Page Fault → recall concept

OS program

Question : How does the page fault handler decide which main memory to replace when there is a page fault and no empty spots?

Aside: Disk access speed

- Processor vs main memory ⇒ 2 orders of magnitude difference
  1ns ~100ns

- Hard disk speed? ⇒ msecs.
  ⇒ order of 4 slower.

- So, the OS page fault handler code must be written based on a realistic model

Page Replacement Policies

Principle of Locality of Reference: If memory address A is referenced at time t, then it and its neighbouring memory locations are likely to be referenced in the near future.

commonly seen

Temporal Locality of Reference

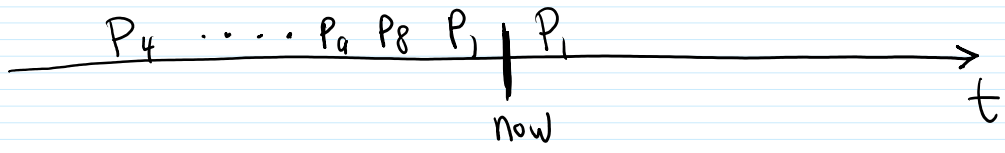
Spatial Locality of reference

- Locality of Reference ⇒ heuristic argument

	Same Address (temporal)	Neighbours (spatial)
Instructions	- Loops - functions	- Loops - sequential code
Data	- local variables - loop variables	- Array (stepping through)

Data	- local variables - loop variables (index)	- Array (stepping through)
	. . .	

- Based on this principle, what would be a good page replacement policy?



- Let's a page fault occurs for page Px
  - ↳ $P_1, P_2, \ldots P_n$

* Pick from them the page that was referenced least recently. *

① LRU – Least Recently Used – Policy.

ⓐ • Keep track of when each page was last used.
  - ↳ timestamp
  - ↳ LRU page → smallest timestamp

ⓑ • Or, keep track of the recently used pages → stack
  - ↳ LRU: at the bottom of the stack
  - ↳ LRU must update every memory access

* LRU might be too expensive in practise *