بسم الله الرحمن الرحیم

# ۲۳۶

# «سیستم عامل»

جلسه ۲۳: مدیریت حافظه (۱۰)

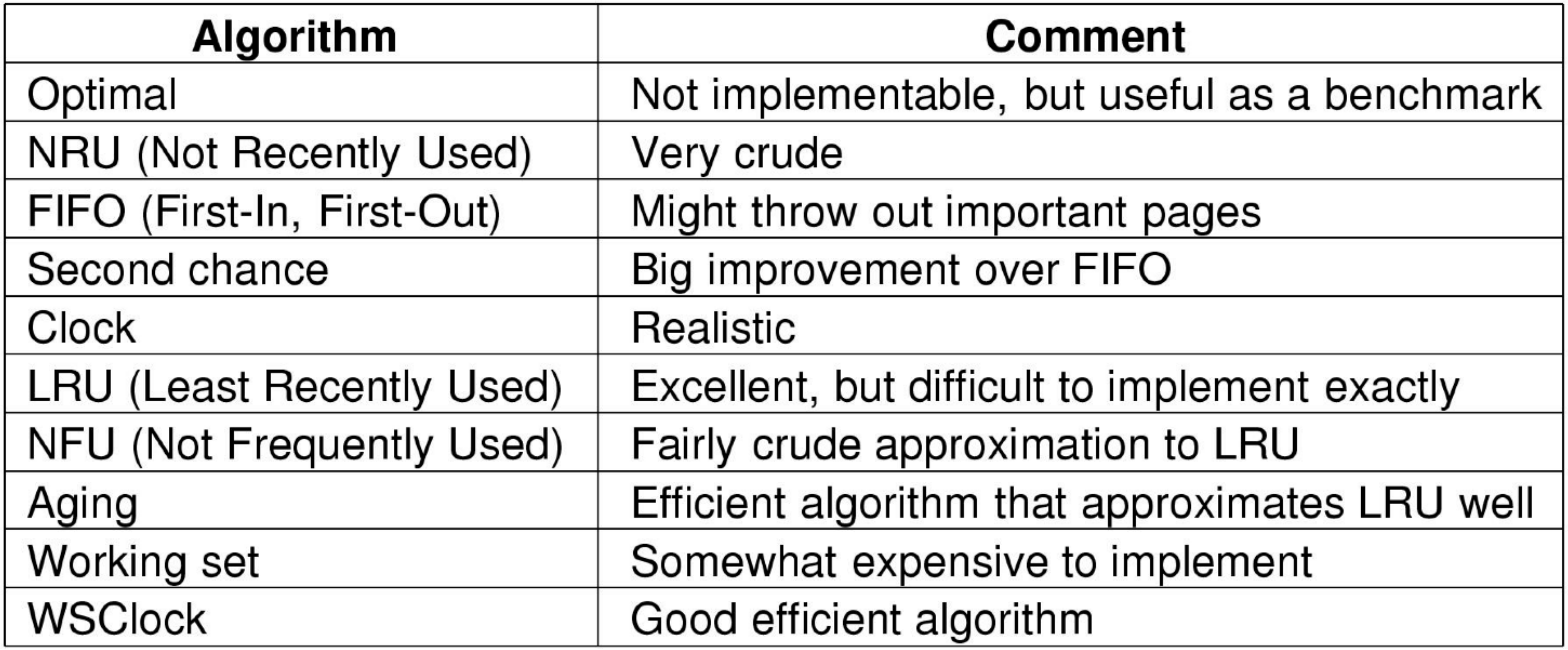
# But which algorithm is best?

# Comparing algorithms through modeling

- **Run a program**
  - Look at all memory references
  - Don't need all this data
  - Look at which pages are accessed
    **0000001222333300114444001123444**
  - Eliminate duplicates
    **012301401234**

- **This defines the Reference String**
  - Use this to evaluate different page replacement algorithms
  - Count page faults given the same reference string

# Summary



| Algorithm | Comment |
|---|---|
| Optimal | Not implementable, but useful as a benchmark |
| NRU (Not Recently Used) | Very crude |
| FIFO (First-In, First-Out) | Might throw out important pages |
| Second chance | Big improvement over FIFO |
| Clock | Realistic |
| LRU (Least Recently Used) | Excellent, but difficult to implement exactly |
| NFU (Not Frequently Used) | Fairly crude approximation to LRU |
| Aging | Efficient algorithm that approximates LRU well |
| Working set | Somewhat expensive to implement |
| WSClock | Good efficient algorithm |

# Local vs. global page replacement

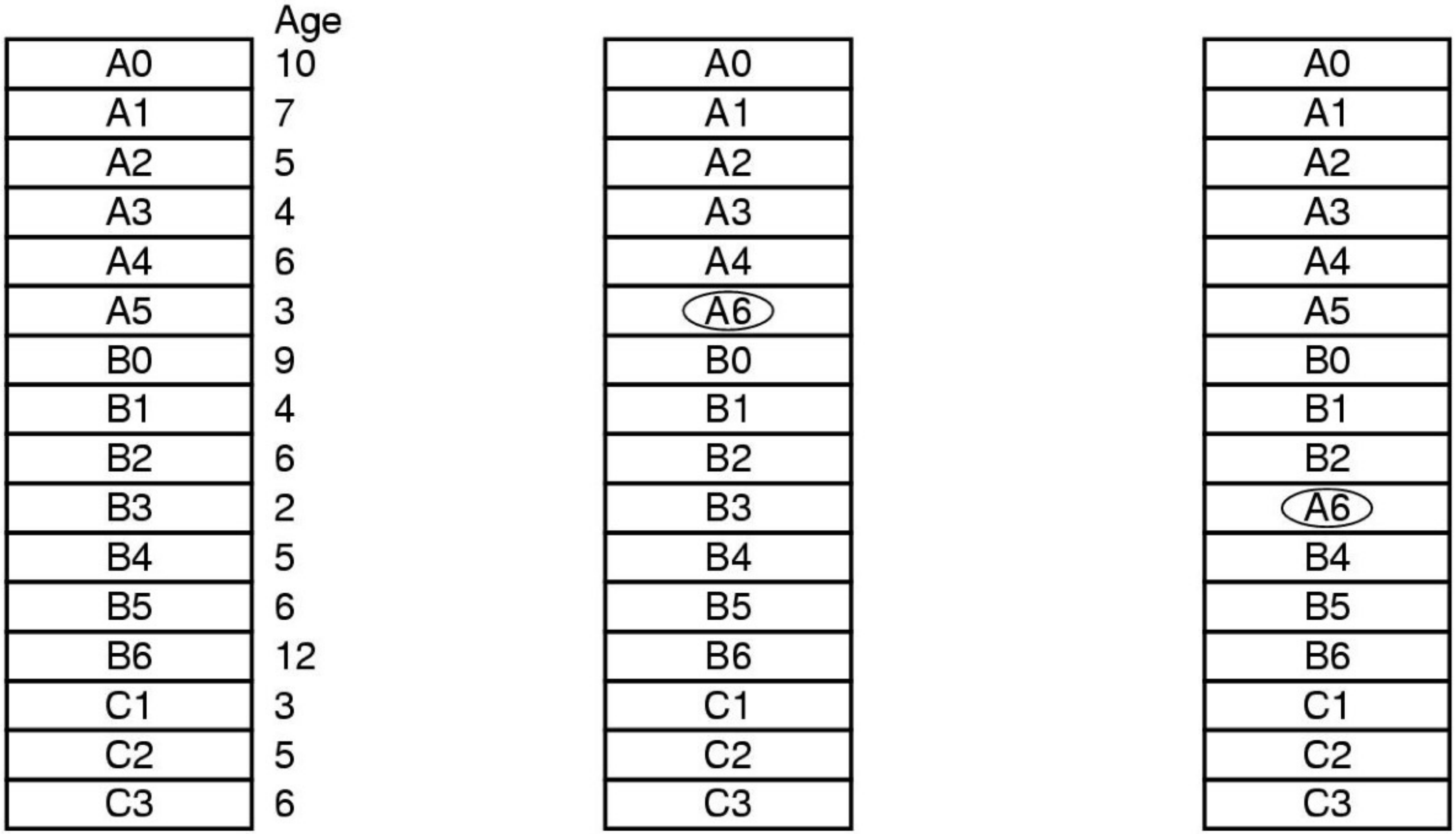
- Assume several processes: A, B, C, ...
- Some process gets a page fault (say, process A)
- Choose a page to replace.

- **Local page replacement**
  - Only choose one of A's pages

- **Global page replacement**
  - Choose any page

# Local vs. global page replacement

- Example: Process has a page fault...



| Original | Age | Local | Global |
|---|---|---|---|
| A0 | 10 | A0 | A0 |
| A1 | 7 | A1 | A1 |
| A2 | 5 | A2 | A2 |
| A3 | 4 | A3 | A3 |
| A4 | 6 | A4 | A4 |
| A5 | 3 | A6 | A5 |
| B0 | 9 | B0 | B0 |
| B1 | 4 | B1 | B1 |
| B2 | 6 | B2 | B2 |
| B3 | 2 | B3 | A6 |
| B4 | 5 | B4 | B4 |
| B5 | 6 | B5 | B5 |
| B6 | 12 | B6 | B6 |
| C1 | 3 | C1 | C1 |
| C2 | 5 | C2 | C2 |
| C3 | 6 | C3 | C3 |

Original

Local

Global

# Local vs. global page replacement

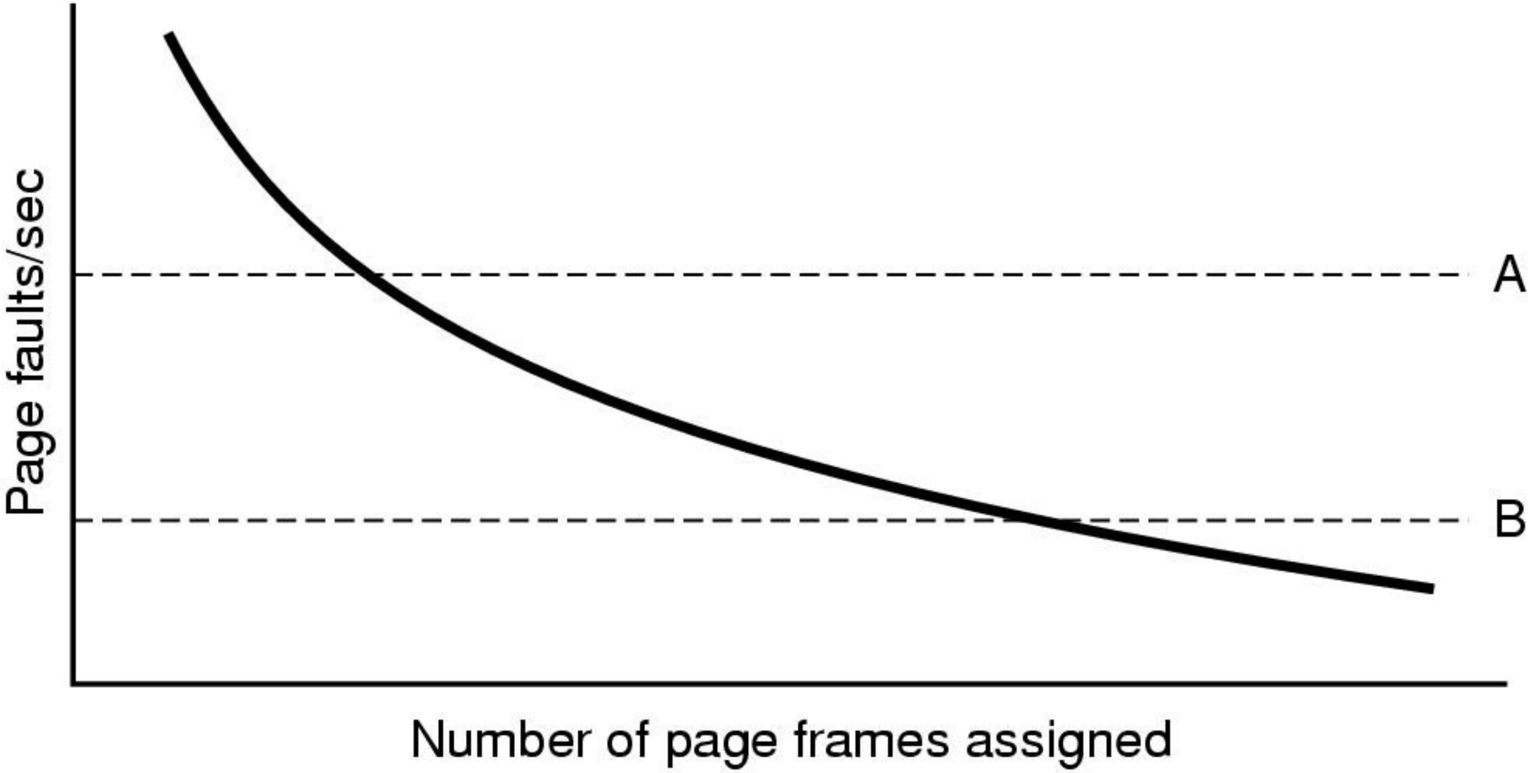
- **Assume we have**
  - 5,000 frames in memory
  - 10 processes
- **Idea: Give each process 500 frames**

- **Fairness?**
  - Small processes: do not need all those pages
  - Large processes: may benefit from even more frames

- **Idea:**
  - Look at the size of each process (… but how?)
  - Give them a pro-rated number of frames
  - With a minimum of (say) 10 frames per process

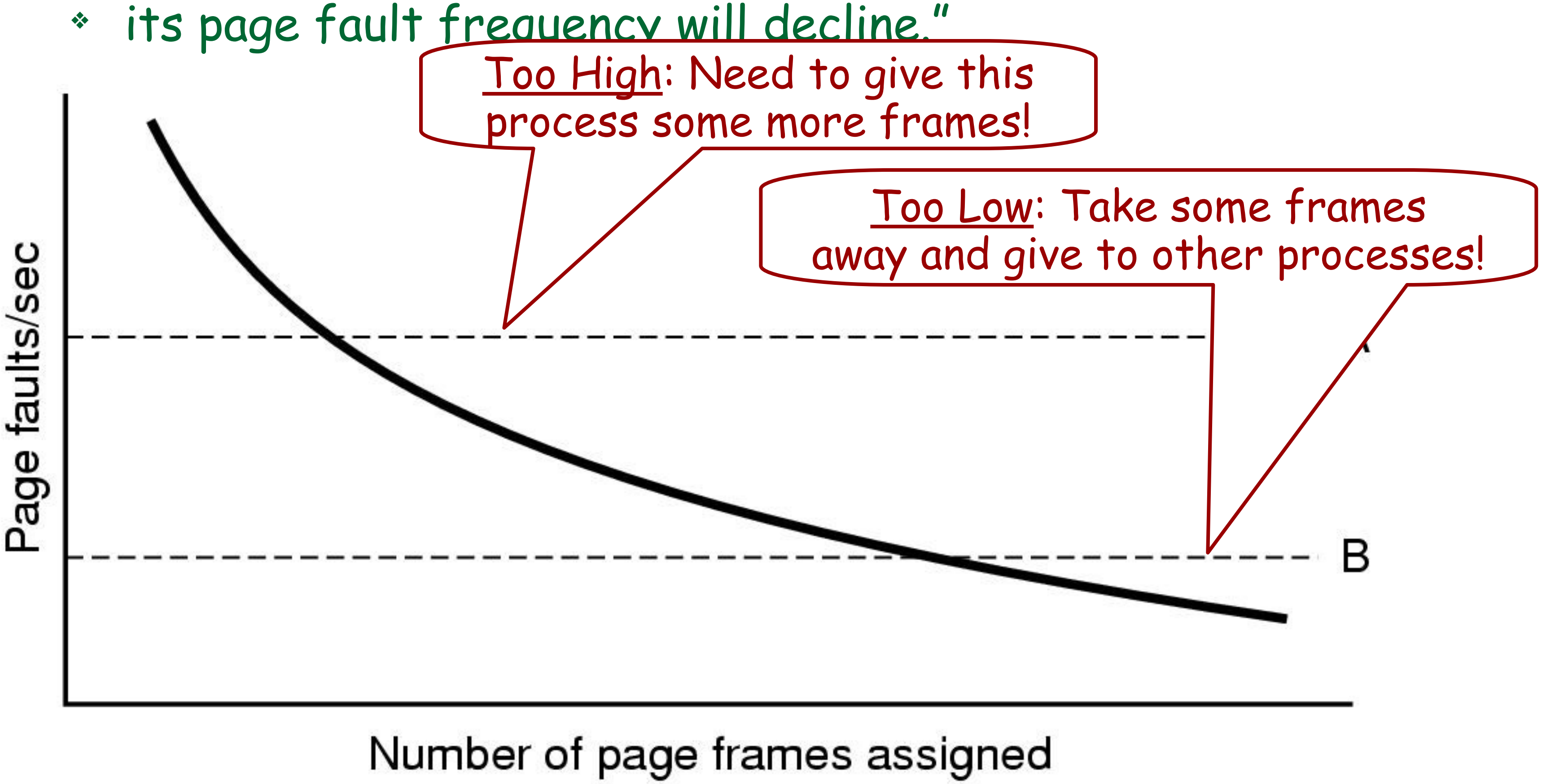
# Page fault frequency

- **“If you give a process more pages,**
  - its page fault frequency will decline.”



# Page fault frequency

- "If you give a process more pages,
  - its page fault frequency will decline."



Too High: Need to give this process some more frames!

Too Low: Take some frames away and give to other processes!

# Page fault frequency

- Measure the page fault frequency of each process.
- Count the number of faults every second.

- May want to consider the past few seconds as well.

# Page fault frequency

- **Measure the page fault frequency of each process.**
- **Count the number of faults every second.**

- **May want to consider the past few seconds as well.**

- **Aging:**
  - Keep a running value.
  - Every second
    - **Count number of page faults**
    - **Divide running value by 2**
    - **Add in the count for this second**

# Load control

- Assume:
  - The best page replacement algorithm
  - Optimal global allocation of page frames

# Load control

- Assume:
  - The best page replacement algorithm
  - Optimal global allocation of page frames

- Thrashing is still possible!

# Load control

- Assume:
  - The best page replacement algorithm
  - Optimal global allocation of page frames
- Thrashing is still possible!
  - Too many page faults!
  - No useful work is getting done!
  - Demand for frames is too great!

# Load Control

- **Assume:**
  - The best page replacement algorithm
  - Optimal global allocation of page frames

- **Thrashing is still possible!**
  - Too many page faults!
  - No useful work is getting done!
  - Demand for frames is too great!

- **Solution:**
  - Get rid of some processes (temporarily).
  - Swap them out.
  - “Two-level scheduling”

# Spare slides

# Belady's anomaly

- **If you have more page frames (i.e., more memory)...**
  - You will have fewer page faults, right???

# Belady's anomaly

- **If you have more page frames (i.e., more memory)...**
  - You will have fewer page faults, right???
- Not always!

# Belady's anomaly

- **If you have more page frames (i.e., more memory)...**
  - You will have fewer page faults, right???

- Not always!

- **Consider FIFO page replacement**
- **Look at this reference string:**

  012301401234

# Belady's anomaly

- **If you have more page frames (i.e., more memory)...**
  - You will have fewer page faults, right???
- Not always!
- **Consider FIFO page replacement**
- **Look at this reference string**

  012301401234
- **Case 1:**
  - 3 frames available --> 9 page faults
- **Case 2:**
  - 4 frames available --> 10 page faults

# Belady's anomaly

All pages frames initially empty

| | | 0 | 1 | 2 | 3 | 0 | 1 | 4 | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Youngest page | | 0 | 1 | 2 | 3 | 0 | 1 | 4 | 4 | 4 | 2 | 3 | 3 |
| | | | 0 | 1 | 2 | 3 | 0 | 1 | 1 | 1 | 4 | 2 | 2 |
| Oldest page | | | | 0 | 1 | 2 | 3 | 0 | 0 | 0 | 1 | 4 | 4 |
| | | P | P | P | P | P | P | P | | | P | P | |

9 Page faults

FIFO with 3 page frames

# Belady's anomaly

All pages frames initially empty

| | | 0 | 1 | 2 | 3 | 0 | 1 | 4 | 0 | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Youngest page | | 0 | 1 | 2 | 3 | 0 | 1 | 4 | 4 | 4 | 2 | 3 | 3 | |
| | | | 0 | 1 | 2 | 3 | 0 | 1 | 1 | 1 | 4 | 2 | 2 | |
| Oldest page | | | | 0 | 1 | 2 | 3 | 0 | 0 | 0 | 1 | 4 | 4 | |
| | | P | P | P | P | P | P | P | | | P | P | | 9 Page faults |

FIFO with 3 page frames

| | | 0 | 1 | 2 | 3 | 0 | 1 | 4 | 0 | 1 | 2 | 3 | 4 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Youngest page | | 0 | 1 | 2 | 3 | 3 | 3 | 4 | 0 | 1 | 2 | 3 | 4 | |
| | | | 0 | 1 | 2 | 2 | 2 | 3 | 4 | 0 | 1 | 2 | 3 | |
| Oldest page | | | | 0 | 1 | 1 | 1 | 2 | 3 | 4 | 0 | 1 | 2 | |
| | | | | | 0 | 0 | 0 | 1 | 2 | 3 | 4 | 0 | 1 | |
| | | P | P | P | P | | | P | P | P | P | P | P | 10 Page faults |

FIFO with 4 page frames

# Belady's anomaly

LRU ???

# Which page size is best?

- Smaller page sizes...
  - Advantages
    - Less internal fragmentation
      - On average: half of the last page is wasted
    - Working set takes less memory
      - Less unused program in memory
  - Disadvantages
    - Page tables are larger
    - Disk-seek time dominates transfer time (It takes the same time to read a large page as a small page)

# Which page size is best?

**Let**

$s$ = size of average process
$e$ = bytes required for each page table entry
$p$ = size of page, in bytes

$s/p$ = Number of pages per process
$es/p$ = Size of page table
$p/2$ = space wasted due to internal fragmentation

overhead = $se/p + p/2$

# Which page size is best?

- Overhead = se/p + p/2
- Want to choose p to minimize overhead.
- Take derivative w.r.t. p and set to zero

  $-se/p^2 + 1/2 = 0$
- Solving for p...

  $p = \mathrm{sqrt}\ (2se)$

# Which page size is best?

s = size of average process

e = bytes required for each page table entry

- **Solving for p...**

  p = sqrt (2se)

**Example:**

**p = sqrt (2 * 1MB * 8) = 4K**

# Which page size is best?

s = size of average process

e = bytes required for each page table entry

- **Solving for p...**

  p = sqrt (2se)

**Example:**

**p = sqrt (2 * 8MB * 4) = 8K**