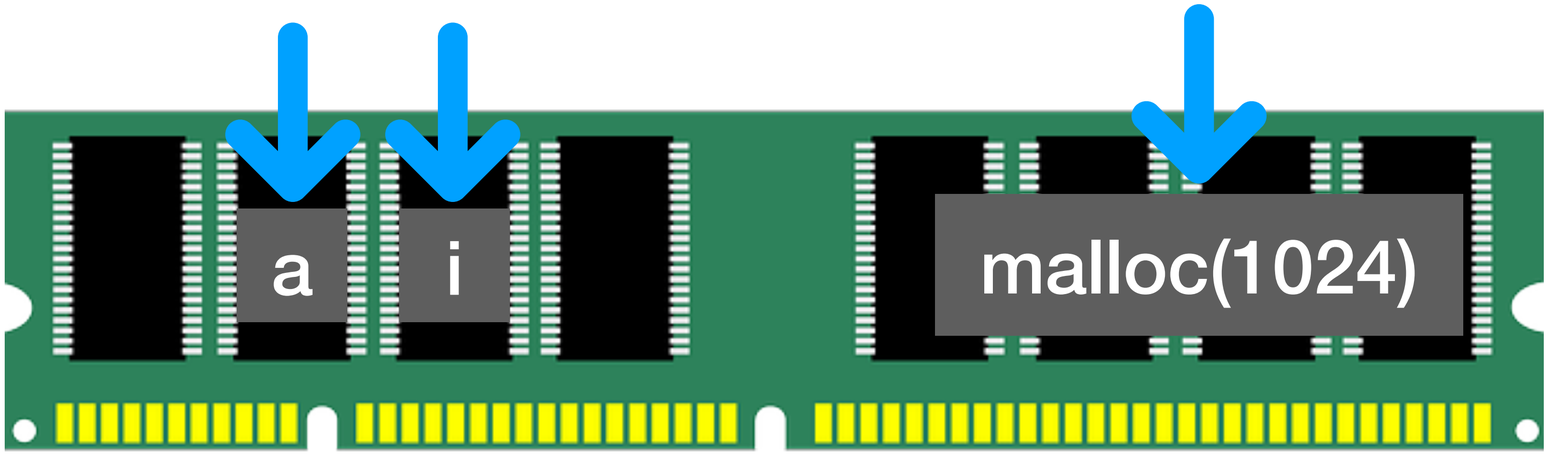


Symbolic Execution



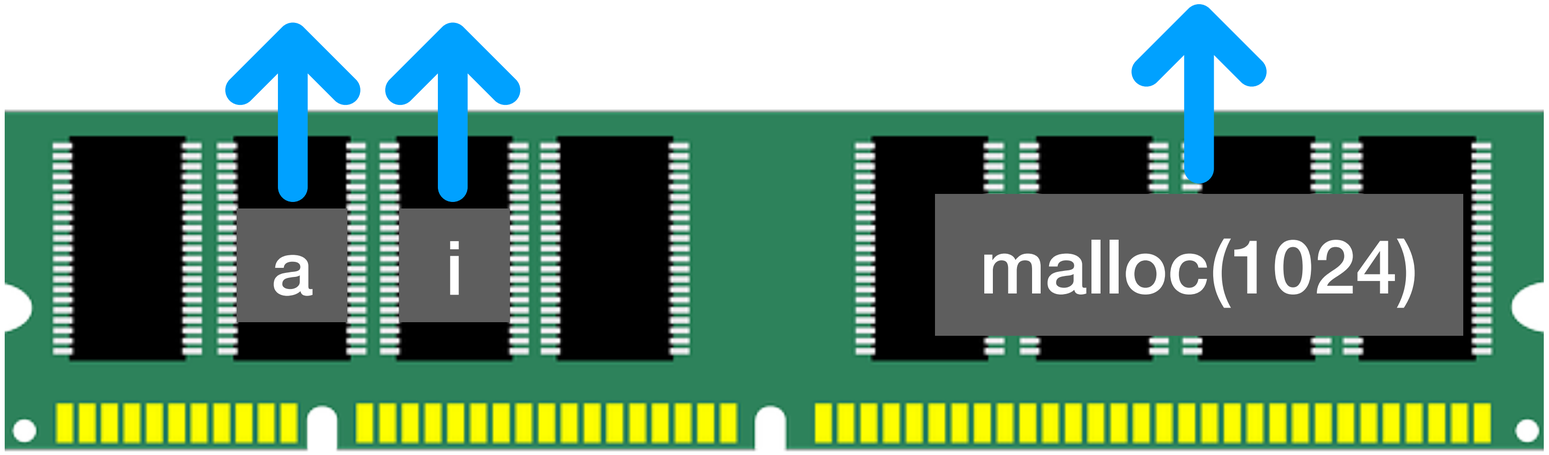
```
char * a = malloc(1024);
int32 i = 10;

a[i]++;
if (i != 12345) {
   a[i-2] = a[i] * 2;
} else {
   a[i+2] = a[i] - 2;
}
```

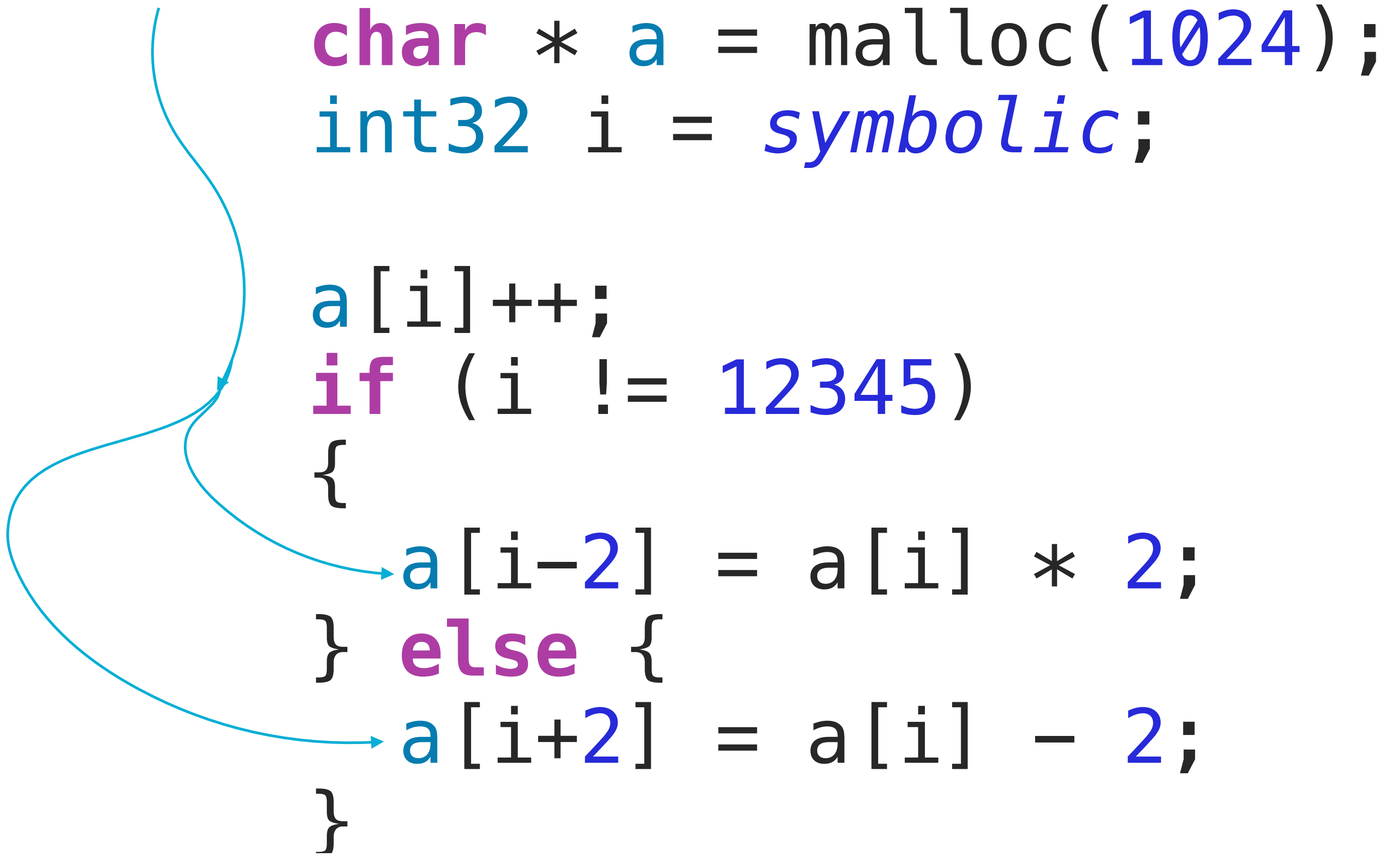


```
char * a = malloc(1024);
int32 i = 10;

a[i]++;
if (i != 12345) {
   a[i-2] = a[i] * 2;
} else {
   a[i+2] = a[i] - 2;
}
```



(Dynamic) Symbolic Execution

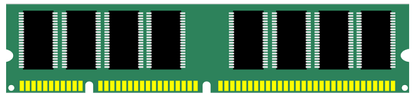


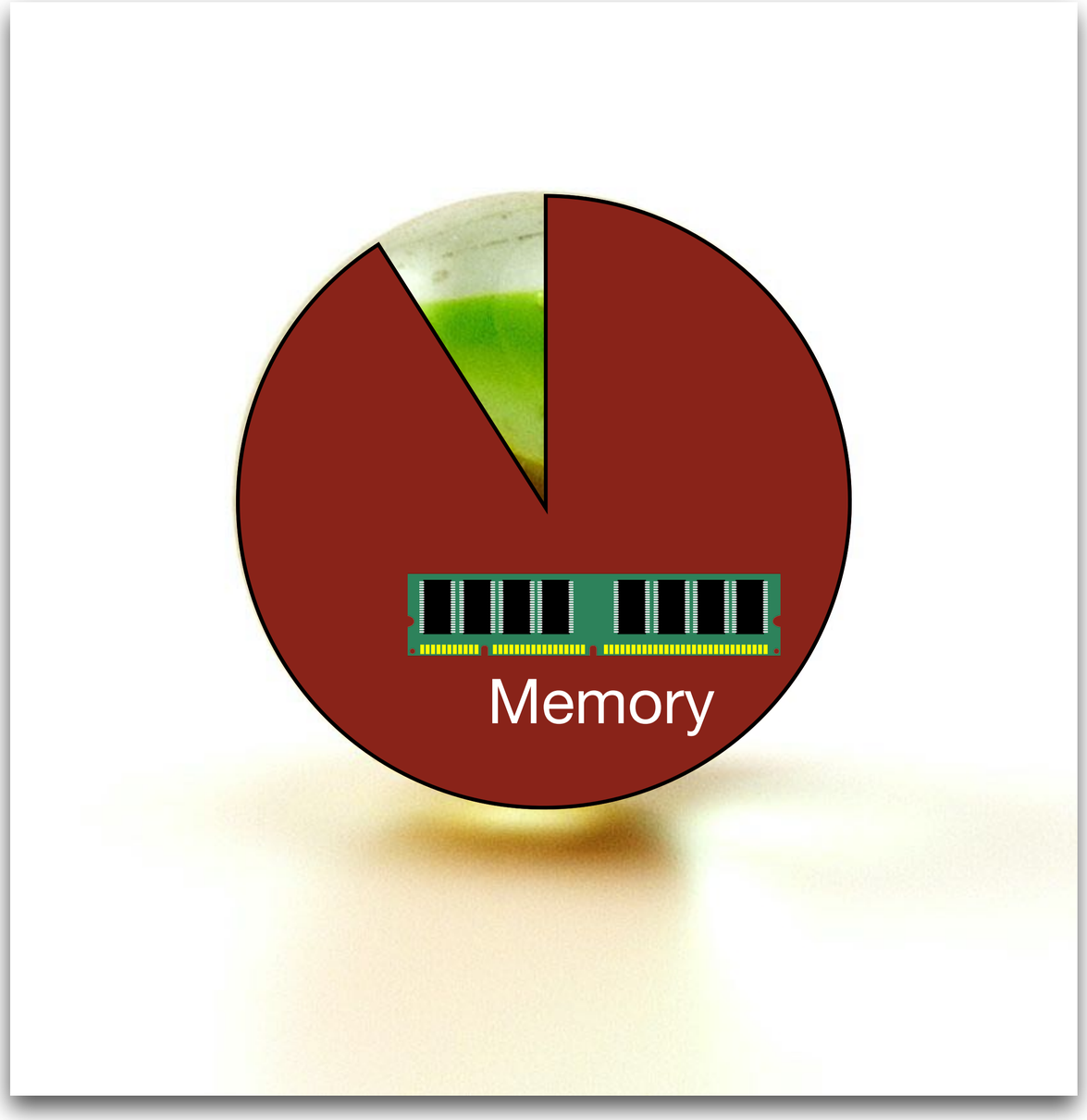
```
char * a = malloc(1024);
int32 i = symbolic;

a[i]++;
if (i != 12345)
{
    a[i-2] = a[i] * 2;
} else {
    a[i+2] = a[i] - 2;
}
```

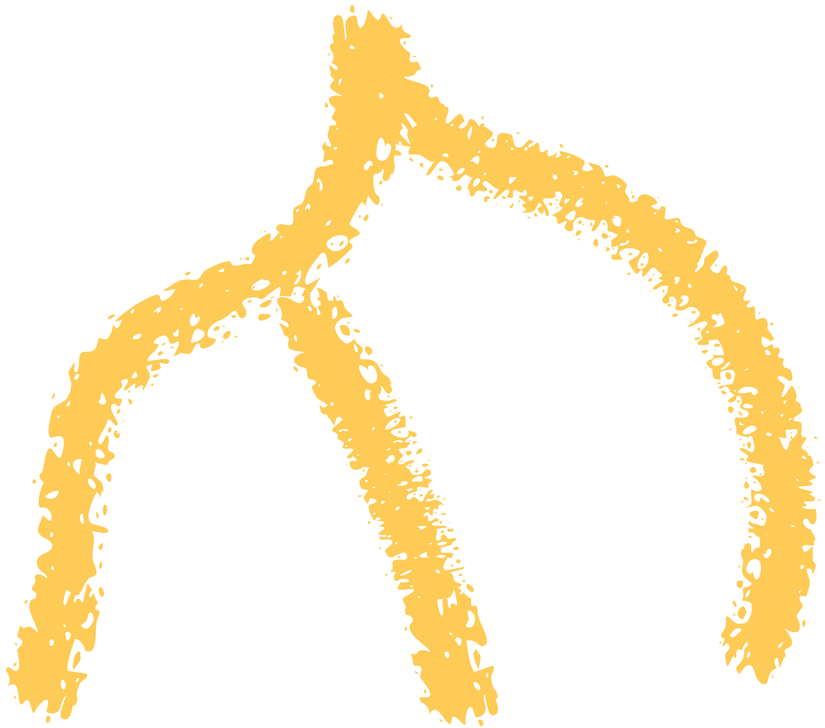
Application State

A Simplified View

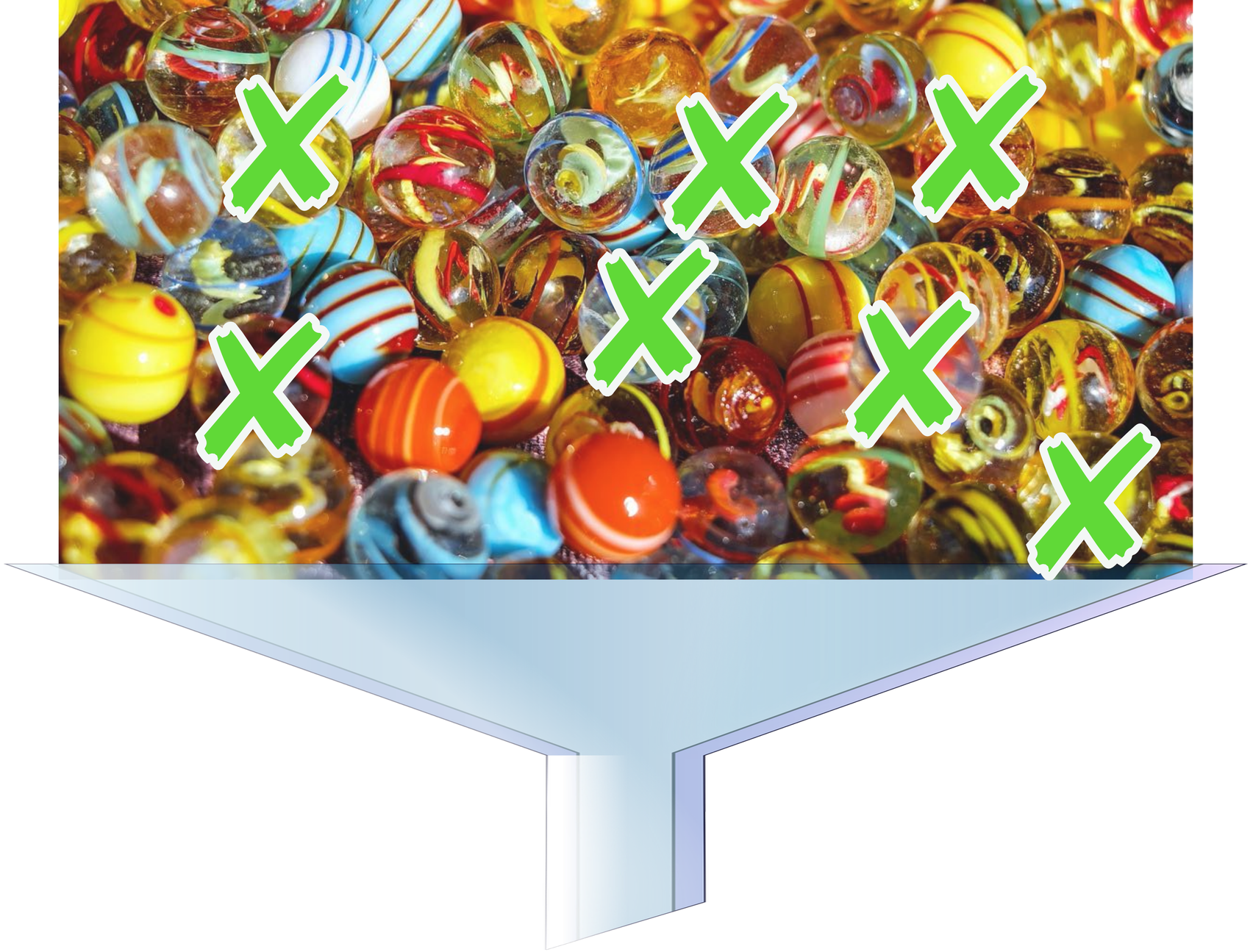
- Path Constraints
- Registers (i.e., program counter)
- Allocated Memory 
  - Stack
  - Heap



The many states …



The many states …



Goal

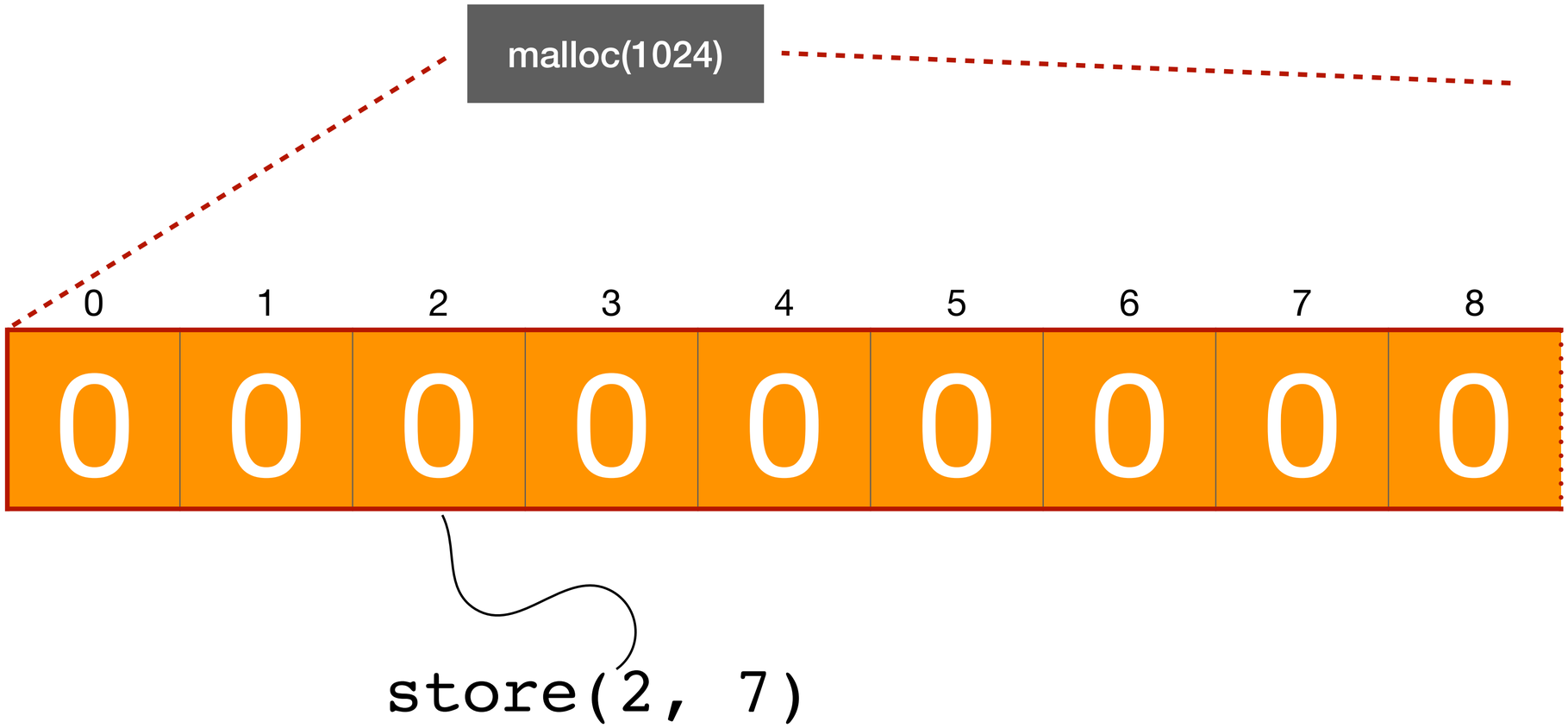
- Scale symbolic execution
  - Avoid premature termination of states
  - Sort/Reason about states

Handling Allocations

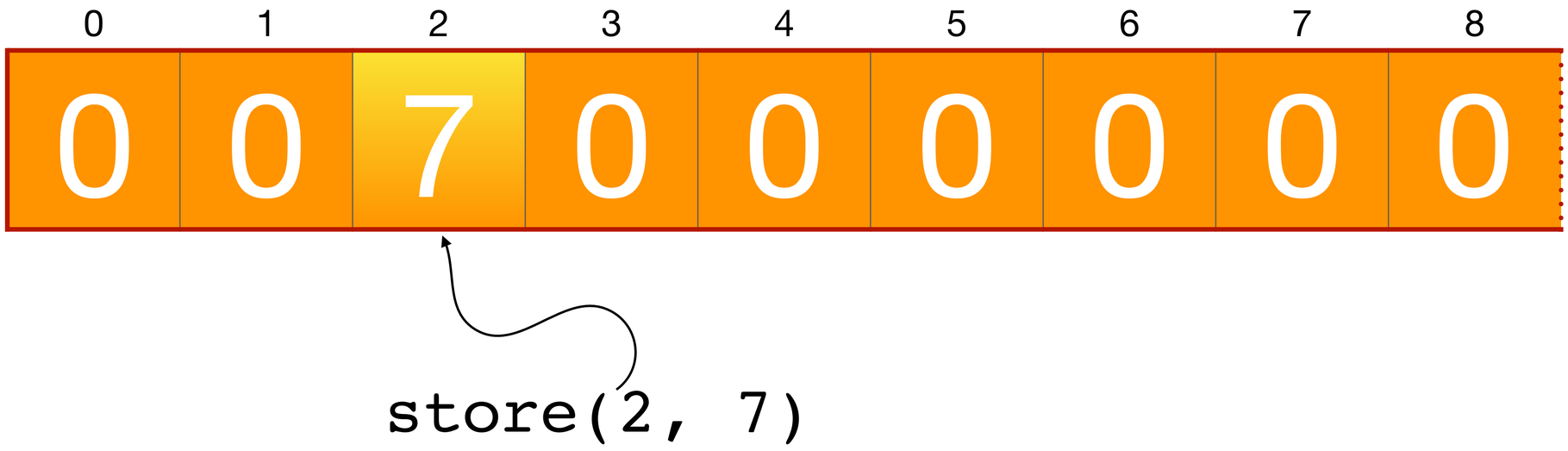
State of the art

Copy on Write (Cow)

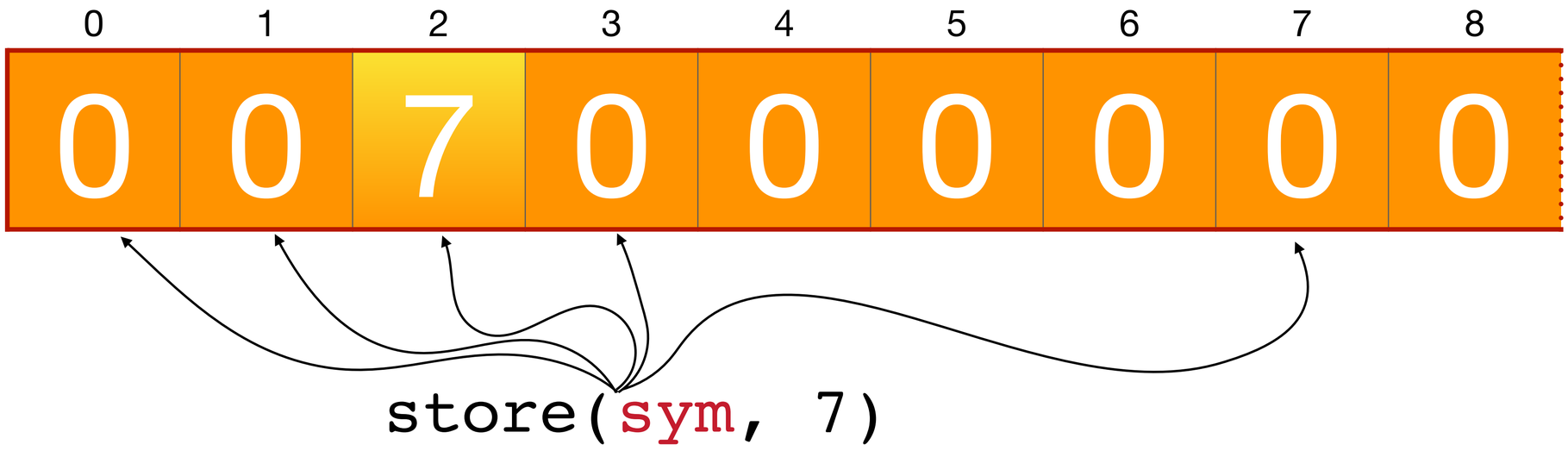

```
char * a = malloc(1024);
...
if (i != 12345)
{
   ...
} else {
   ...
}
```


malloc(1024)
0
1
2
3
4
5
6
7
8
0 0 0 0 0 0 0 0 0
store(2, 7)

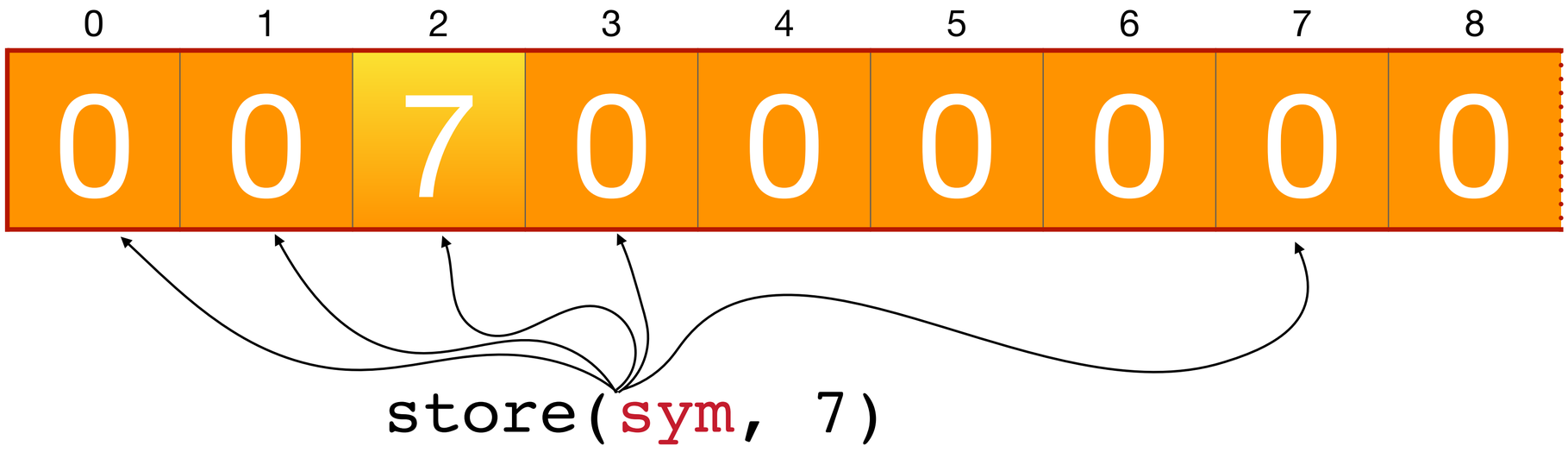
Handling Symbolics



Handling Symbolics



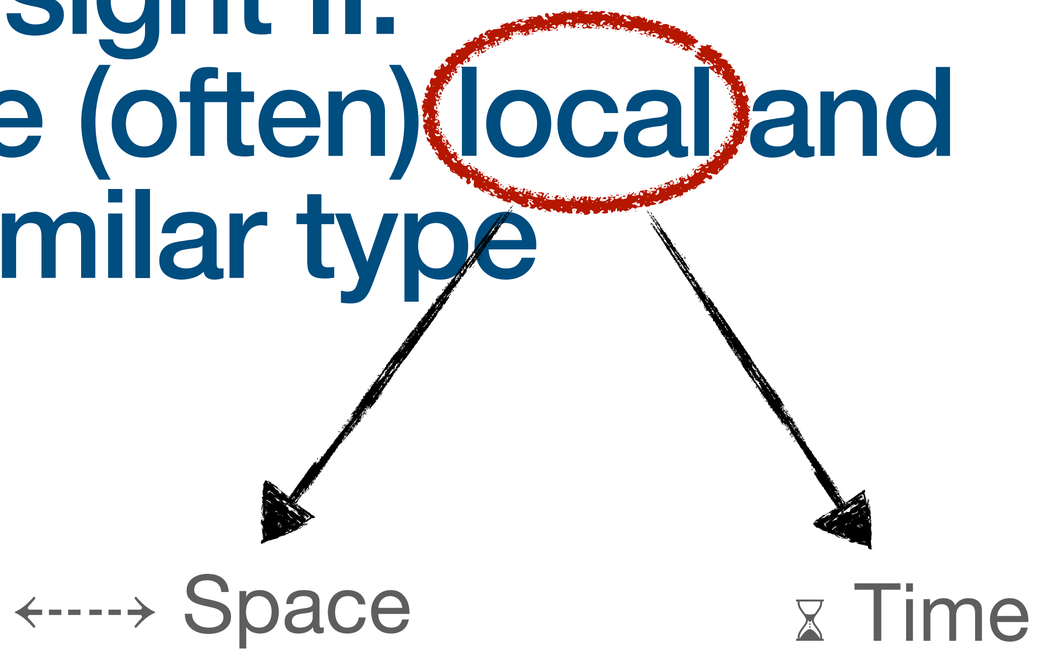
Handling Symbolics



Fine-grained Memory-Object Representation

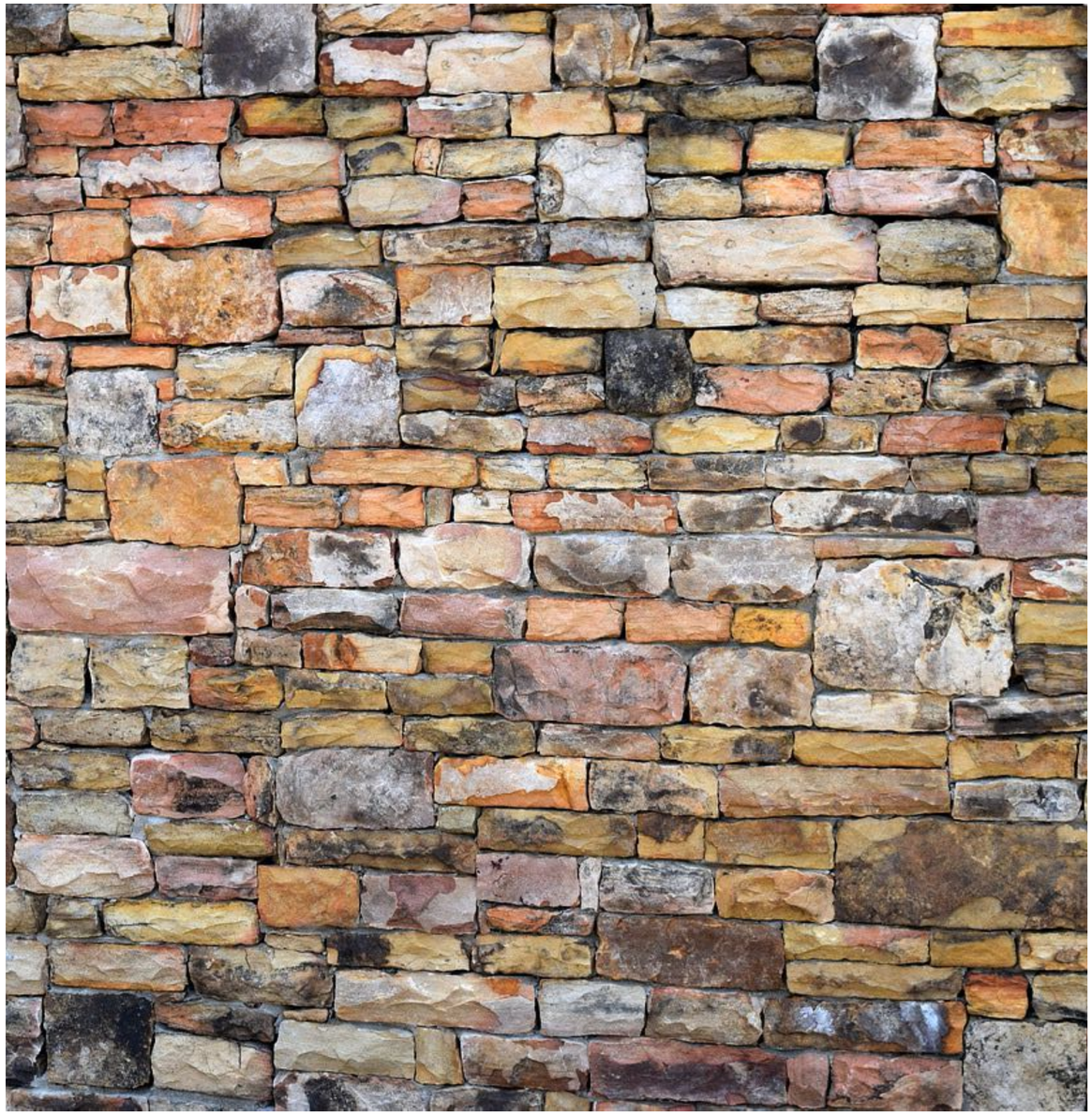
Insight I:
Differences are small and common parts large

Insight II:
Changes are (often) local and of similar type



Basic Building Blocks

Everything is a Layer

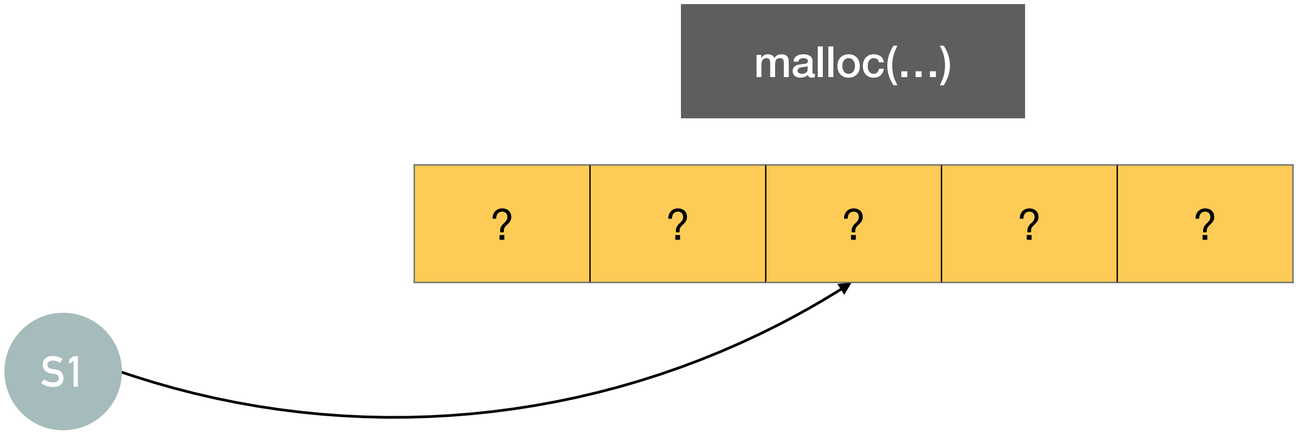


Example Scenario

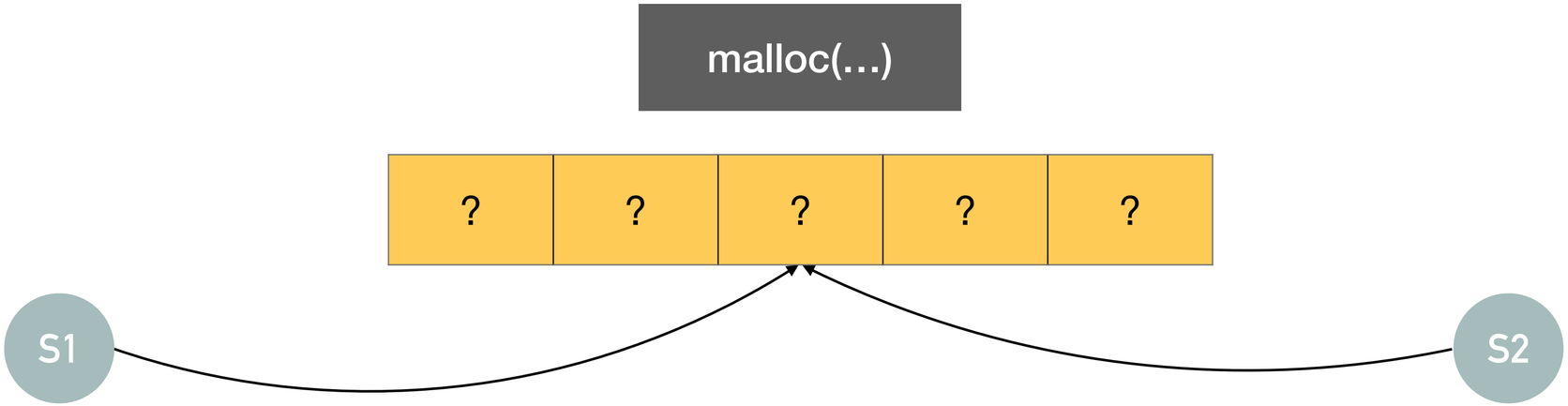
malloc(…)

S1

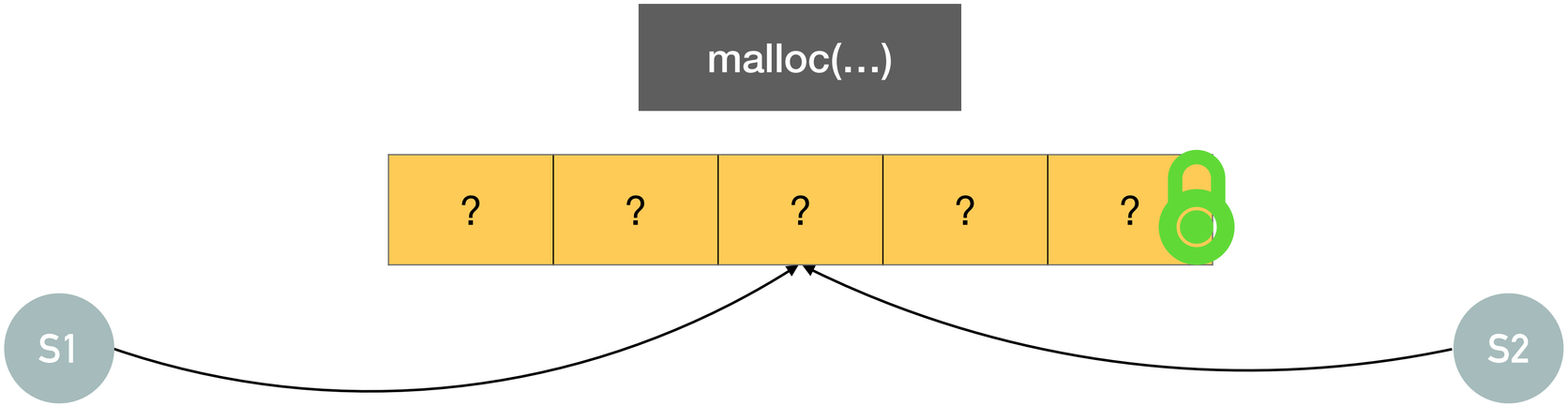
Example Scenario



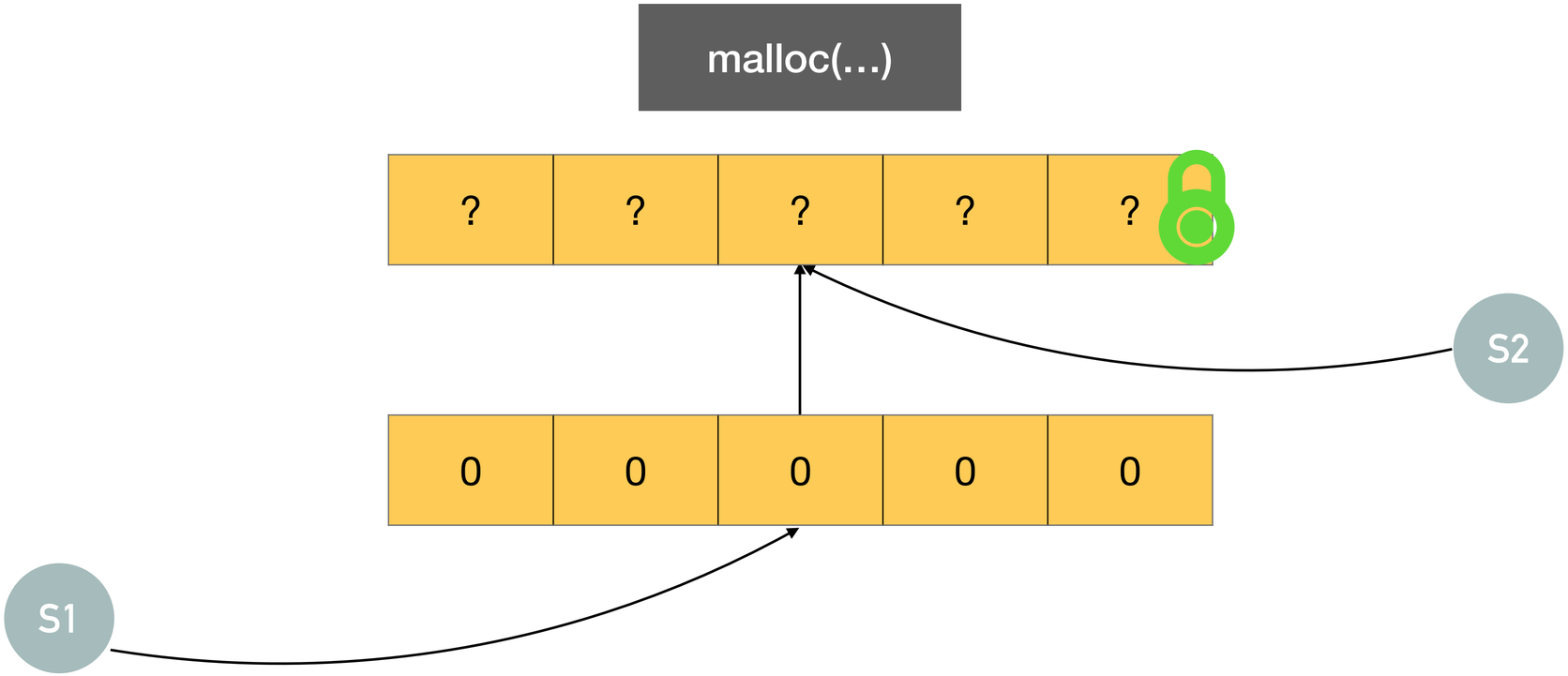
Example Scenario



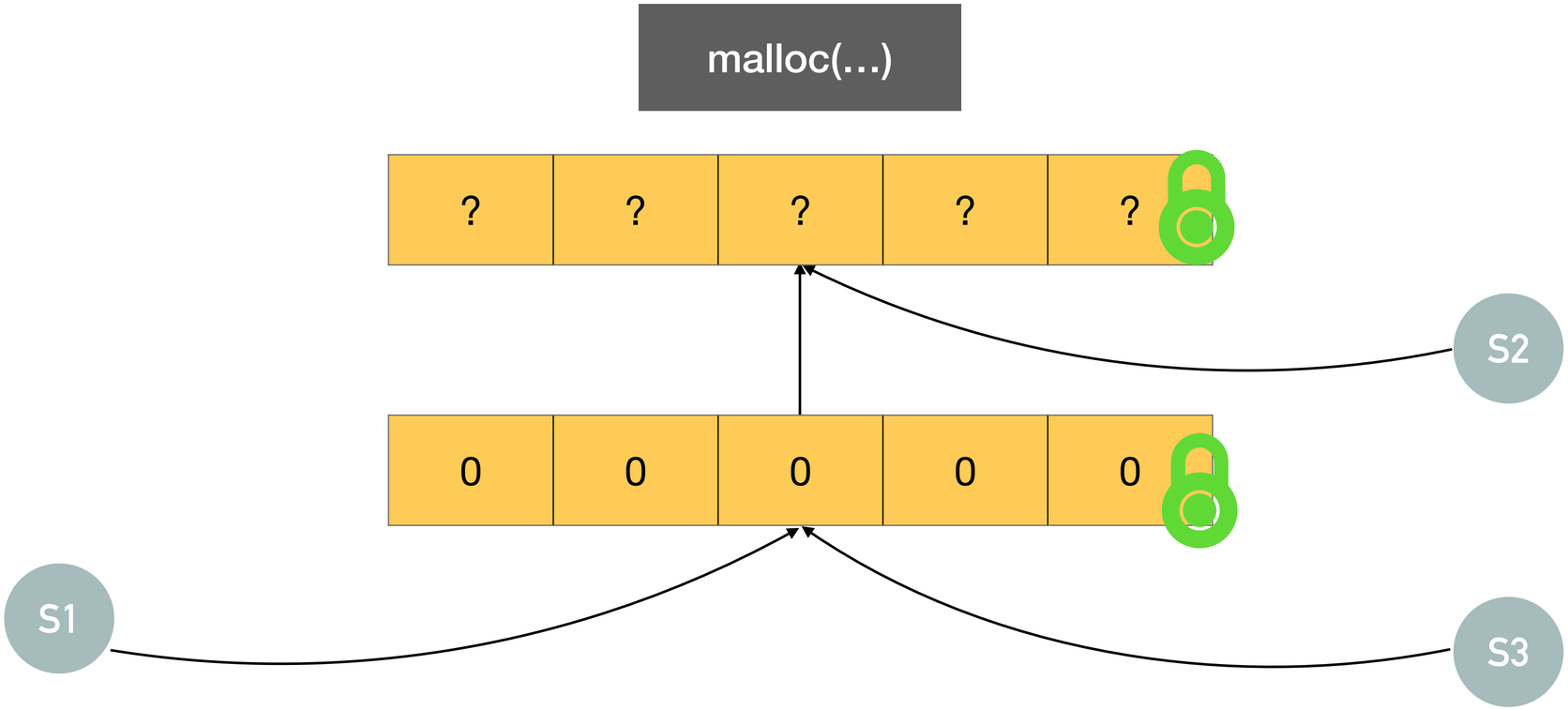
Example Scenario



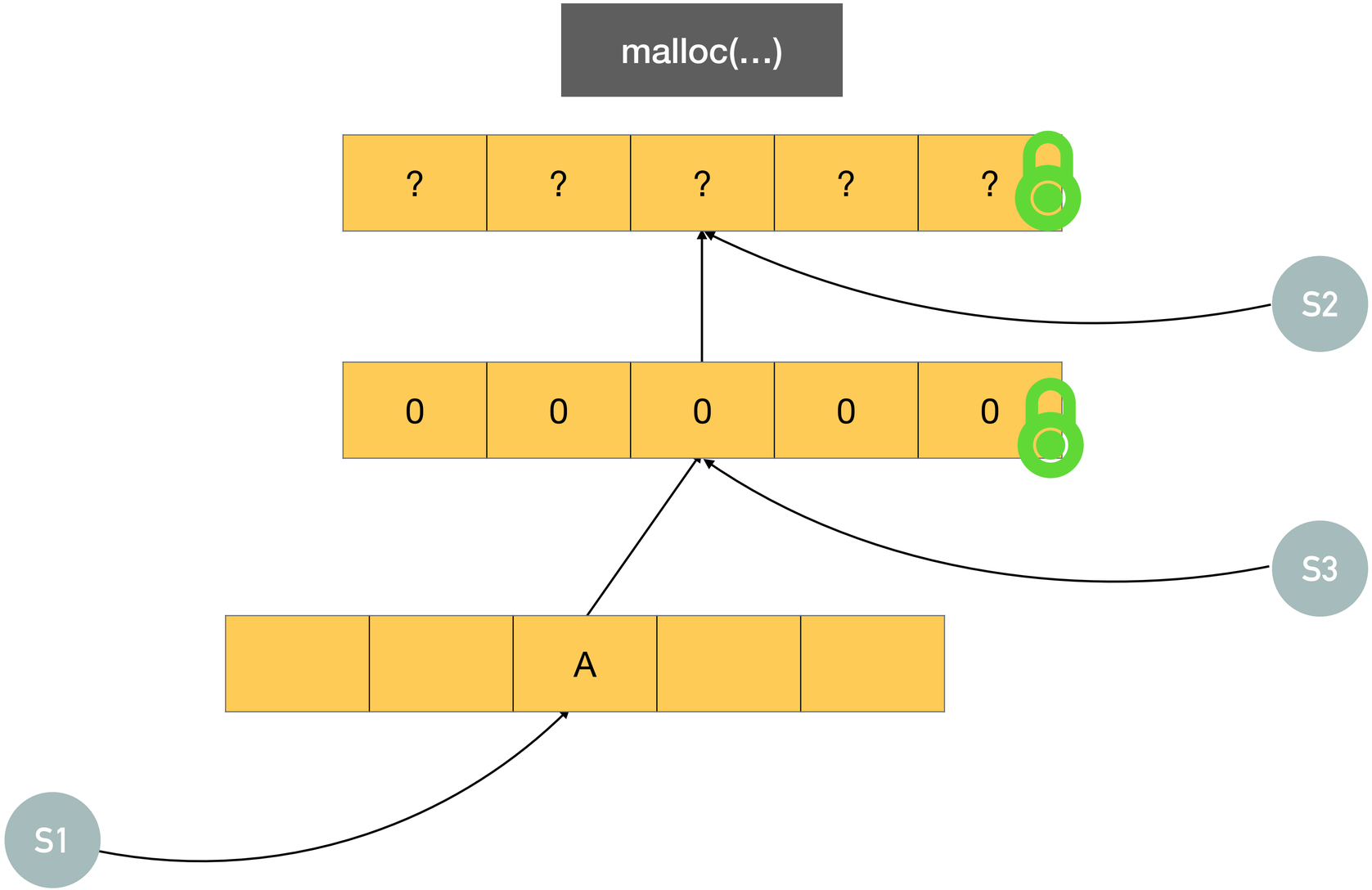
Example Scenario



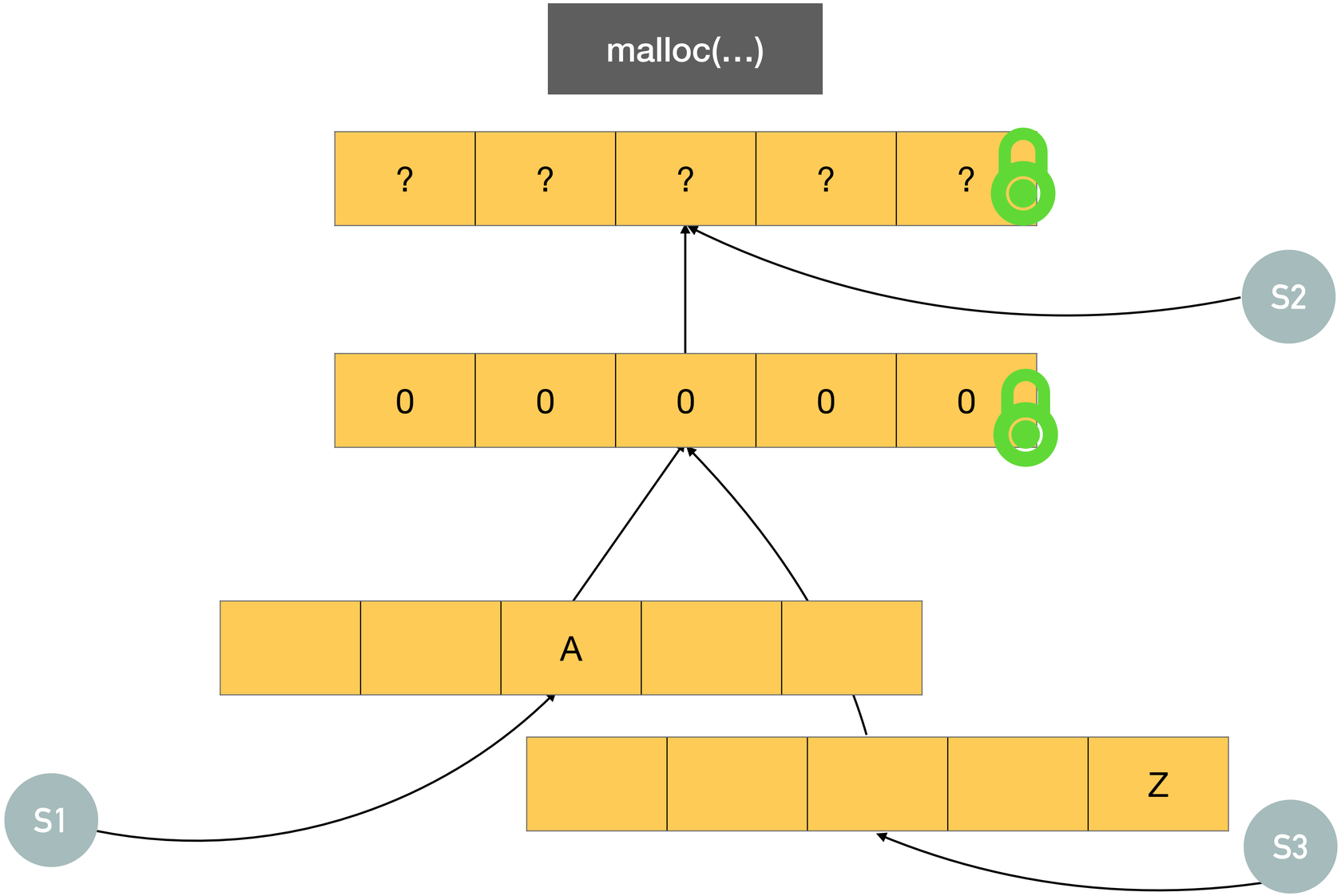
Example Scenario



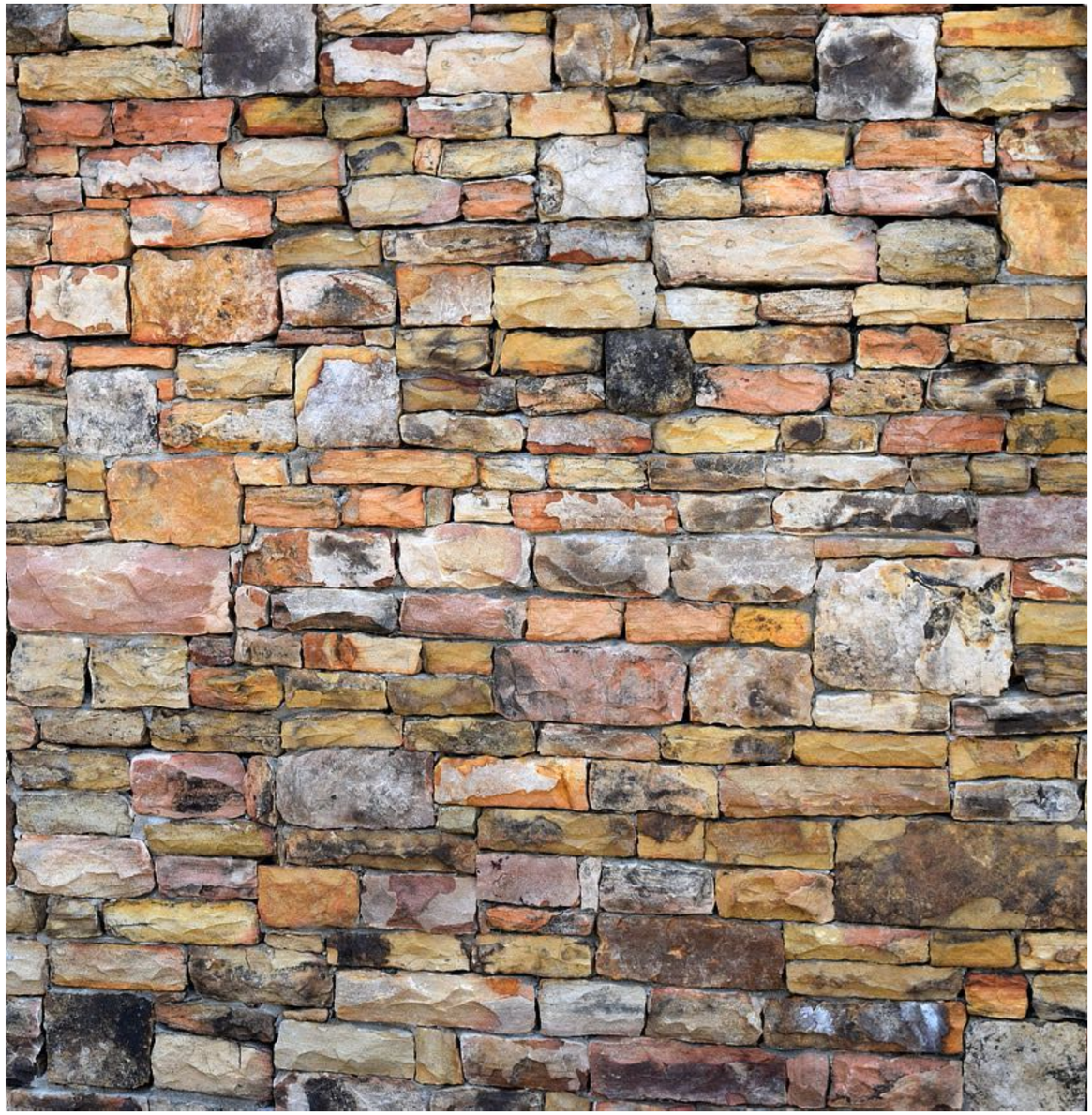
Example Scenario



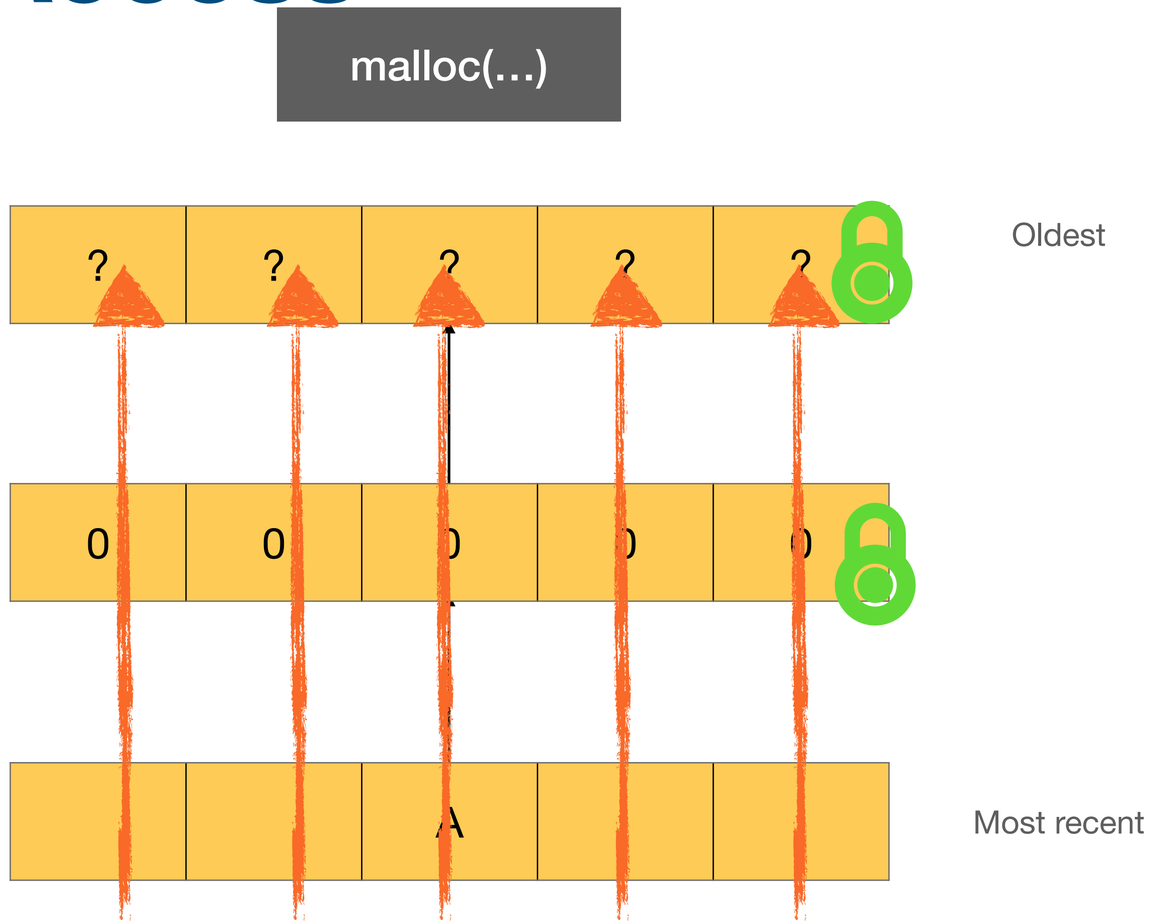
Example Scenario



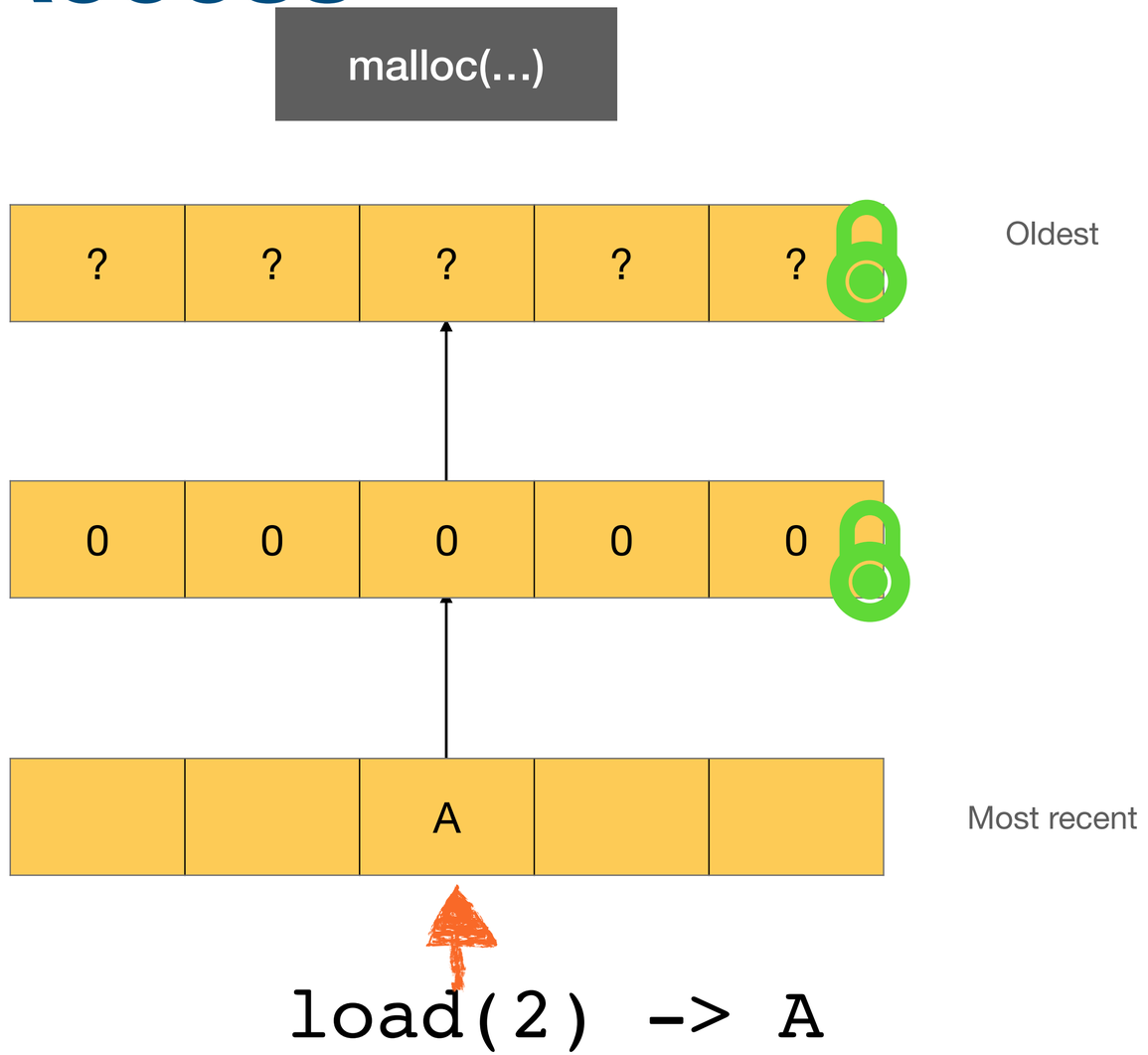
Optimisations



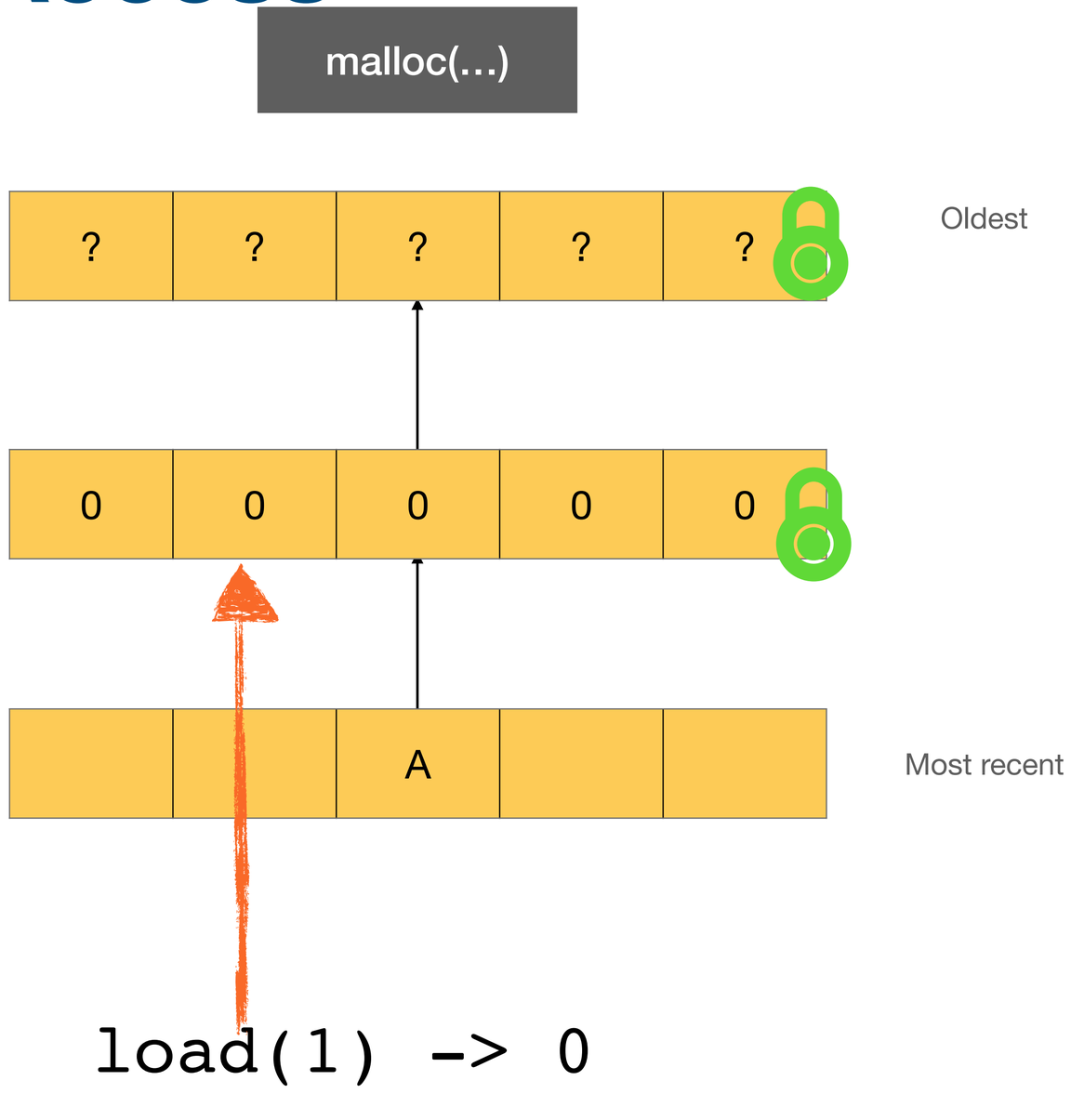
Index-based Access



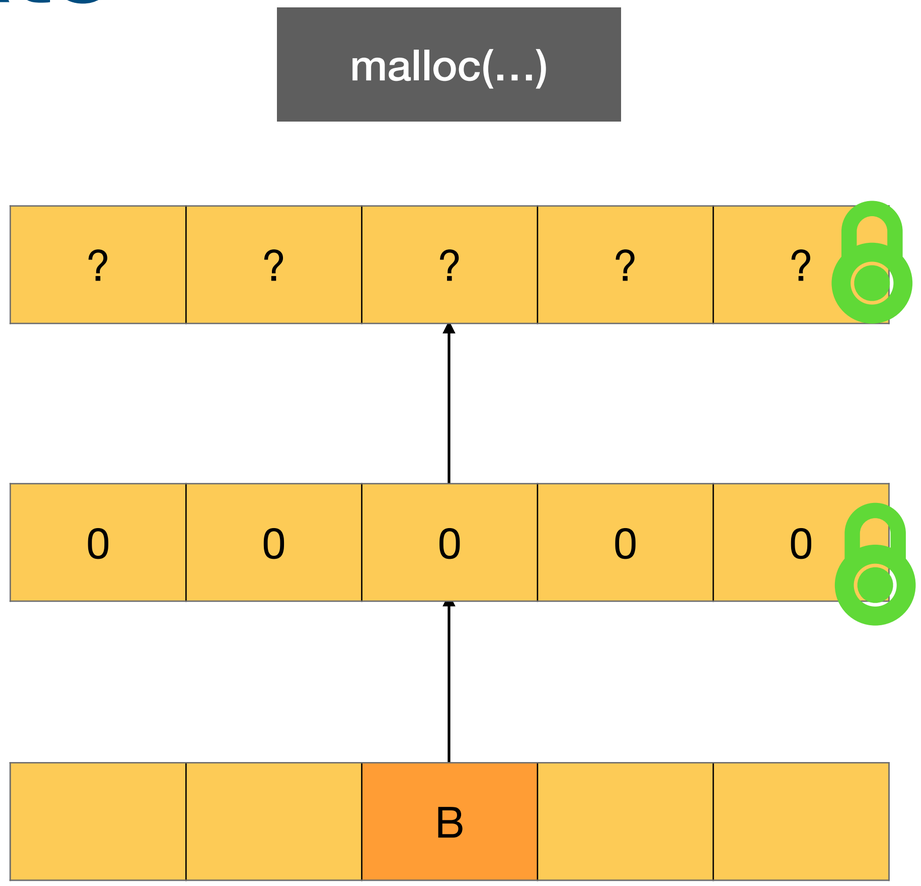
Index-based Access



Index-based Access

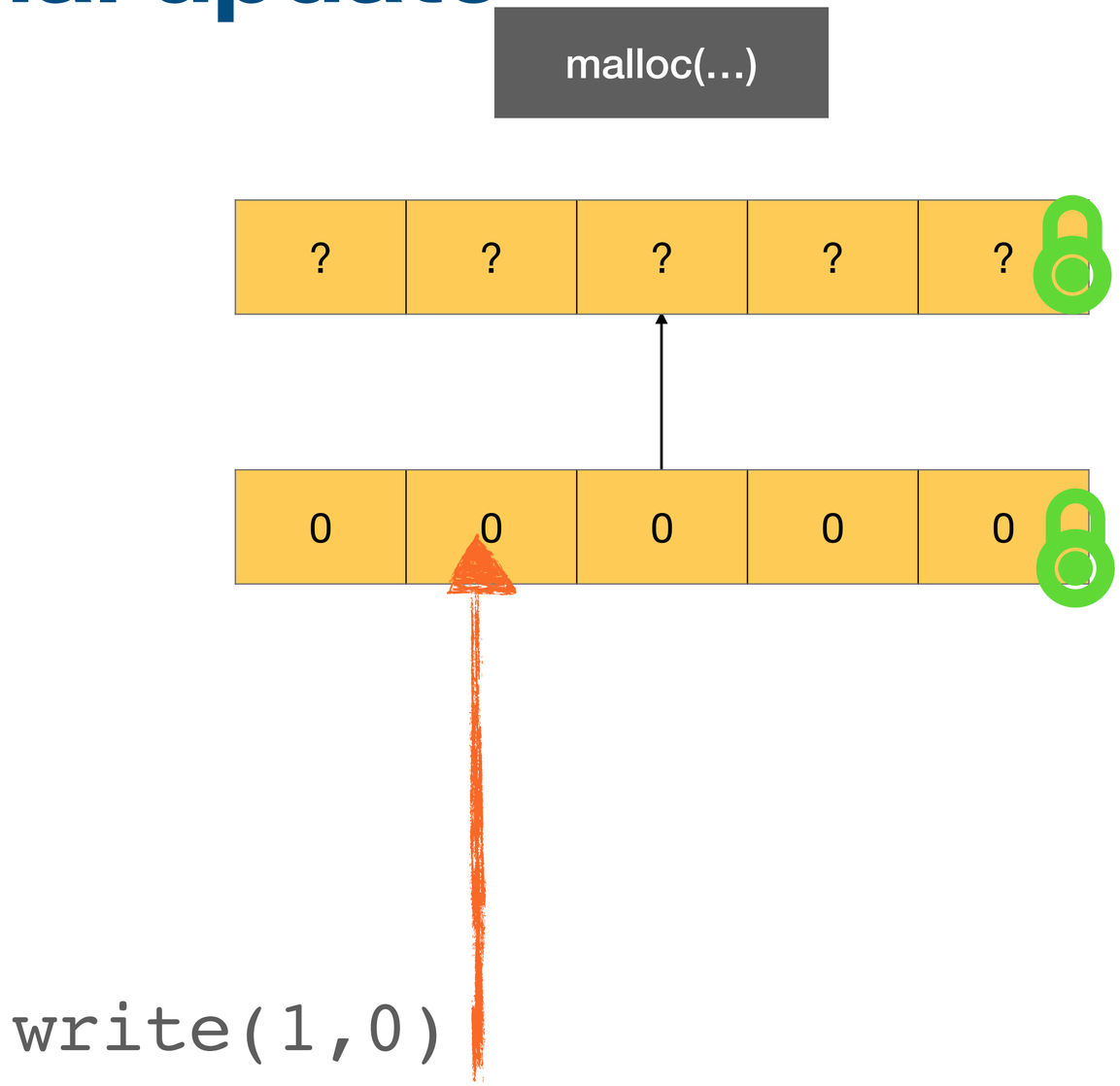


In-place update



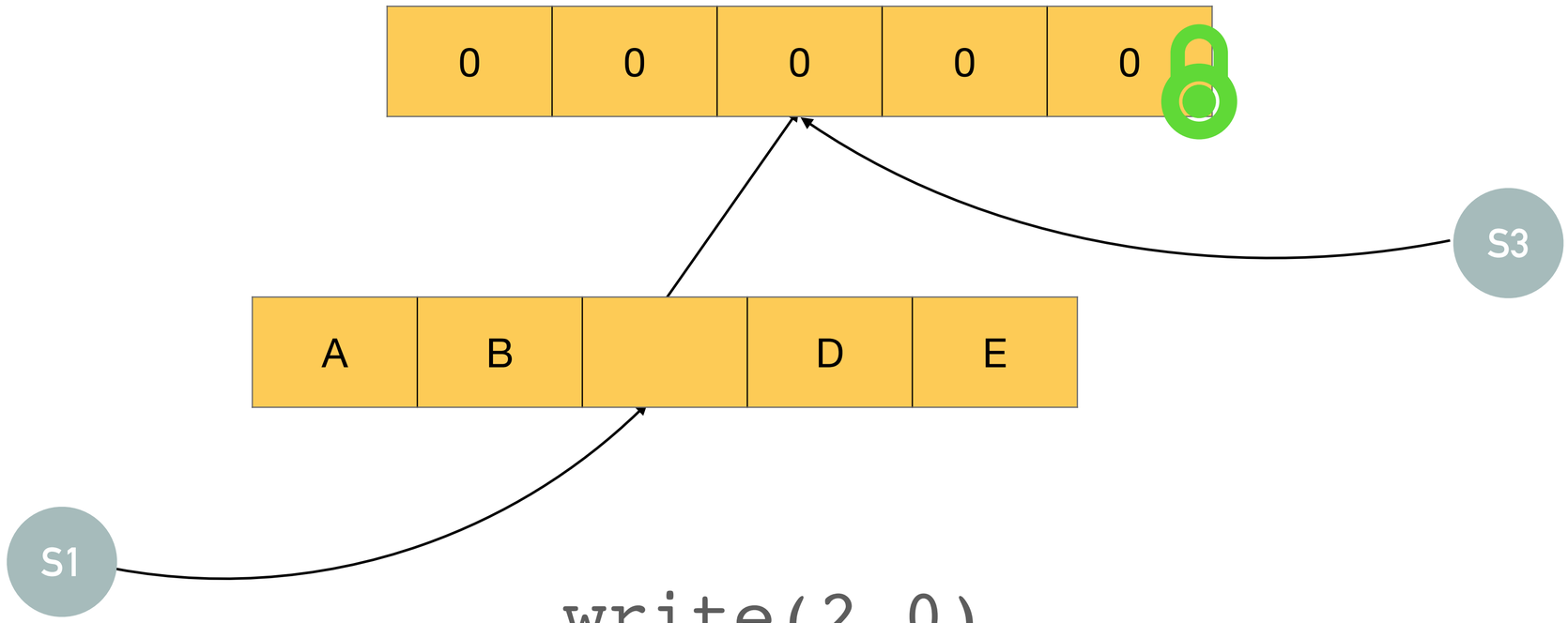

```
write(2,B)
```

Conditional update

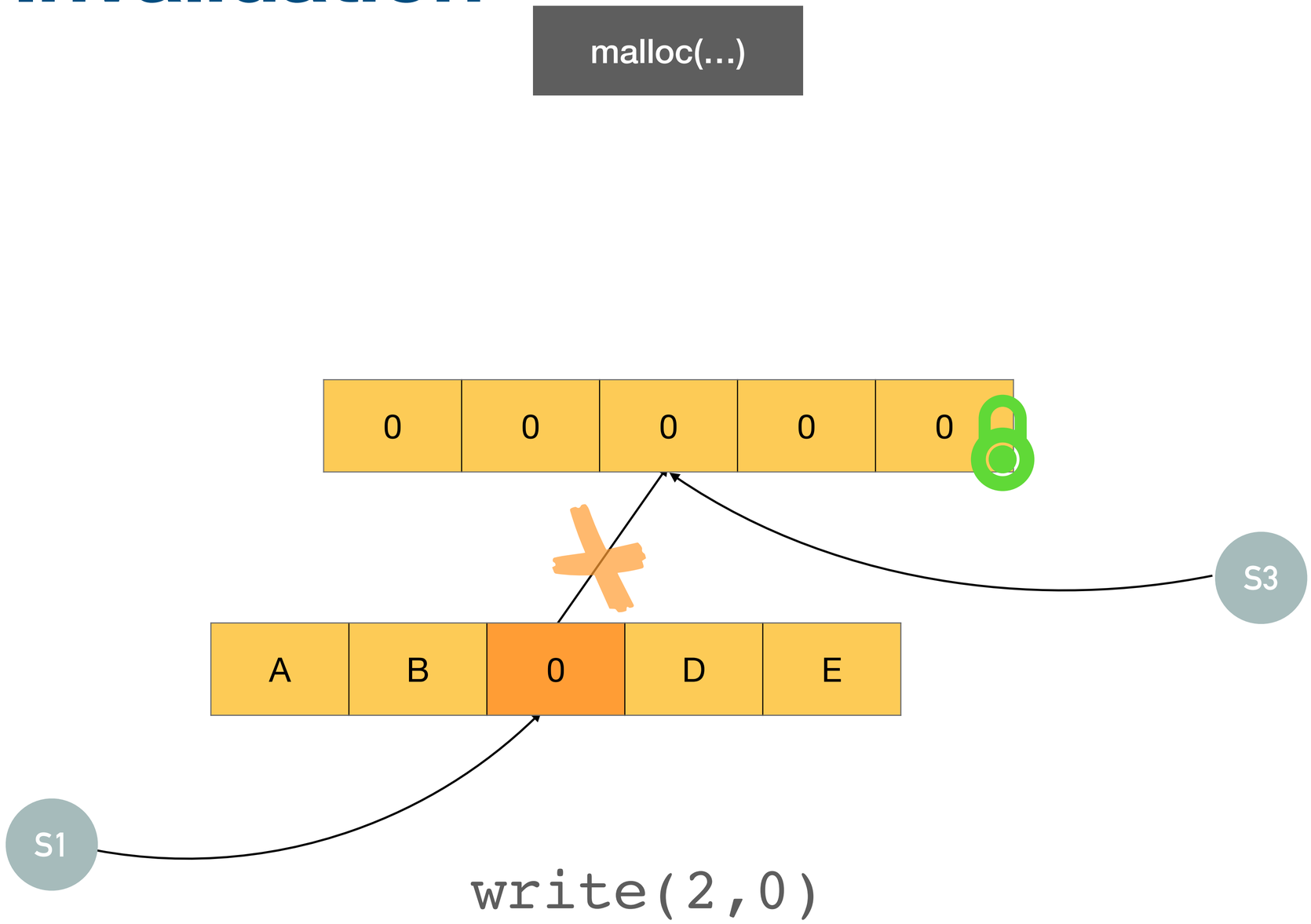


Layer Invalidation

malloc(...)

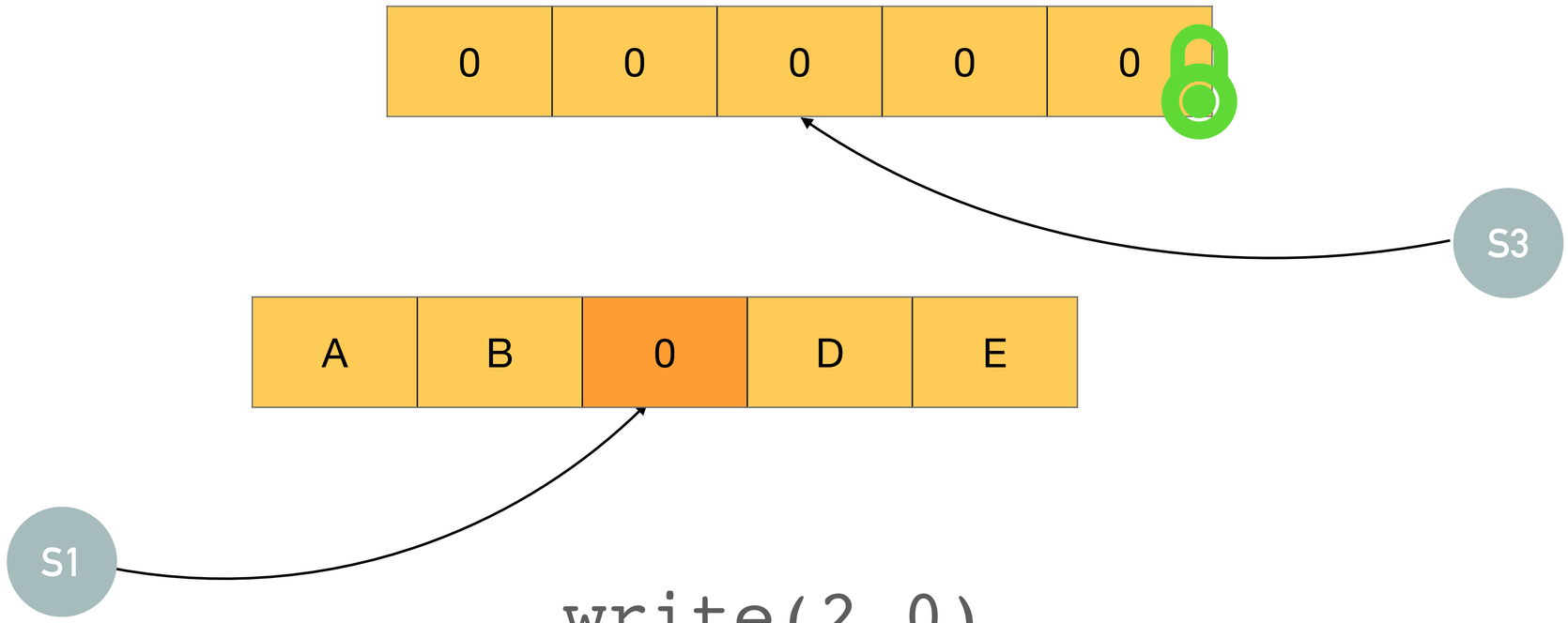


Layer Invalidation



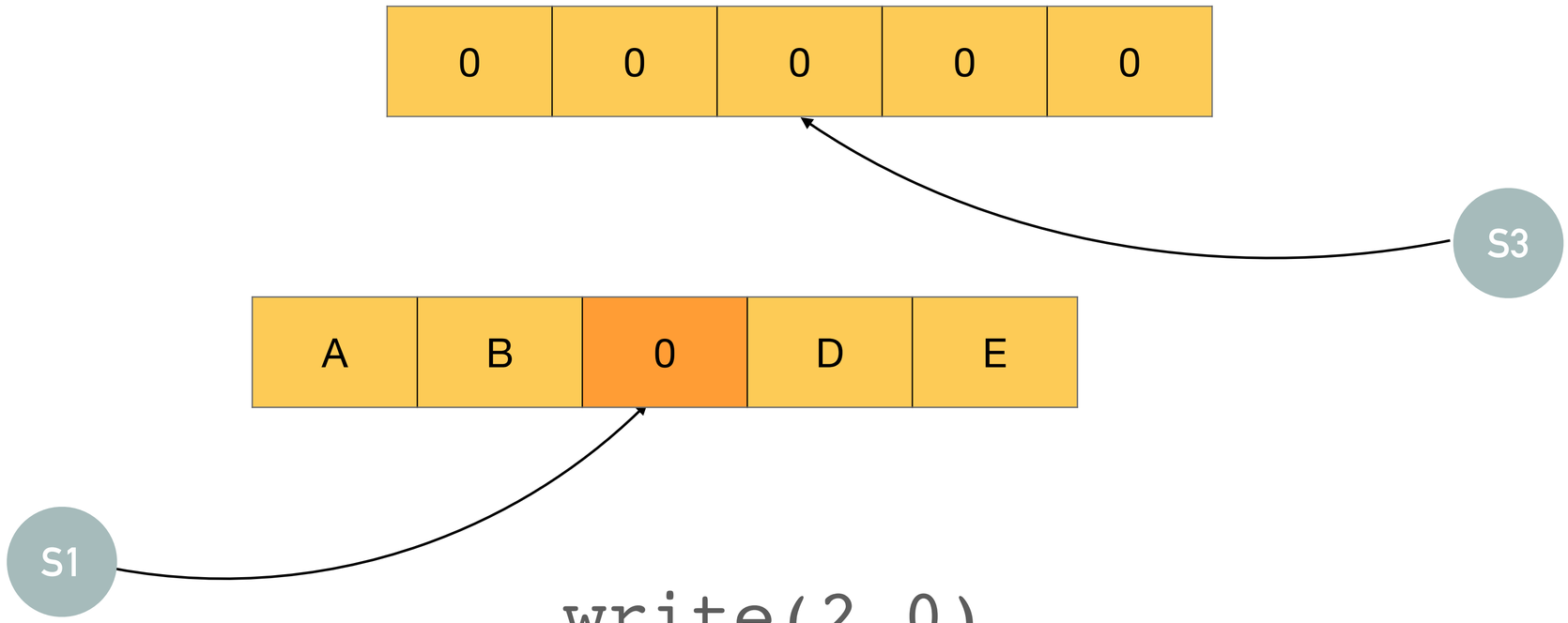
Layer Invalidation

malloc(...)

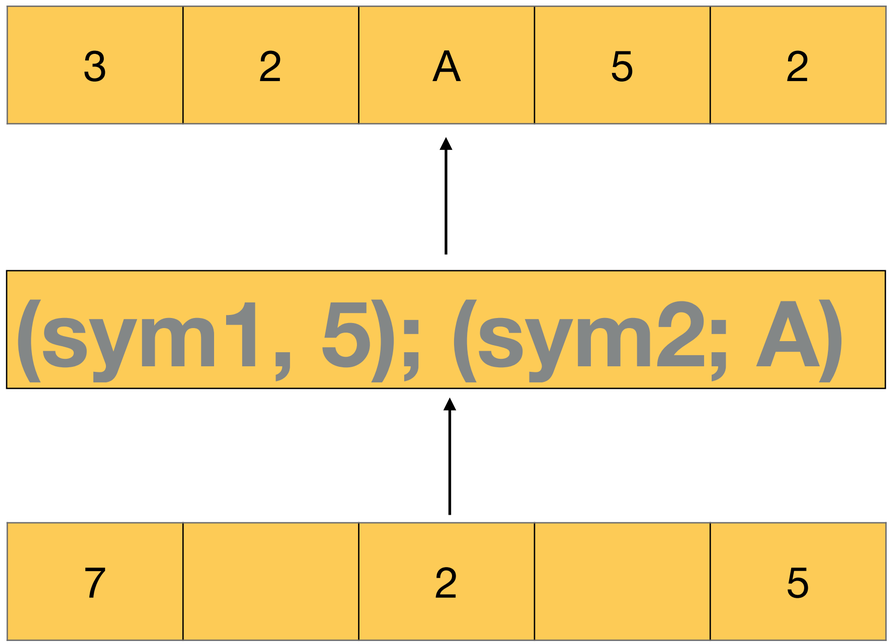


Layer Invalidation

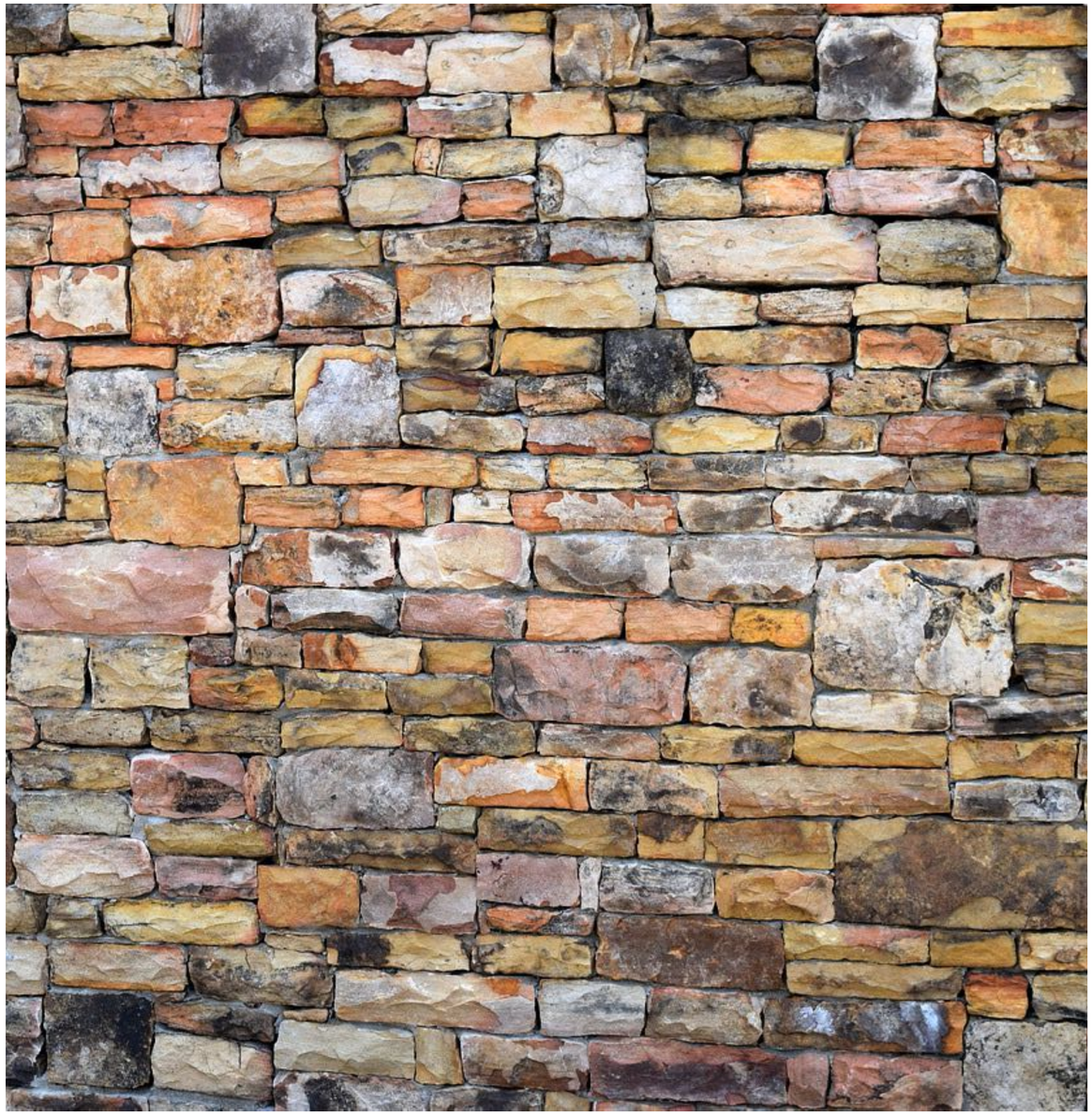
malloc(...)



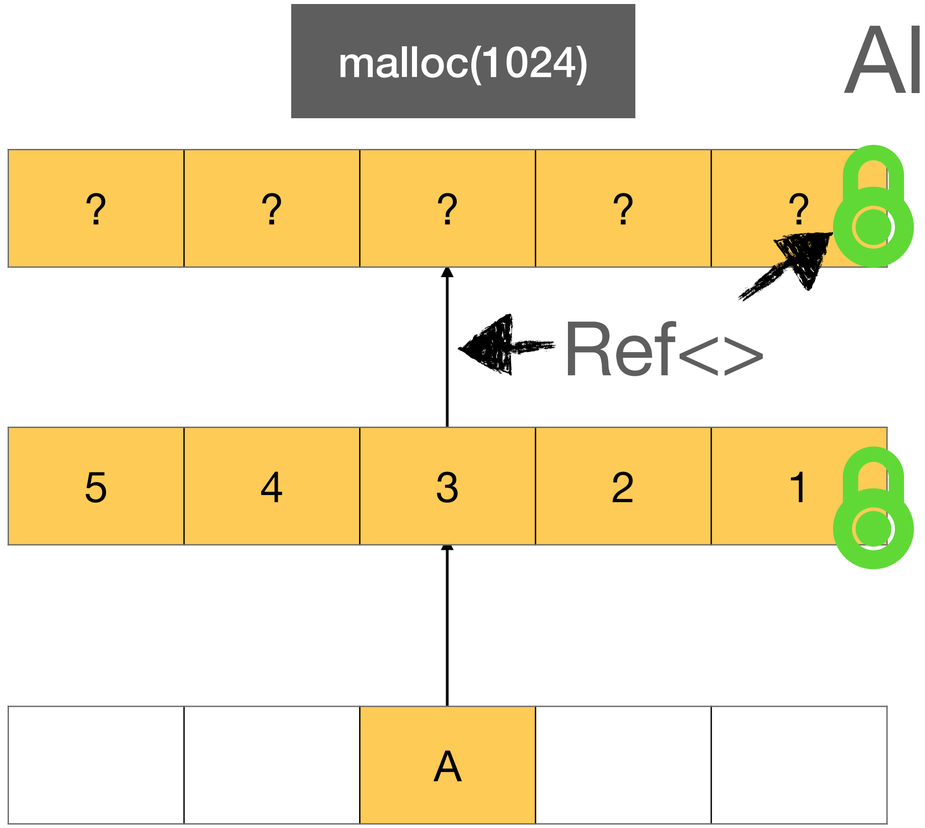
Handling Symbolic Indices



Implementation



Layer Types



Allocated Space

~ 10 byte

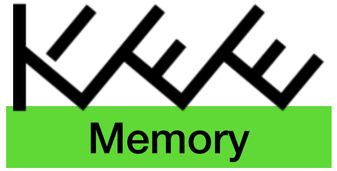
Initialised bytes
sizeof() * 1bit

Map: index -> value

Evaluation

Benchmarks

vs.



GNU Coreutils

Search Strategies

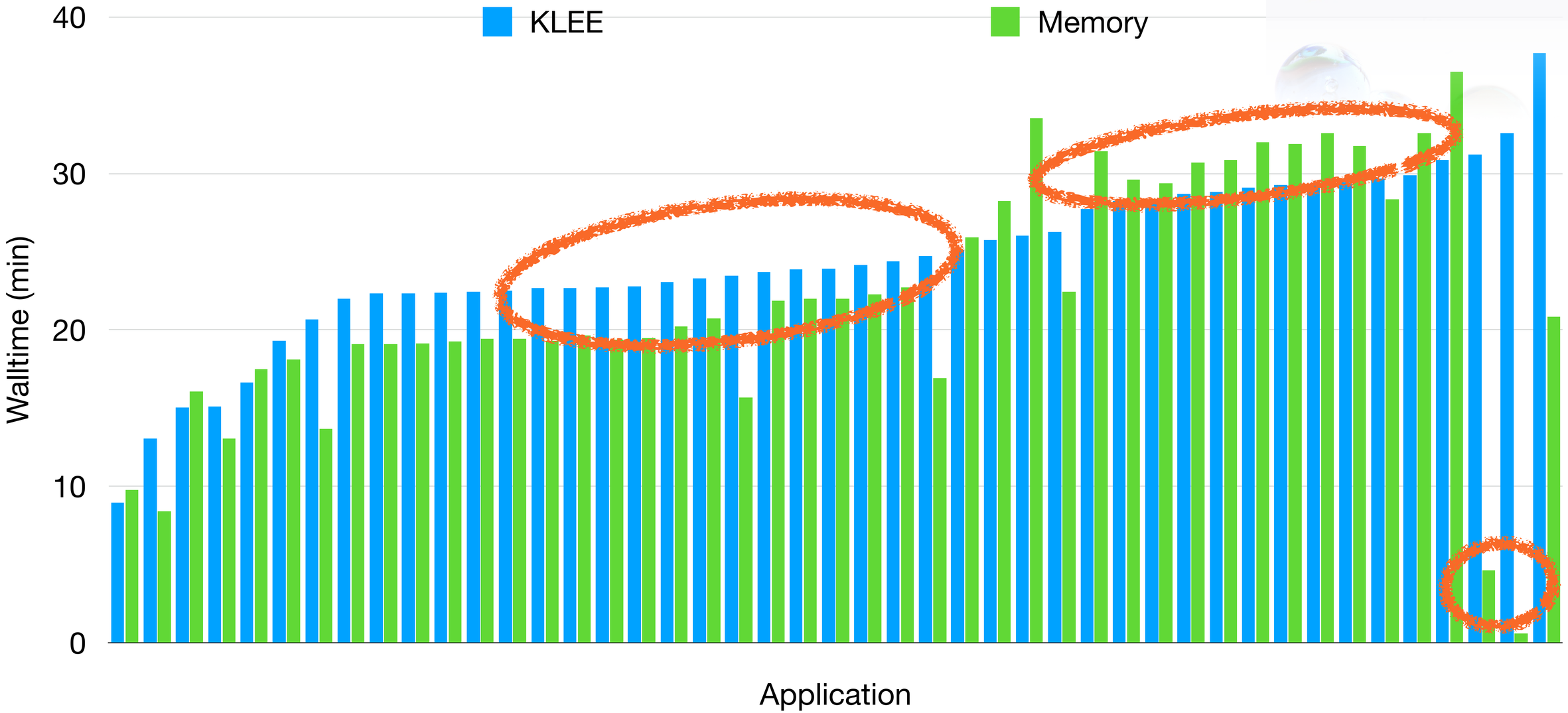
Depth-First

Random + Target Uncovered

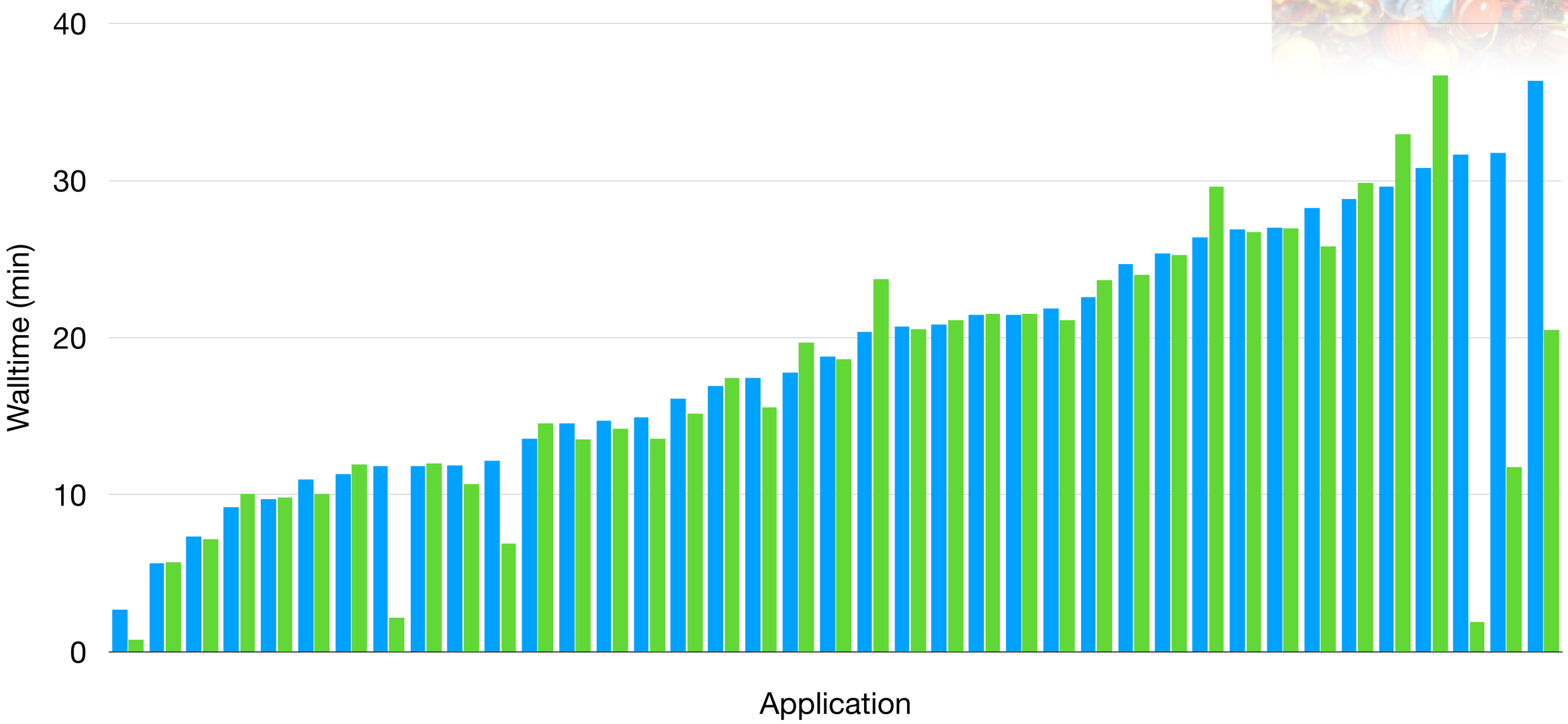
Breadth-First

RQ1: Changes in Execution Time

Walltime - Depth First Search

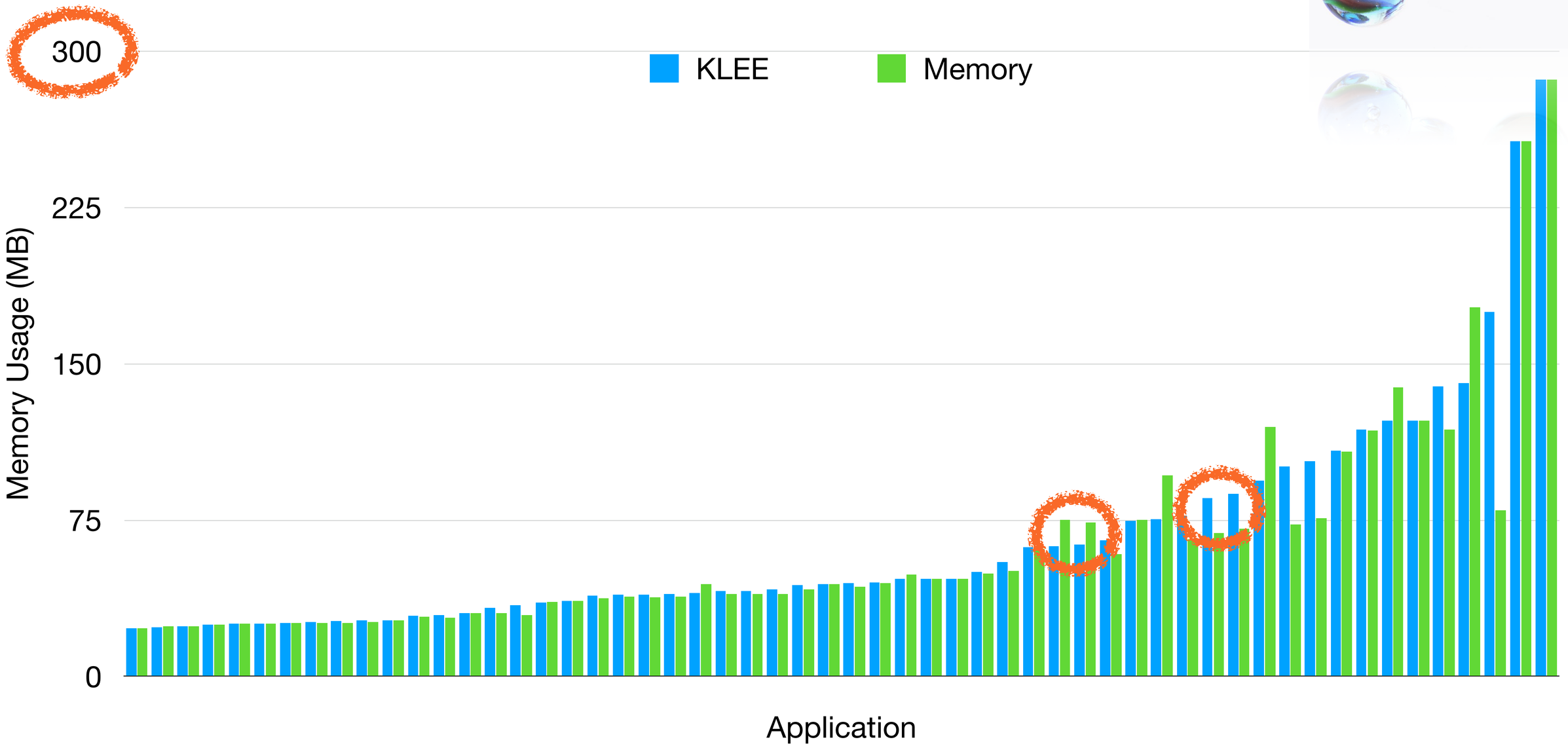


Walltime - Breadth First Search

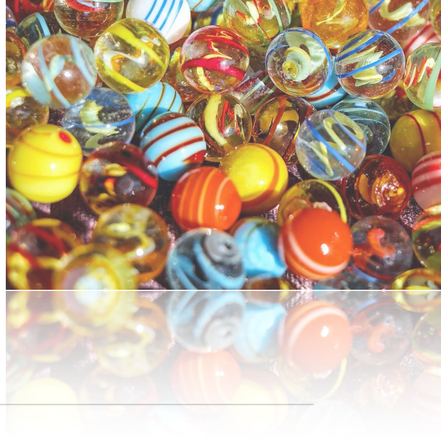


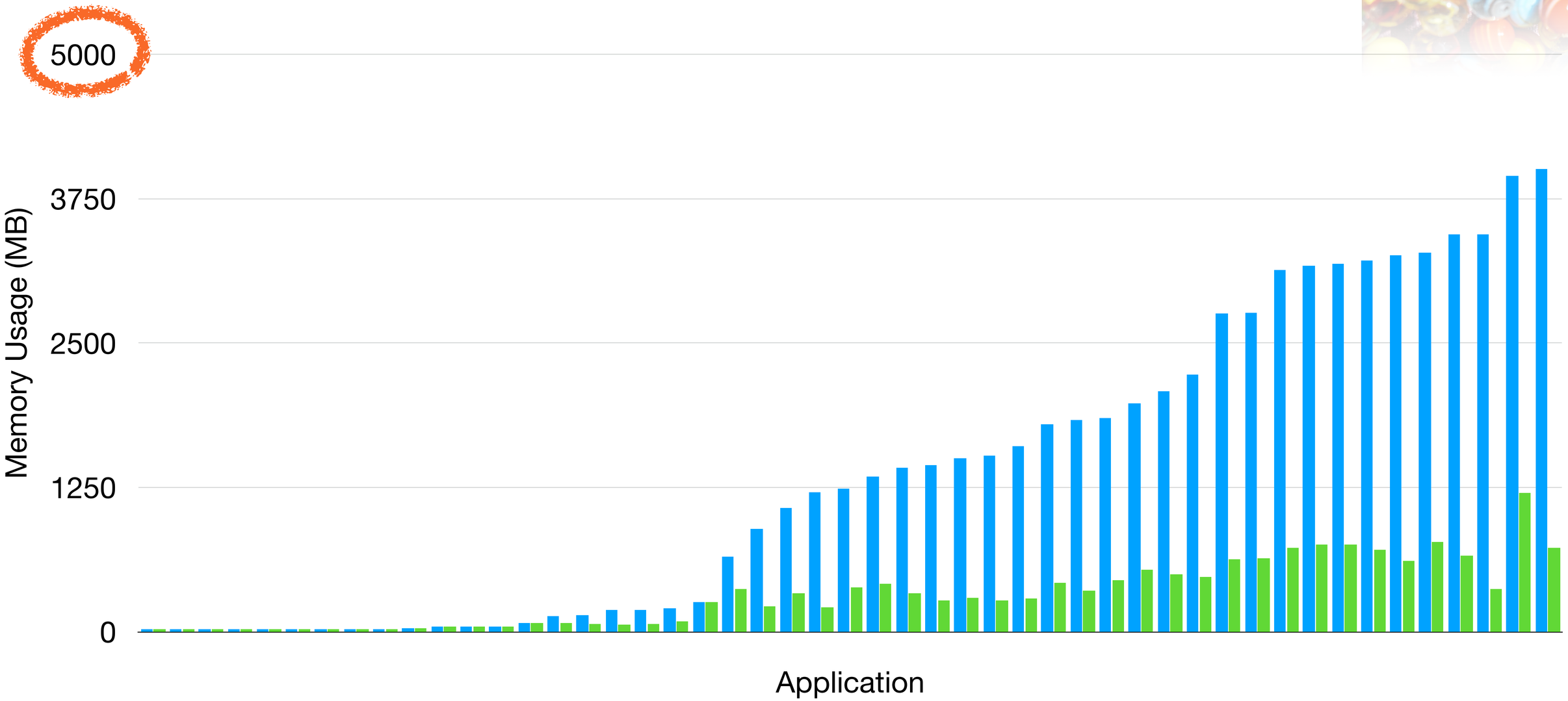
RQ2: Changes in Memory Consumption

Memory Usage - Depth First Search

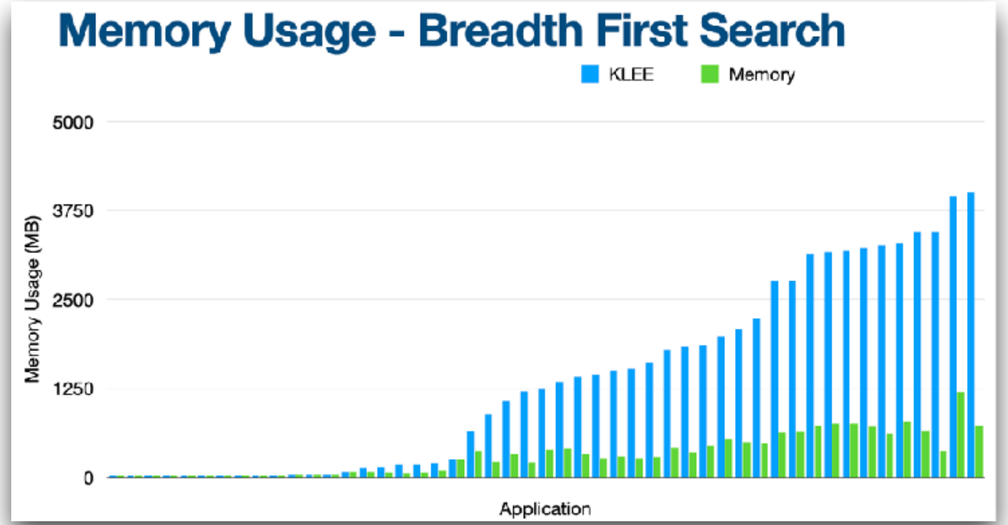


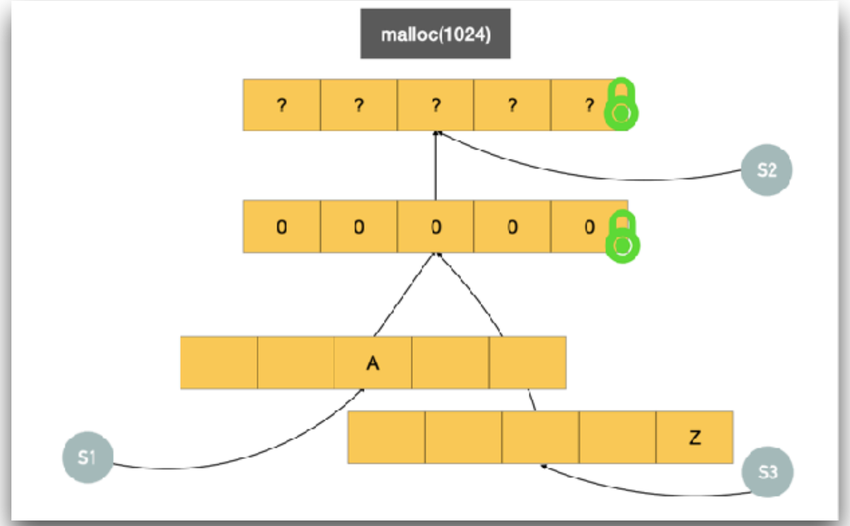
Memory Usage - Breadth First Search

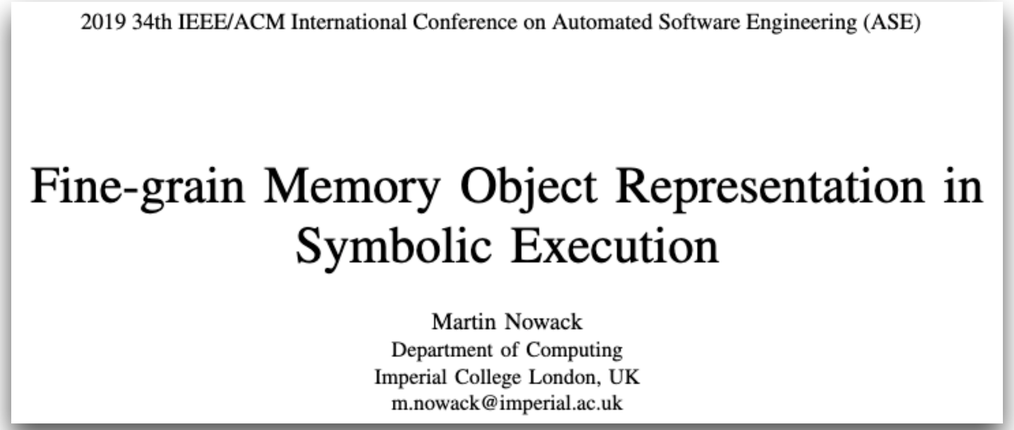




Summary







https://srg.doc.ic.ac.uk/people/martin-nowack/