External Merge Sort

Which variant is implemented by demo code??

Simple 2 way merge sort algorithm.

- Pass 0: Produce runs that are one page long.
  - Read each page into memory, sort it, write it out.

- Merge Pair of runs to produce longer runs until only one run is left.
  - While the number of runs at end of previous pass is > 1.
    - While there are runs to be merged from previous pass.
    - Choose next two runs(from previous pass)
    - Merge the runs and write to the output buffer.
    - Force output buffer to disk one page at a time.

How file on a disk is represented in the code??

- Refer files: DiskFile.h, DiskFile.cpp, Common.h.
- Class DiskFile
  - Vector<Page> data
  - Int totalPages
  - Int size

  - Parameterized constructor:

```
DiskFile(int s){
        size = 0;
        totalPages = s;
        data.resize(totalPages);
}
```

- Member Functions of DiskFile
  - void readDiskFile()
    - Read input from cin and push pages to Diskfile. Update totalPages and size.
  - void writeDiskFile():
    - Write DiskFile to cout : pagewise.
  - void DiskFileCopy(DiskFile &f, int startPage, int endPage)
    - Copy contents of f to dest. DiskFile on [startPage to endPage]

How a Disk Page is represented in code??

- Refer files: Page.cpp, Page.h
- class **Page**{
- public:
- vector<int> arr
- int validEntries
- //initializes an empty page with invalid entries i.e. -1
- Page(){
- arr.resize(DISK_PAGE_SIZE, -1);
- validEntries = 0;
- }
- void writePage(): writes valid entries of a page to cout. Invalid entries are represented by -1
- void fillPage(vector<int> &v): Fills page with contents of vector v.
- };

Main Memory??

- Refer Files: MainMemory.h, MainMemory.cpp
- Class MainMemory
  - vector<Frame> data
  - int totalFrames
  - vector<bool> valid
  - Parameterized constructor

```
MainMemory(int s){
        totalFrames = s;
        data.resize(s);
        valid.resize(s);
        for(int i = 0; i < s; i++)
                valid[i] = false;
        cout << "Mem created" << endl;
}
```

- Member functions:
- int loadPage(DiskFile &f, int n) : loads nth page of disk file f to an empty frame if available. Returns -1 if no unallocated frame is available.
- int getEmptyFrame(): Checks if an unallocated frame is available and returns its number if so else -1.
- int getValidEntries(int f): returns number of valid entries in frame f.
- int getVal(int f, int i): returns value stored at location i in frame f.
- void setVal(int f, int i, int val):assigns value val to ith location of frame f.
- void writeFrame(DiskFile &inputFile, int f, int p): write Frame f to file at page p
- void freeFrame(int f): Unallocates frame f.

Frame??

- Refer files: Frame.h, Frame.cpp
- class Frame{
- public:
- vector<int> arr;
- int validEntries;
- //initializes an empty page with invalid entries i.e. -1
- Frame()
- {
- arr.resize(MEM_FRAME_SIZE, -1);
- validEntries = 0;
- }
- void fillFrame(vector<int> &v): fills frame with data from vector v
- void printFrame(): Prints all valid entries of a frame to cout

External Merge Sort

- Refer Files: ExtMergeSort.h, ExtMergeSort.cpp
- Class ExtMergeSort
- int runSize; // size of run in terms of number of pages
- int totalPass; // number of passes performed
- int totalRuns;

- ExtMergeSort(){
- runSize = 0;
- totalPass = 0;
- totalRuns = -1;
- }

- void firstPass(DiskFile &inputFile, MainMemory &memory);
- void sortFrame(MainMemory &memory, int f);
- void merge(DiskFile &inputFile, MainMemory &memory, int leftStart, int mid, int rightEnd);
- void twoWaySort(DiskFile &inputFile, MainMemory &memory);

void twoWaySort(DiskFile &inputFile, MainMemory &memory)

- Checks if main memory has atleast 3 frames.
- Calls firstPass(inputFile, memory)
- Till the runsize is not equal to totalPages:
  - Merge next 2 runs at a time.
  - merge(inputFile, memory, leftStart, mid, rightEnd);

void firstPass(DiskFile &inputFile, MainMemory &memory)

- For each Page in inputFile to mainMemory
  - Frame = loadPage(inputFile, pagenum)
  - sortFrame(memory, frame)
  - writeFrame(inputFile, frame, pagenum)
  - freeFrame(frame)
  - runSize = 1
  - totalPass = 1
  - totalRuns = totalPages

merge(DiskFile &inputFile, MainMemory &memory, int leftStart, int mid, int rightEnd)

- Merges two runs : [leftStart, mid], [mid+1, rightEnd]
- Create a temp DiskFile of FinalRunSize(after merging)
- While either of the runs have pages left
  - Load 1 page from each run to frames
  - Get one empty frame to store merge result
  - When result frame gets full write it to temp DiskFile
- Copy temp DiskFile to original DiskFile

Common.h

- DISK_PAGE_SIZE – update this if you want to change the number of integers that can be stored in 1 page of disk file. Default page size is 3 i.e. 3 integers can be stored in 1 page.
- MEM_FRAME_SIZE- update this if you want to change the number of integers that can be stored in 1 frame of main memory. Default frame size is 3 i.e. 3 integers can be stored in 1 page.
- Note: To run program correctly :
  DISK_PAGE_SIZE and MEM_FRAME_SIZE should be set to same value.

Main.cpp

- It reads an integer: total number of frames in mainMemory
- Creates MainMemory
- Create DiskFile
- Perform Sorting
- Write DiskFile to cout