Giả sử HĐH cần hỗ trợ các người dùng làm việc với 01 file nhị phân chứa 02 dãy phần tử. Ở cấp độ người làm ra HĐH bạn hãy thiết kế /tổ chức hình thức lưu trữ phù hợp sao cho file này có thể truy xuất được từ nhiều người trên nhiều môi trường khác nhau *(ví dụ như HĐH trên máy ảnh số tạo ra file ảnh mỗi lần chụp, sau đó file ảnh này có thể chép vào máy tính /đưa lên web và được truy xuất /hiển thị bởi các hệ thống khác).*

Biết rằng:
- Dãy thứ nhất có thể có **rất nhiều phần tử** và t**hường xuyên** có nhu cầu **thêm /xóa** phần tử.
- Dãy thứ hai có số phần tử ít hơn hẳn và thỉnh thoảng cũng có nhu cầu thêm /xóa phần tử.
- Loại tập tin này sẽ được **dùng nhiều và lâu dài**.

QUY ĐỊNH:

*1) Làm theo nhóm **2-3 SV**.*
*2) Bài nộp là 1 file nén (có tên là **<Mã SV>.zip**) chứa file báo cáo <Mã SV>.PDF và folder **<Mã SV>** chứa mã nguồn. // <Mã SV> là chuỗi MaSV1_MaSV2_MaSV3.*
3) Các bài có lấy nội dung từ đâu đó đưa vào phải có trích dẫn rõ tài liệu được tham khảo.
*4) Trình bày **tối thiểu 5 phương án thiết kế** (ngoại trừ các phương án đã nêu trong ví dụ bên dưới), trong đó mỗi phương án có phân tích các ưu điểm /các cải tiến đã được đưa ra để khắc phục khuyết điểm của phương án trước; và nêu ít nhất 1 hạn chế vẫn còn tồn tại của phương án này.*
Ví dụ :

Phương án thiết kế #0:

0.1 Danh sách các thành phần được lưu trữ

Bao gồm 2 thành phần:

+ Thành phần #1: Các phần tử của dãy thứ nhất.

+ Thành phần #2: Các phần tử của dãy thứ hai.

0.2 Tổ chức lưu trữ trong file:

Các phần tử của dãy thứ nhất	Các phần tử của dãy thứ hai

*Các phần tử của dãy #1 được lưu bắt đầu ngay tại offset 0, giả sử dãy #1 có N1 phần tử và kích thước mỗi phần tử là Se1 (byte), khi này kích thước dãy #1 là S1=N1*Se1 => Offset bắt đầu của dãy #2 sẽ là S1.*

(Vd, nếu dãy #1 có 1000000 phần tử, mỗi phần tử có kích thước 888 byte thì sẽ chiếm S1=888000000 byte đầu; và dãy #2 sẽ bắt đầu tại offset S1=888000000 (và nếu dãy #2 có 1000 phần tử, mỗi phần tử 666 byte thì sẽ chiếm 666000 byte kể từ offset S1).

0.3 Cách thức thực hiện các chức năng chính

+ Thêm 1 phần tử vào dãy #1: thực hiện thêm vào cuối dãy #1 bằng cách

- *đọc tất cả các phần tử trên dãy #2 vào bộ nhớ,*
- *nhảy đến offset S1 ghi nội dung phần tử mới,*
- *sau đó ghi vào file toàn bộ nội dung dãy #2.*

+ Thêm 1 phần tử vào dãy #2: thêm vào cuối dãy #2 bằng cách nhảy đến cuối dãy và ghi nội dung phần tử.

+ Xóa 1 phần tử trên dãy #1:

- *đọc nội dung phần tử cuối dãy #1 (tại offset S1-Se1)*
- *đọc tất cả các phần tử trên dãy #2 vào bộ nhớ,*
- *nhảy đến vị trí của phần tử muốn xóa ghi nội dung phần tử cuối này vào (để không phải dồn hàng loạt phần tử trên dãy #1),*
- *nhảy đến offset S1-Se1 ghi vào file toàn bộ nội dung dãy #2.*

+ Xóa 1 phần tử trên dãy #2:

- *đọc nội dung phần tử cuối dãy #2 (tại offset S1+S2-Se2, trong đó S2 là kích thước dãy #2 và Se2 là kích thước mỗi phần tử trên dãy #2),*
- *nhảy đến vị trí của phần tử muốn xóa ghi nội dung phần tử cuối này vào (để không phải dồn hàng loạt phần tử trên dãy #2).*

0.4 Ưu khuyết điểm của thiết kế này

☺ Xóa phần tử bằng hình thức lấy nội dung phần tử cuối ghi đè lên thì không phải dồn dãy nên thời gian thực hiện sẽ nhanh hơn và độ an toàn cao hơn.
☺ Thêm phần tử vào cuối dãy thì không phải dồn nên cũng nhanh hơn và an toàn hơn
☹ Việc thêm /xóa trên dãy #1 thì tuy không phải dồn trên dãy #1 nhưng vẫn phải dồn trên dãy #2
☹ Do chỉ lưu nội dung các phần tử chứ không lưu trữ số phần tử nên không xác định được S1 và S2, và trước mắt là vì vậy KHÔNG THỂ TRUY XUẤT ĐƯỢC NỘI DUNG DÃY #2. ***DO ĐÓ THIẾT KẾ NÀY KHÔNG THỂ SỬ DỤNG ĐƯỢC.***

Phương án thiết kế #1:

1.1 Danh sách các thành phần được lưu trữ

Bao gồm 04 thành phần:

+ Thành phần #1: Số nguyên không dấu 4 byte lưu số phần tử của dãy thứ nhất (N1).

+ Thành phần #2: Nội dung các phần tử của dãy thứ nhất (mỗi phần tử có kích thước Se1).

+ Thành phần #3: Số nguyên không dấu 2 byte lưu số phần tử của dãy thứ hai (N2).

+ Thành phần #4: Nội dung các phần tử của dãy thứ hai (mỗi phần tử có kích thước Se2).

***1.2 Tổ chức lưu trữ trong file**:*

** Trật tự các thành phần được phân bố như sau:*

Số phần tử dãy #1	Nội dung các phần tử của dãy #1	Số phần tử dãy #2	Nội dung các phần tử của dãy #2
Off 0	*Off 4*	*Off S1+4*	*Off S1+6*

*(Các phần tử của dãy #1 được lưu bắt đầu ngay tại offset 4, kích thước dãy #1 là S1=N1*Se1 => Offset bắt đầu của dãy #2 sẽ là S1+4)*

** Vị trí của thành phần trên file:*

+ Thành phần #1: offset 0.

+ Thành phần #2: offset 4.

+ Thành phần #3: offset S1+4.

+ Thành phần #4: offset S1+6.

(Vd, nếu dãy #1 có N1=1000000, Se1=888 thì S1=888000000 byte; và dãy #2 sẽ bắt đầu tại offset S1+6=888000006; nếu dãy #2 có N2=1000, Se2=666 byte thì S2=666000 byte).

1.3 Cách thức thực hiện các chức năng chính

+ Thêm 1 phần tử vào dãy #1: thực hiện thêm vào cuối dãy #1 bằng cách

- *đọc tất cả các phần tử trên dãy #2 vào bộ nhớ (do S2 không lớn nên có thể chứa trong bộ nhớ),*
- *nhảy đến offset S1+4 ghi nội dung phần tử mới,*
- *ghi vào file số 2 byte N2 vào file (lúc này con trỏ file đang ở offset S1+4+Se1*
- *ghi tiếp toàn bộ nội dung dãy #2 vào file*
- *tăng N1 thêm 1 và ghi N1 (4 byte) vào offset 0.*

+ Thêm 1 phần tử vào dãy #2: : thêm vào cuối dãy #2 bằng cách

- *nhảy đến cuối dãy (offset S1+6+S2) và ghi nội dung phần tử mới; ;*
- *tăng N2 thêm 1 và ghi N2 (2 byte) vào offset S1+4.*

+ Xóa 1 phần tử trên dãy #1:

- *đọc nội dung phần tử cuối dãy #1 (tại offset S1+4-Se1)*
- *đọc tất cả các phần tử trên dãy #2 vào bộ nhớ,*
- *nhảy đến vị trí của phần tử muốn xóa ghi nội dung phần tử cuối này vào (để không phải dồn hàng loạt phần tử trên dãy #1),*
- *nhảy đến offset S1-Se1 ghi vào file số 2 byte N2 và toàn bộ nội dung dãy #2;*
- *giảm N1 đi 1 và ghi N1 (4 byte) vào offset 0.*

+ Xóa 1 phần tử trên dãy #2:

- *đọc nội dung phần tử cuối dãy #2 (tại offset S1+6+S2-Se2,*
- *nhảy đến vị trí của phần tử muốn xóa ghi nội dung phần tử cuối này vào (để không phải dồn hàng loạt phần tử trên dãy #2).*
- *giảm N2 đi 1 và ghi N2 (2 byte) vào offset S1+4.*

1.4 Ưu khuyết điểm của thiết kế này

☺ Xóa phần tử bằng hình thức lấy nội dung phần tử cuối ghi đè lên thì không phải dồn dãy nên thời gian thực hiện sẽ nhanh hơn và độ an toàn cao hơn (việc dồn dãy sẽ khiến phải thực hiện nhiều thao tác ghi trên file, mà thao tác ghi file nếu thực hiện quá nhiều lần sẽ dễ dẫn đến hư file).
☺ Thêm phần tử vào cuối dãy thì không phải dồn nên cũng nhanh hơn và an toàn hơn.
☹ Việc thêm /xóa trên dãy #1 thì tuy không phải dồn trên dãy #1 nhưng vẫn phải dồn trên dãy #2. ***Dãy #1 lại hay có thao tác thêm xóa*** *nên việc dồn này sẽ thường xuyên xảy ra khiến cho dữ liệu* ***dãy #2 sẽ dễ hư hỏng.***
☹ Nếu giá trị N1 bị sai thì không chỉ khiến nội dung dãy #1 bị lỗi mà ***toàn bộ dãy #2 sẽ bị sai.***

Phương án thiết kế #2:

2.1 Danh sách các thành phần được lưu trữ

Bao gồm 04 thành phần:

+ Thành phần #1: Số nguyên không dấu 2 byte lưu số phần tử của dãy thứ hai (N2).

+ Thành phần #2: Nội dung các phần tử của dãy thứ hai (mỗi phần tử có kích thước Se2).

+ Thành phần #3: Số nguyên không dấu 4 byte lưu số phần tử của dãy thứ nhất (N1).

+ Thành phần #4: Nội dung các phần tử của dãy thứ nhất (mỗi phần tử có kích thước Se1).

2.2 Tổ chức lưu trữ trong file:

** Trật tự các thành phần được phân bố như sau:*

Số phần tử dãy #2	**Nội dung các phần tử của dãy #2**	**Số phần tử dãy #1**	**Nội dung các phần tử của dãy #1**
Off 0	*Off 2*	*Off S2+2*	*Off S2+6*

*(Các phần tử của dãy #2 được lưu bắt đầu tại offset 2, kích thước dãy #2 là S2=N2*Se2 => Offset bắt đầu của dãy #1 sẽ là S2+2)*

** Bảng tham số các thành phần trên file:*

Offset	*Length(B)*	*Type*	*Ý nghĩa*
0	**2**	**Uint16**	N2 – số nguyên ko dấu 16bit chứa số phần tử của dãy #2
2	*Se2*	*Type2*	Phần tử 0 của dãy #2 *(phần tử đầu)*
2 + Se2	*Se2*	*Type2*	Phần tử 1 của dãy #2
2 + 2Se2	*Se2*	*Type2*	Phần tử 2 của dãy #2
…	*Se2*	*Type2*	
2 + (N2-1)*2Se2	*Se2*	*Type2*	Phần tử (N2-1) của dãy #2 *(phần tử cuối)*
2 + S2	**4**	**Uint32**	N1 – số nguyên ko dấu 32bit chứa số phần tử của dãy #1
S2 + 6	*Se1*	*Type1*	Phần tử 0 của dãy #1 *(phần tử đầu)*
S2+6 + Se1	*Se1*	*Type1*	Phần tử 1 của dãy #1
S2+6 + 2Se1	*Se1*	*Type1*	Phần tử 2 của dãy #1
…	*Se1*	*Type1*	
S2+6 + S1-Se1	*Se1*	*Type1*	Phần tử (N1-1) của dãy #1 *(phần tử cuối)*

2.3 Cách thức thực hiện các chức năng chính

+ Thêm 1 phần tử vào dãy #1: : thêm vào cuối dãy #1 bằng cách

- *nhảy đến cuối dãy (offset S1+S2+6) và ghi nội dung phần tử mới; ;*
- *tăng N1 thêm 1 và ghi N1 (4 byte) vào offset S2+2.*

+ Thêm 1 phần tử vào dãy #2: nếu làm trên nội bộ file thì chắc chắn là phải dồn dãy #1 nằm ở phía sau, nhưng do dãy #1 có kích thước lớn nên việc dồn này là rất bất ổn (mất rất nhiều thời gian và dễ gây ra hư hỏng), vì vậy giải pháp là tạo ra 01 file mới và chép dữ liệu từ file cũ qua. Thuật toán tương ứng như sau:

- *mở file dữ liệu F1 (theo chế độ Read+Binary) và mở file tạm F2 (theo chế độ Write+Binary)*
- *đọc số nguyên 2 byte N2 từ file F1*
- *ghi số nguyên 2 byte (N2+1) vào file F2*
- *lặp N2 lần các việc {đọc phần tử có kiểu Type2 (kích thước Se2) từ file F1; ghi phần tử vừa đọc vào file F2}*
- *ghi phần tử muốn thêm vào file F2*
- *đọc số nguyên 4 byte N1 từ file F1*
- *ghi số nguyên 4 byte N1 vào file F2*
- *lặp N1 lần các việc {đọc phần tử có kiểu Type1 (kích thước Se1) từ file F1; ghi phần tử vừa đọc vào file F2}*
- *đóng 2 file F1 và F2*
- *đổi tên mở rộng của file F1 sang "Bak" // thay vì xóa đi thì giữ lại như là một hình thức backup*
- *đổi tên file F2 thành tên của file F1 ban đầu*

+ <u>Xóa 1 phần tử trên dãy #1:</u>

- *đọc nội dung phần tử cuối dãy #1 (tại offset S1+S2+6-Se1)*
- *nhảy đến vị trí của phần tử muốn xóa ghi nội dung phần tử cuối này vào (để không phải dồn hàng loạt phần tử trên dãy #1),*
- *giảm N1 đi 1 và ghi N1 (4 byte) vào offset S2+6.*

+ <u>Xóa 1 phần tử trên dãy #2</u>: nếu làm trên nội bộ file thì chắc chắn là phải dồn dãy #1 nằm ở phía sau và đó là điều rất không ổn như đã phân tích ở thao tác thêm vào dãy #2, và giải pháp cũng sẽ là tạo ra 01 file với thuật toán tương ứng như sau:

- *mở file dữ liệu F1 (theo chế độ Read+Binary) và mở file tạm F2 (theo chế độ Write+Binary)*
- *đọc số nguyên 2 byte N2 từ file F1*
- *ghi số nguyên 2 byte (N2-1) vào file F2*
- *lặp N2 lần các việc {đọc phần tử có kiểu Type2 (kích thước Se2) từ file F1; nếu phần tử đọc được không phải là phần tử cần xóa thì ghi nó vào file F2}*
- *đọc số nguyên 4 byte N1 từ file F1*
- *ghi số nguyên 4 byte N1 vào file F2*
- *lặp N1 lần các việc {đọc phần tử có kiểu Type1 (kích thước Se1) từ file F1; ghi phần tử vừa đọc vào file F2}*
- *đóng 2 file F1 và F2*
- *đổi tên mở rộng của file F1 sang "Bak" // thay vì xóa đi thì giữ lại như là một hình thức backup*
- *đổi tên file F2 thành tên của file F1 ban đầu*

1.4 Ưu khuyết điểm của thiết kế này

*😊 Thiết kế trước khiến cho dữ liệu **dãy #2 dễ hư hỏng** vì liên tục bị dồn (ghi đè toàn bộ phần tử) do việc thêm /xóa trên dãy #1 được thực hiện thường xuyên. Thiết kế này **đã khắc phục được vấn đề** nghiêm trọng đó bằng cách đưa dãy #2 lên đầu file và dãy #1 xuống cuối.*

😊 Thiết kế này cũng đã loại bỏ luôn thao tác dồn khi thêm /xóa trên dãy phía trước – tuy giải pháp tạo ra file mới cũng gây tốn chi phí (thời gian) nhưng độ an toàn cao hơn – thậm chí còn có thêm một bản backup!

☹ Như vừa phân tích: việc chép 1 file dữ liệu lớn ra 1 file khác sẽ gây tốn kém thời gian.

*☹ Thiết kế này vẫn có hạn chế chưa khắc phục là khi giá trị N2 bị sai thì không chỉ khiến nội dung dãy #2 bị lỗi mà **toàn bộ dãy #1 sẽ bị sai.***