Các thành viên nhóm

Họ và tên	MSSV	Đóng góp - Mức độ hoàn thành
Nguyễn Hữu Thương	21120393	100%
Nguyễn Hoàng Quân	21120403	100%
Lê Phan Xuân Dũng	21120435	100%

Link github của đồ án:
https://github.com/thuongthuongthuong/sortbigfile_book.git

Giới thiệu bài toán:

- Sắp xếp file có kích thước lớn (khoảng 2.8 GB) theo "The ID of the book" từ nhỏ đến lớn. Trong trường hợp 2 rating có "book id" bằng nhau thì chúng ta sắp xếp chúng ngẫu nhiên.
- Chúng ta sắp xếp file này bằng cách áp dụng thuật toán Merge Sort để split file gốc lớn thành nhiều file nhỏ theo chunk_size đã xác định và sắp xếp các file split này, sau khi sắp xếp tất cả các file split chúng ta merge sort các file nhỏ này thành 1 file lớn.
- Kết quả sau khi sắp xếp thành công chúng ta tạo 1 file .csv có tên là "sorted_books_rating.csv"

Mô tả kiến trúc và thuật toán được áp dụng vào bài toán

1. Đầu tiên chúng ta tạo 1 cấu trúc struct Book gồm các phần tử string ID và string data, sau đó chúng ta tạo 1 operator để so sánh giá trị ID trả về để tìm Min ID dùng cho sắp xếp sau này (chúng ta đặt thêm điều kiện để khi truyền vào giá trị ID cuối cùng của 1 file sẽ trả về đó là giá trị Max ID)

2. Hàm createsplitfile(string inputfile, int sizechuck) (hàm tách file theo chunk_size đã cho) với kiểu dữ liệu là integer để truyền về numberfile tách từ file gốc.

- Tạo 1 biến int numberfile = 0 để đếm xem có bao nhiêu file đã được tách ra với sizechuck truyền vào.
- Mở file cần tách theo string inputfile đã truyền vào. Tạo 1 vector<Book>str để lưu số quyển sách trong 1 chunk, đồng thời chúng ta tách dòng đầu của file gốc (các trường dữ liệu ID, Name, …)
- Tạo 1 vòng while(1) với mỗi lần lặp chúng ta reset vector<Book>str, sau đó kiểm tra xem có phải cuối file gốc không, nếu không thì cúng ta tiến hành lấy dữ liệu và lưu vào vector<Book>str. Hoặc break; nếu đã đủ số lượng book trong sizechuck hoặc file đã đọc xong.
- Chúng ta tiến hành sort chunk book theo kích thước vector sort(str.begin(), str.end()).
- Chúng ta tạo 1 biến string s để tạo tên file dùng để lưu.
- Mở và ghi dữ liệu vào file output fout.open(s, ios_base::out | ios_base::binary).
- Sau khi ghi dữ liệu xong chúng ta đóng file lại và cộng thêm vào numberfile++ đã tạo.
- Nếu hết dữ liệu thì chúng ta kết thúc vòng lặp while.
- Đóng file input và truyền về giá trị numberfile đã tách về.

3. Hàm fineminbook(vector<Book>tmp) để tìm ID nhỏ nhất trong vector book

- Tạo 1 biến Book min = tmp[0]; để lưu giá trị ban đầu, 1 biến index để xác định vị trí của của giá trị ID nhỏ nhất.
- Chạy vòng for để tìm giá trị ID nhỏ nhất có trong vector<Book>tmp truyền vào.
- Trả về vị trí của min_book tìm được.

4. Hàm mergefile(string outputfilename, int sizechunk, int numberfile) để gộp file đã chia thành 1 file dựa vào thuật toán Merge sort.

- Đầu tiên chúng ta mở file output: out.open(outputfilename, ios_base::out | ios_base::binary). Sau đó chúng ta ghi dòng đầu gồm các tiêu đề trường dữ liệu (ID, Name, …)
- Tạo 1 mảng gồm numberfile để lưu.
- Chạy vòng for với số lần chạy numberfile để tạo tên file và mở file để đọc với tên file đã tách tương ứng.
- Tạo 1 vector<Book>minbook.
- Chạy vòng for với số lần chạy numberfile ứng với mỗi file book đã tách lấy ra 1 phần tử book để xét nhỏ nhất để thêm vào file output.
- Tạo 1 biến int count = 0 để đếm xem số file đã tách nhỏ đã hoàn thành sau việc lấy dữ liệu.
- Chạy vòng lặp đếm khi đủ số file đã tách so với với số file đã hoàn thành while (count != numberfile).
  + Tìm vị trí của giá trị ID nhỏ nhất int indexminbook = fineminbook(minbook), và ghi dữ liệu vào file output.
  + Nếu file input còn dữ liệu thì chúng ta lấy dữ liệu của book tiếp theo từ file đã tách thứ indexminbook. Đổi con trỏ tới phần tiếp theo để thêm vào vector
  + Hoặc nếu dữ liệu của split_file đã hết thì thế vào vị trí của file đã đọc hết bằng 1 book có ID rỗng " ".
- Chạy dòng for để đóng toàn bộ luồng đọc file input readsplitfile[i].close();.
- Đóng file ouput và tiến hành xóa toàn bộ file_split tạm bằng cách chạy vòng for và remove(namefilechunk.c_str()).

5. Hàm sortbigfile(string namefileinput, string namefileoutput, int sizechunk) để gọi 2 hàm thực thi trên để sắp xếp bài toán

- Tạo biến numberfile = createsplitfile(namefileinput, sizechunk) để lưu số file đã tách.
- Gọi hàm mergefile(namefileoutput, sizechunk, numberfile) để gộp các file đã tách vào "sorted_books_rating.csv" và xóa các file đã tách ở trên.

6. Hàm main()

- Tạo 1 string inputfilename với giá trị là tên file chúng ta cần sort.
- Tạo 1 string outputfilename với giá trị là tên file "sorted_books_rating.csv" đã sort thành công.
- Tạo 1 biến int chunksize = 50000 với giá trị là số book tối đa trong 1 file tách.
- Gọi hàm thực thi sortbigfile(inputfilename, outputfilename, chunksize).

Độ phức tạp thuật toán (áp dụng thuật toán Merge sort) :

1. Tạo split file với chunk_size là số lượng book trong 1 split file
   - Đọc và thêm (push_back) vào vector: O(chunk_size + 1)
   - Sort vector book: O(chunk_size * log(chunk_size))
     ⇨ Độ phức tạp của 1. : O((chunk_size+1) + chunk_size * log(chunk_size))
2. Merge split file:

B1: Khởi tạo 1 vector minbook đầu tiên: O(number_file) (number_file: số lượng file book đã split)

B2: Tìm min trong vector minbook: O(number_file)

B3: Ghi min vào output file: O(1)

B4: Đổi minbook với book tiếp theo trong split_file chứa minbook: O(1)

B5: Lặp lại B2 , B3, B4 cho đến khi các split file hết dữ liệu

⇨ Độ phức tạp của 2. : O(number_file + (number_file + 2) * size_chunk * number_file)

⇨ Độ phức tạp của thuật toán: O(number_file + (number_file + 2) * size_chunk * number_file) + O((chunk_size+1) + chunk_size * log(chunk_size))