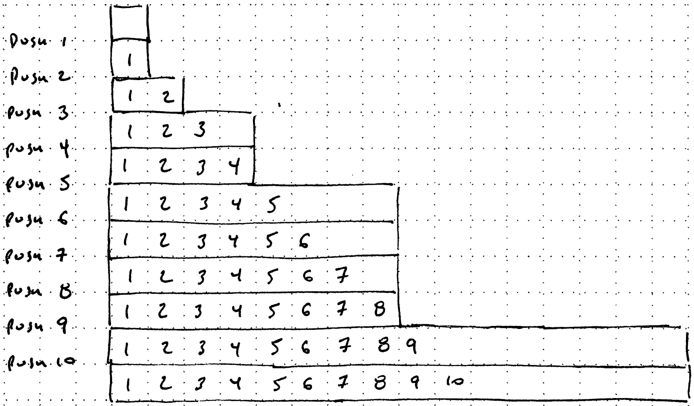
- So far all of our complexity analysis has been for a *single* execution of algorithm or operation (calling `push_back` once)
- Amortized complexity analysis considers the cost of a *sequence* of N executions of algorithm or operation (calling `push_back` N times)

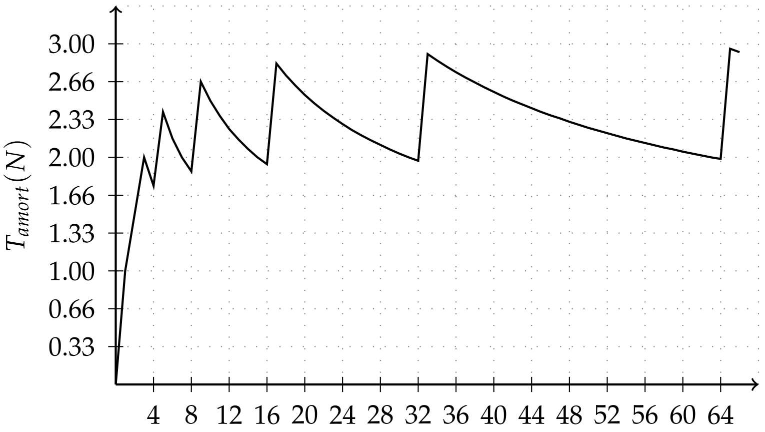
$$T_{amort}(N) = \frac{T_{tot}(N)}{N}$$

- `push_back_v1` always allocates a new array and copies N elements

- `push_back_v2` *sometimes* allocates a new array then copies N elements, and *sometimes* just writes a single element



N	$T(N)$	$T_{tot}(N)$	$T_{amort}(N)$
1	1	1	1.00
2	1+1	3	1.50
3	2+1	6	2.00
4	1	7	1.75
5	4+1	12	2.40
6	1	13	2.17
7	1	14	2.00
8	1	15	1.88
9	8+1	24	2.67
10	1	25	2.50
⋮			
16	1	31	1.94
17	16+1	48	2.82
18	1	49	2.72
⋮			
⋮			
32	1	63	1.97
33	32+1	96	2.91
34	1	97	2.85
⋮			
64	1	127	1.98
65	64+1	192	2.95
66	1	193	2.92


$T_{amort}(N)$