Replacement Selection

Problem

- Implement the disk-based sorting using the replacement selection
- Replacement selection
  - Refer to the lecture note in week 11
  - Generate initial runs with using heap in the initial pass
  - Then, merge the runs to complete sorting

Problem

- Implement the function **`run_ext_sort`**
  - Input argument
    - `data_size` (integer) – the size of the entire data
    - `page_size` (integer) – the size of one page
    - `num_page` (integer) – the number of pages in the entire buffer
  - Input file
    - `input.bin` (a binary data file integer numbers)
  - Output file
    - `output.bin` (the sorted binary data file)

run_ext_sort.h

- This file should not be modified

```
#pragma once

void run_ext_sort(int data_size, int page_size, int num_pages);
```

Submission

- [student_number].zip to **ETL**
  - E.g., 2000_00000.zip
  - Please include all codes for **`run_ext_sort`**
  - **`run_ext_sort.h`** should not be modified
  - TA will test your **`run_ext_sort`** function using **`run_ext_sort.h`**

Caution

- Write the comments so that TA can understand you codes
- The filenames should be in lower case
- The filenames in `#include` statements should also be in lower case
- Do not use memory more than the entire buffer size to store the data
- Other information, however, could be store regardless of the buffer size
  - The sizes of runs, etc
- There are no restrictions other than those mentioned above

Caution

- Unlike the previous assignment, the codes should work for any number of page
  - When the size of the entire buffer is B pages, your codes should perform $(B-1)$-way merging
    - $(B-1)$ input buffers
    - 1 output buffer
  - A simple algorithm can be used to finding the minimum from the input buffers

An Example of 3-way Merging

- When the size of the entire buffer is 4 pages
- The size of one page is 2