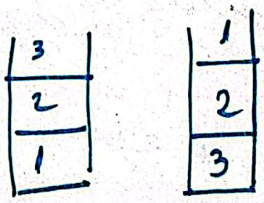
* Complexitate: $N \rightarrow$ timpul maxim de rulare pt. o intrare cu dimensiune N

* Complexitate amortizată $A(N)$, atunci pt
* m operații, timpul total de execuție $O(m A(N))$

Simulare coadă folosind 2 stive

S_{in} S_{out}

Când S_{out} e gol, inserăm toate elementele din S_{in}; insert 1,2,3
pop, pop
insert 4
pop, pop



insert: $O(1)$ CN

pop { $O(N)$ CN / $O(1)$ CA

CN (Cat Neplanabil)
CA (Complexitate Amortizată) - în practică e greu de calculat

---

pop: transfer N elemente
N pop

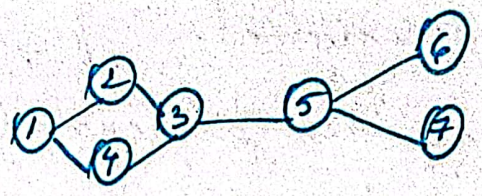
* Complexitate aleatoare / randomizată - depinde de variabile aleatoare; structură aleatoare de date: hash map

C++: STL (vector, nu int tab[]) e.g.
folosim vector<int> tab etc.

heap este priority_queue map - face și sortare
unordered_map ← hash

$V = \{v_1, \dots, v_n\}$

Id → $1, \dots, n$ → pt. grafuri statice
$A, B, C, \dots$



muchia $\{u, v\}$ → uv, vu

Focus: grafuri neorientate, nepondurate

dacă m = nr. muchii atunci $\sum_{v \in V} d(v) = 2 \cdot m$

$N(3) = \{2, 4, 5\}$

$N[3] = N(3) \cup \{3\} = \{2, 3, 4, 5\}$

$M = \{1, 3, 7\}$

$N(M) = \{2, 4, 5\}$

Vecinătatea este disjunctă cu mulț. considerată (deschisă)

Reprezentări: - matrice adiacență $O(n^2)$ spațiu
- liste adiacență $O(n+m)$ optimal

Complexitatea grafurilor depinde de 2 variabile

(m, n) → bivariabilă; m – nr. muchii
n – nr. noduri

$n + \sum_{v \in V} d(v) = n + 2m \rightarrow n + m$

$\left.\begin{matrix} uv \\ u < v \end{matrix}\right\}$ ⇒ punem v în lista lui u – optimizare

$m \geq \frac{n}{2} \Rightarrow O(n+m) = O(m)$