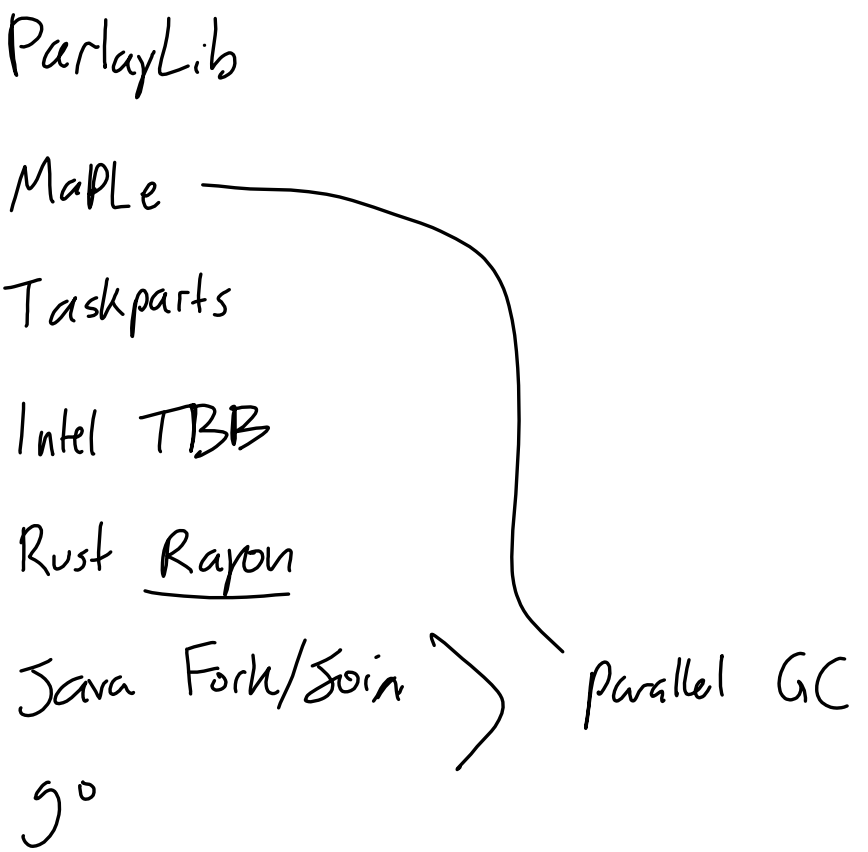
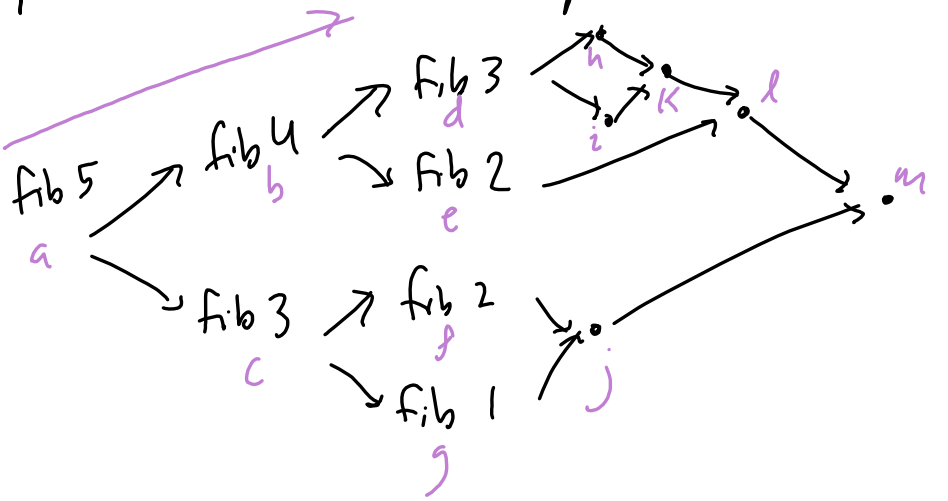

ParlayLib
MaPLe
Taskparts
Intel TBB
Rust Rayon
Java Fork/Join
Go
parallel GC

Greedy Scheduling

"offline model"

- take computation graph as input, #Processors
- produce schedule as output



t	Processors 0	1	2	3
0	a	-	-	-
1	b	c	-	-
2	d	e	f	g
3				
4				
5				
6				

work W

span S

Defn:
Ready vertex:
on timestamp t, vertex v is ready if all dependencies were executed before t.

length of schedule : T

lower bounds : $W/P \leq T$

$S \leq T$

greedy scheduling guarantees : $T \leq W/P + S$

$|\text{work bucket}| + |\text{idle bucket}| = T \cdot P$

$|\text{work bucket}| = W$

on every timestep, each processor is either working or idle

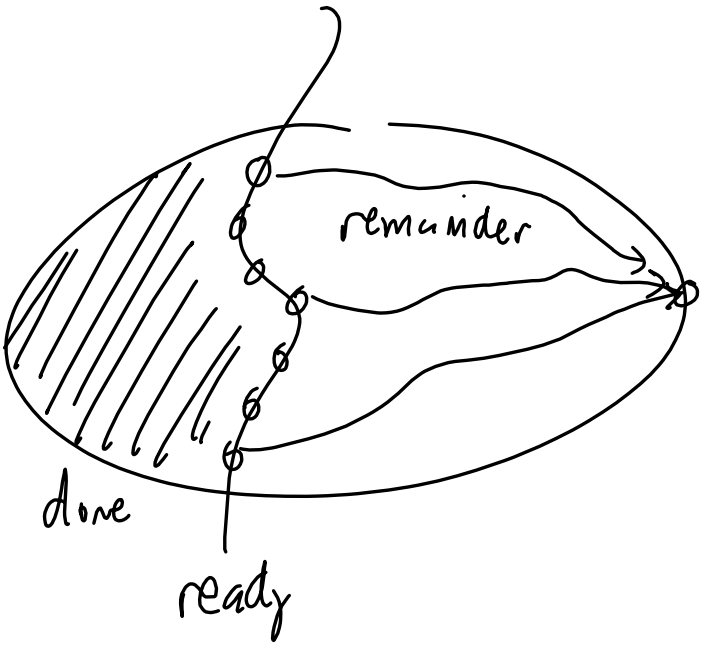
---

two kinds of timesteps :

Saturated : every processor working

unsaturated : at least one procesor idle

$\#\text{unsatured} \leq S$



$$|\text{work bucket}| = W$$

$$|\text{idle bucket}| \leq (\#\text{unsaturated rounds}) \cdot (P-1)$$

$$= S(P-1)$$

$$|\text{work}| + |\text{idle}| = T \cdot P \leq W + S(P-1)$$

$$T \leq W/P + \frac{S(P-1)}{P}$$

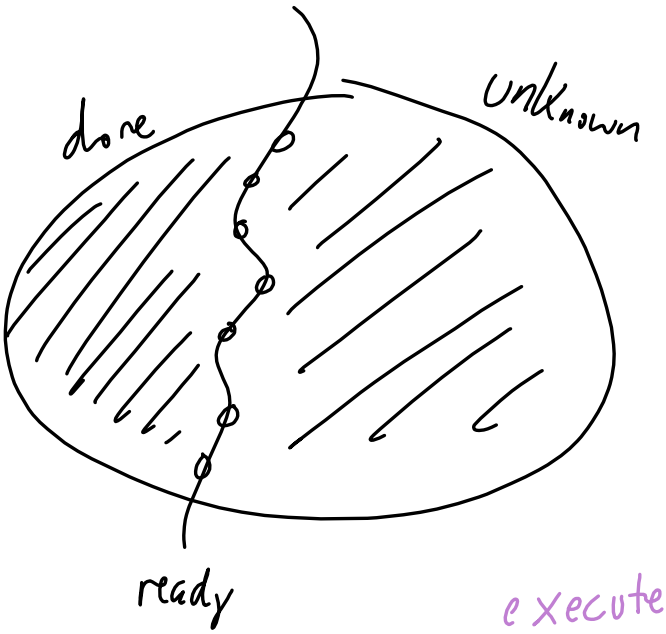
$$T \leq W/p + \frac{S(P-1)}{P}$$

$$T \leq W/p + S$$

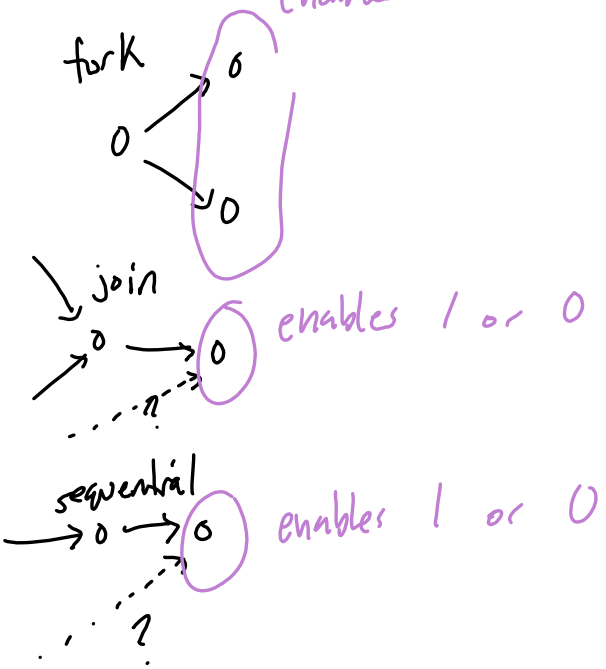
Offline greedy scheduling

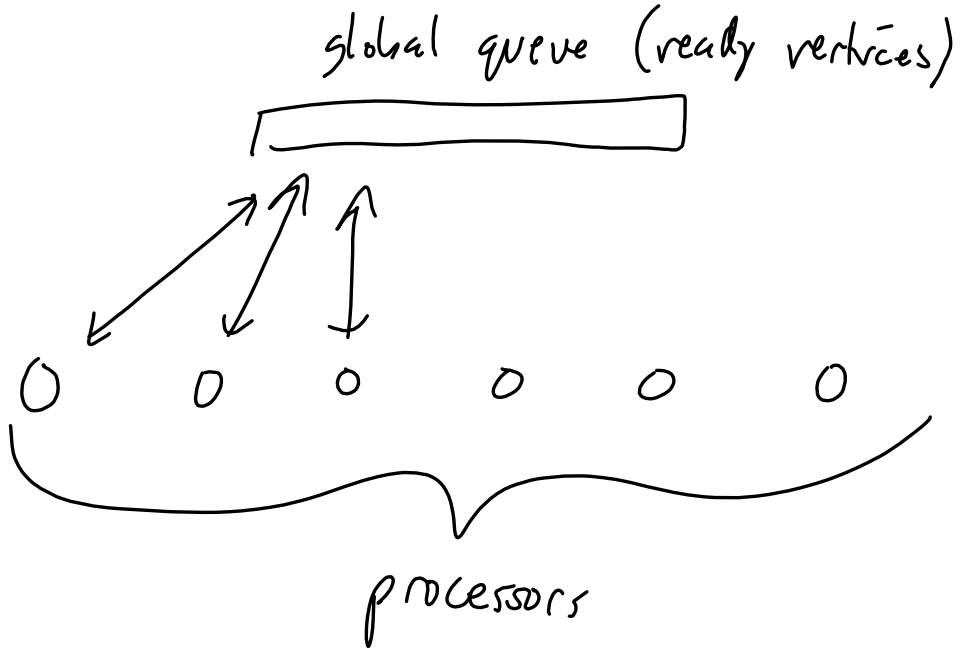
↓

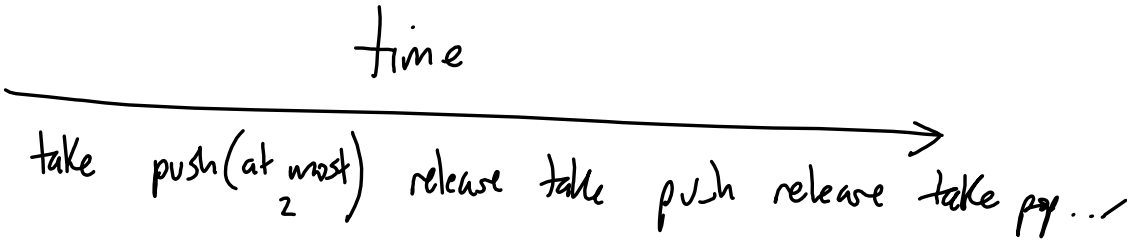
how to do <u>online</u>?



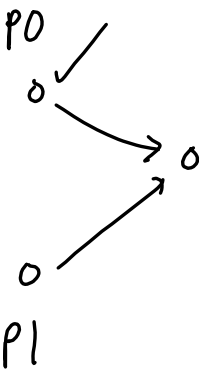
execute(v) → list(v) ↑ enabled

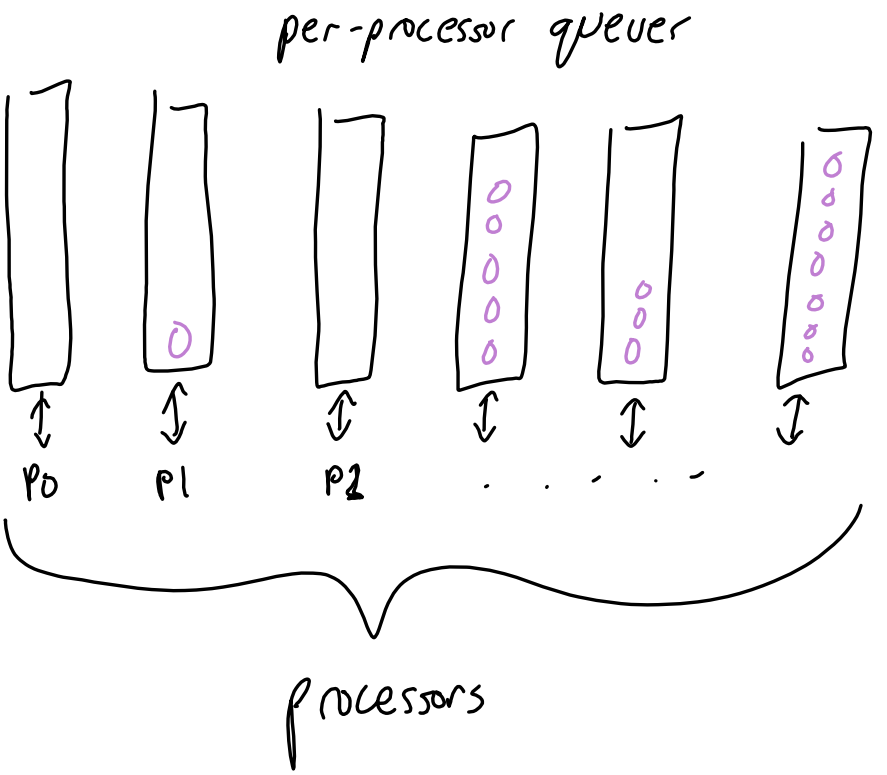



global queue (ready vertices)
processors

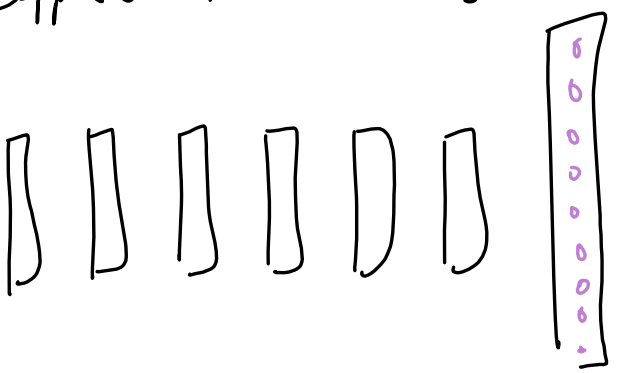

time
take
push (at most 2)
release
take
push
release
take
pop...

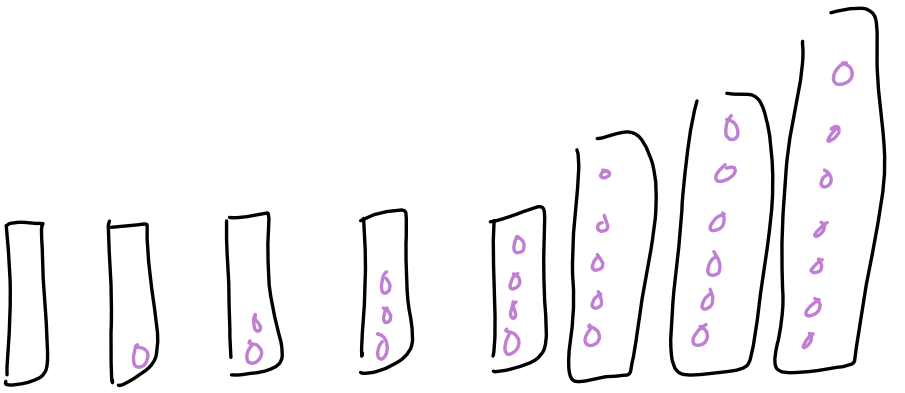
$$T \geq \Omega(W)$$


P0
P1



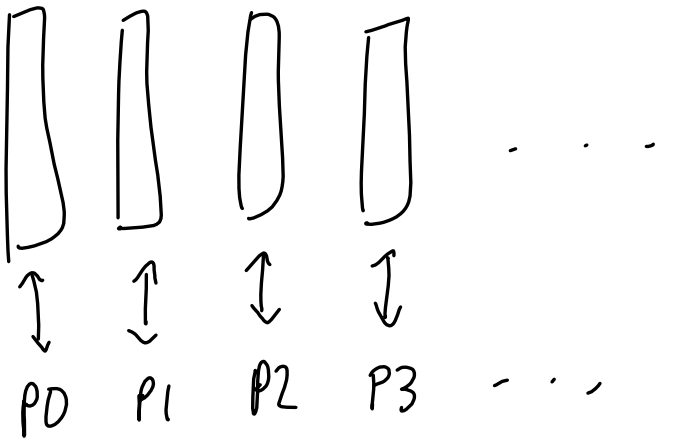
Suppose: round-robin search





"symmetry breaking"

randomized work-stealing

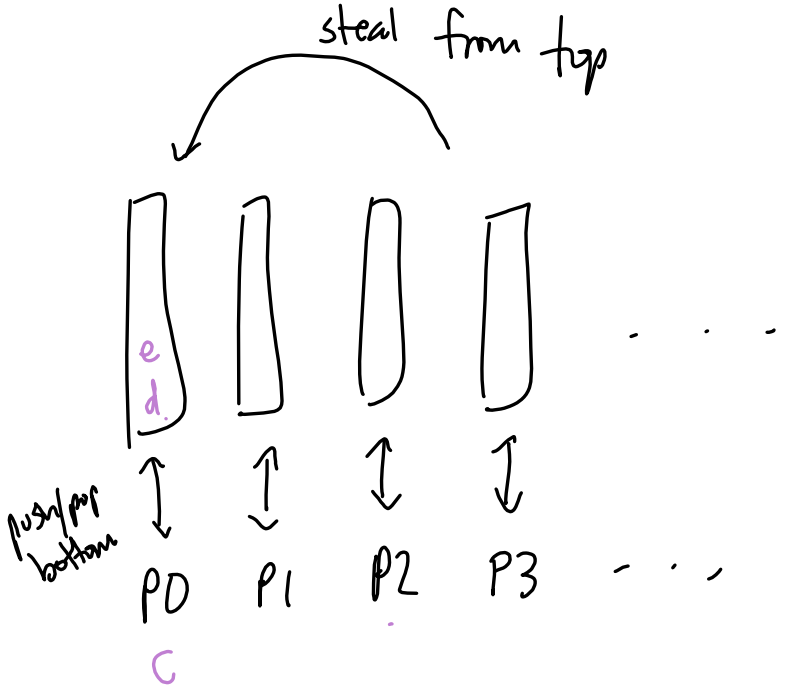


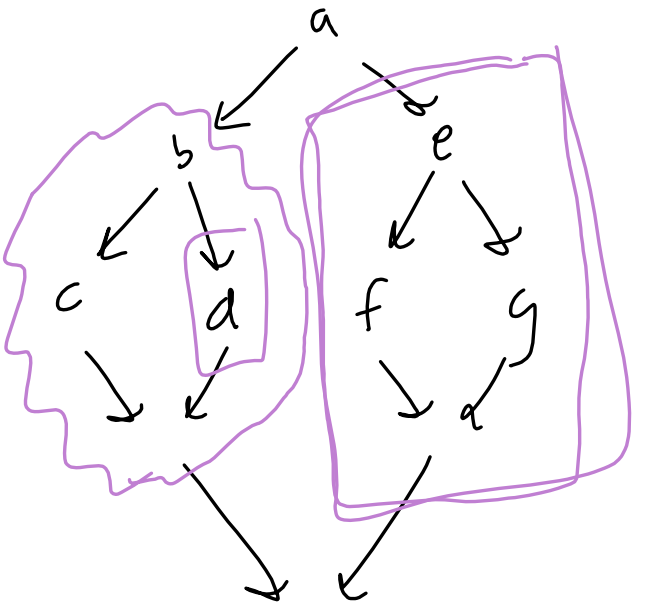
working phase: push/pop from own queue

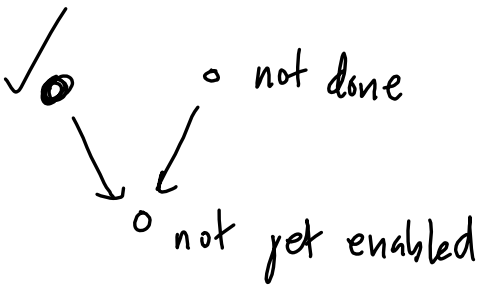
stealing phase: if my queue is empty, randomly choose a victim and try to steal one vertex

"steal half"

"contention": two processes competing for same resource


steal from top
e
d
push/pop bottom
P0
P1
P2
P3
c

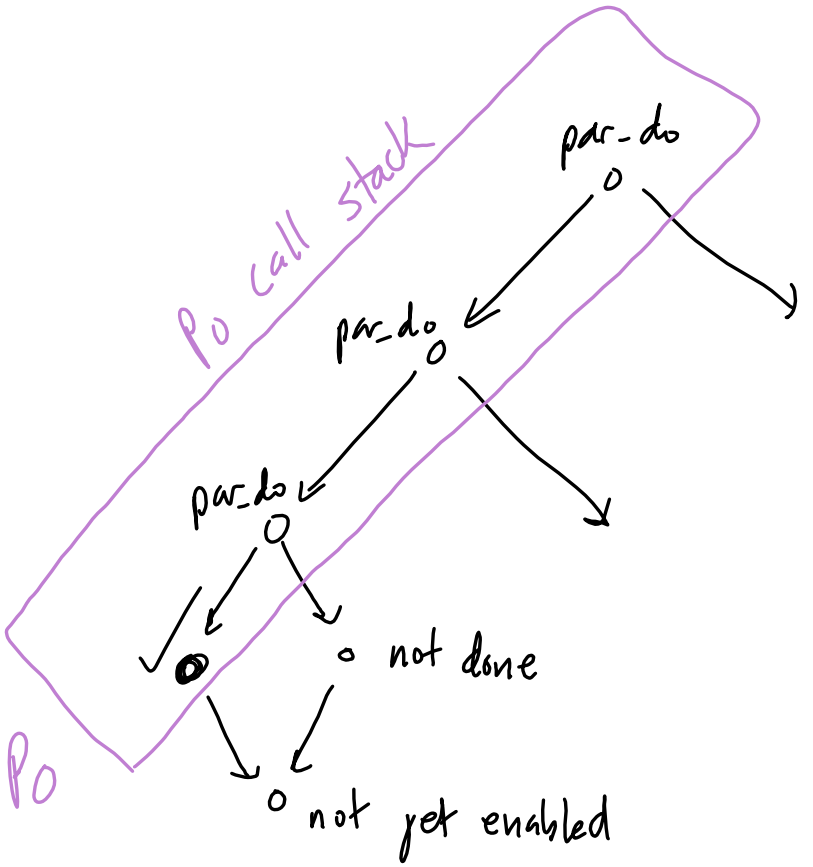

a
b
e
c
d
f
g



"burying the join"

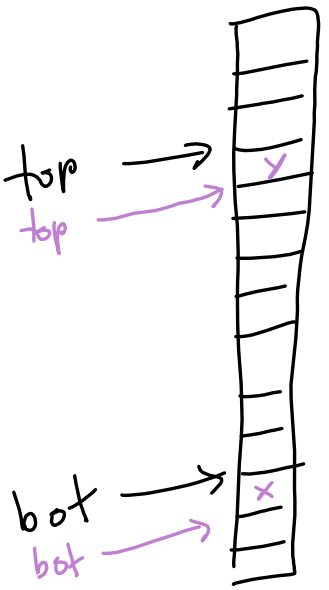
Parlay Lih

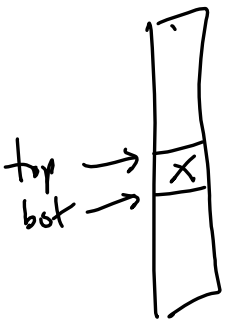
Rust Rapon


P0 call stack
par_do
par_do
par_do
not done
not yet enabled
P0

y = steal()

"deque"
doubly-ended queue





push(x)