Como pensar los problemas de Algoritmos

Estructuras de datos: Características

	Vector	Archivo	Listas ordenadas
Almacenamiento	logico	fisico	Logico
Procesamiento	rapido	Lento	Rapido
Tamaño en T.E.	Fijo	Variable	Variable
Tamaño cada posicion	Informacion	Informacion	Info + sgte
Busqueda directa	SI→V[N]	fseek	Buscar secuencialmente
Persistencia	NO	SI	NO
Busqueda binaria	SI	SI	NO→arbol
Busqueda secuencial	SI	NO se recomienda	SI→unica posible
Carga directa	SI →V[N] = valor	Acceder → fseek Grabar → frwrte	Utilizando diferentes funciones
Carga secuencial	SI→v[ultimaPos] = valor	Acces. a ultima pos grabar	SI Insertar ordenado
Carga sin repetir	Buscar No esta agregar	NO se recomienda Est. Auxiliares	Buscar No esta agregar
Ordenamiento	PUP Metodos de ordenam.	PUP Solo usando est. aux	Genera insertarOrdenado
Definicion	Td Nombre[tamaño]	FILE*f; f= fopen(.....)	Definir Nodo → struct Punt Nodo* X = NULL
Acceso	V[N]	Acceder → fseek leerr → fread	Buscar secuencialmente
Corte de control	SI	SI	SI
Apareo	SI	SI	SI
Modificar la pos N	V[N] = valor	Apuntar → fseek Leer→fread Modificar en memoria Apuntar → fseek Grabar → fwrite	Buscar secuencialmente modificar

Estructuras de datos: Criterios de selección

1. Priorizar de ser posible acceso directo y velocidad de procesamiento.
    a. Vector en primer lugar si se cumple tamaño fijo, razonable y conocido a priori, y sin necesidad de persistencia→ acceso directo, búsqueda binaria, búsqueda secuencial
2. Si no se conoce el tamaño y no se requiere persistencia
    a. Estructuras enlazadas
        i. Pila si se debe invertir el orden o si es irrelevante
        ii. Colas si se debe mantener
        iii. Listas si se debe generar
3. Archivo si se requiere persistencia: directamente o estructuras auxiliares después cargar al archivo

Estructuras → muchos datos del mismo tipo

Archivos
Vectores → memoria → mejorar tiempos de procesamiento → tamaño fijo
Listas ordenadas → memoria →mejorar tiempos de procesamiento → tamaño variable

Observen como vienen los datos:

1. Es posible que los datos de partida estén en la forma en que nos piden los resultados→ procesamos directamente el archivo
2. Es posible que los datos de partida NO estén en la forma en que nos piden los resultados→ Debemos cargar en estructuras auxiliares y procesar esta estructura, reordenar los datos del archivo y procesar el archivo

Datos provienen de archivos

1. Tal como están se muestran

Consigna →Mostrar todos los datos del archivo
No requiere guardarlo en otra estructura, leer cada registro y mostrarlo

Dato Archivo ordenado

Consigna →Mostrar todos los datos del archivo ordenados por el mismo campo
No requiere guardarlo en otra estructura, leer cada registro y mostrarlo
Dado un archivo de registros mostrar su contenido por pantalla
Los resultados están en el orden que están en el archivo **No Necesitan estructura auxiliar**

2. Cambiar el orden, u ordenarlo

Consigna es mostrarlo ordenado, si esta desordenado, o el el campo no se corresponde con el resultado
Se requiere una estructura auxiliar memoria para facilitar el procesamiento
Vector → conozca tamaño a priori
Cargo el archivo al vector
Ordeno el vector
Muestro los datos del vector
Lista ordenada → No conozca tamaño a priori
Cargo el archivo a la lista ordenada
Muestro los datos de la lista

3. Ordenar un archivo desordenado

Dado un archivo de registros desordenado mostrar su contenido por pantalla ordenado por el campo1
Dado un archivo de registros desordenado ordenarlo por el campo1
Dado un archivo de registros desordenado mostrar su contenido por pantalla ordenado por el campo1 y a igualdad el campo 1 por el campo 2

Se requiere una estructura auxiliar memoria para facilitar el procesamiento
Vector → conozca tamaño a priori
Cargo el archivo al vector
Ordeno el vector
Me posiciono en la primera posición del archivo
Grabo los datos del vector → archivo se reemplaza y queda ordenado
Lista ordenada → No conozca tamaño a priori

Cargo el archivo a la lista ordenada
Me posiciono en la primera posición del archivo
Grabo los datos del vector → archivo se reemplaza y queda ordenado

---

Buscar en un archivo

1. Archivo ordenada tiene en la clave de búsqueda una PUP

No requiere estructura auxiliar → **búsqueda directa** Pos = Clave-valor 1ra pos
Si decido usarla puedo → vector No Lista→ solo permite secuencial
Clave 101 … 150 → buscar 135 → fseek(f, (135-101)*sizeof(r), SEEK_SET)

2. Archivo ordenado → la posicion no es predecible a priori

Una solución posible → **Busqueda binaria** en el archivo
Otra solución → cargarlo en estructura auxiliar y buscar en ella

3. Archivo desordenado

Necesariamente debemos cargarlo estructura auxiliar
Vector → si se da la condición del tamaño
Búsqueda secuencial
Ordenar el vector → búsqueda binaria → más eficiente que la secuencial
Cada posición solo necesitamos la información
Lista → si no se conoce el tamaño a priori
Búsqueda secuencial
Cada posición necesitamos la información + referencia al siguiente

Dados dos archivos uno con los datos personales de los alumnos, sin orden, el otro ordenado por legajo y materia con las materias cursadas Desarrollar un programa que muestre por pantalla los resultados de las materias y el nombre del estudiante. El campo que vincula ambos archivos es el numero de legajo

---

Datos están en dos estructuras y las quiero agrupar según un determinado orden

1. Dos archivos ordenados → intercalar conservando el orden

Apareo de los archivos sin necesidad de cargarlos en una estructura auxiliar

2. 1 archivo ordenado y el otro no

El ordenado lo dejo como esta
El desordenado lo cargo en una estructura auxiliar → vector ordenado o una lista ordenada
Una solución →apareo archivo ordenado con la estructura auxiliar
Otra solución → ordenar el archivo desordenado→ aparear los archivos

3. Ambos desordenados

Cargar ambos en una estructura auxiliar
Una solución → aparear las estructura auxiliares
Otra solución → ordenar los archivos → aparearlos
Otra solución → si alcanza → cargar ambos archivos misma estructura auxiliar y ordeno

Dados dos archivos ambos ordenados por numero de legajo, uno con los datos personales de los alumnos, el otro ordenado por legajo y materia con las materias cursadas Desarrollar un programa

que muestre por pantalla los resultados de las materias y el nombre del estudiante ordenado por numero de legajo. El campo que vincula ambos archivos es el numero de legajo

---

Porque y cuando corte de control

1. Datos en un archivo ordenado por dos campos (legajo y materia) con repetición del legajo

Mostrar los datos agrupados por el primer campo → no necesito estructura auxiliar

nl	Mat	nota
4	Algo	1
4	Ssl	2
7	Algo	2
7	Md	10
7	Syo	9

Mostrar los datos agrupados por nl → corte de control

NL 4
Mat nota
Algo 1
Ssl 2

Nl 7
Mat nota
Algo 2
Md 10
Syo 9

2. Archivo esta sin orden→ idéntica consigna

Cargarlo estructura auxiliar, ordenada por dos campos
Una solución→ hacer el corte de control en la estructura auxiliar
Otra solución → Ordenar el archivo, y corte de control en el archivo

Dni	Nl	cm	Nota

archivo

Dni	Nl	mat	nota
6	14	Alg	2
9	25	Sint	4
3	18	Syo	3
2	30	md	1
5	35	alg	3
14	24	md	1

Son 6 registros

```
struct tr{int dni; int nl; char cm[20]; int nota};
tr vector[6];
tr r;
```

vector

Dni	nl	mat	nota

```
FILE* f = fopen("nombre","rb+");
I = 0;
1 fread(&r,sizeof(tr),1,f);
while(!feof(f)){
 vector[i]= r;
i++;
 fread(&r,sizeof(r),1,f);
}

2 while(fread(&r,sizeof(r),1,f)){
 vector[i]= r;
i++;
}

3 for(i=0;i<6; i++){
fread(&r,sizeof(r),1,f);
vector[i]= r;
}

4 fread(&r,sizeof(r),1,f);
for(i=0;!feof(f); i++){
vector[i]= r;
fread(&r,sizeof(r),1,f);
}

5 fread(vector,sizeof(r),6,f);→ 5.1 fread(&vector[0],sizeof(r),6,f);
```

vector

Dni	nl	mat	nota
6	14	Alg	2
9	25	Sint	4
3	18	Syo	3
2	30	md	1
5	35	Alg	3
14	24	md	1

EN ESTE PUNTO VECTOR Y ARCHIVO DESORDENADO
ordenarVector(vector,6);// archivo sin orden, vector ordenado
Mostrar por pantalla sin modificar el archivo

```
for(i=0;i<6; i++)
cout <<vector[i].dni<< vector[i].nl.....;
```

Reordenar el archivo

```
 fseek(f,0,SEEK_SET);
1 for(i=0;i<6; i++){
r = vector[i];
fwrite(&r,sizeof(r),1,f);
}

2 for(i=0;i<6; i++){
fwrite(&vector[i],sizeof(r),1,f);
}

3 fwrite(vector,sizeof(r),6,f);
```

Archivo ordenado

Volvemos al principio archivo sin orden y no se conoce el tamaño
Estructura adecuada lista

info				Sgte NODO*
dni	nl	mat	nota	

```
 struct Nodo{
          tr info;
          Nodo* sgte;
};
Nodo* Lista = NULL;

while(fread(&r,sizeof(r),1,f))
 insertarOrdenado(lista,r);
```

archivo sin orden. Pero la lista ordenada

```
mostrar                  fseek(f,o,SEEK_SET);


while(lista!=NULL){      ==
r = pop(lista);          ==
cout<<r.dni.....          fwrite(&r,sizeof(r),1,f)
}
```

A1 A2

2
8
15

3
4
5

R1

8

R2

5

A3

2
3
4
5
8
15

```
fread(&r1;sizeof(r1),1,A1);
fread(&r2;sizeof(r2),1,A2);
while(!feof(A1)&&!feof(A2)){
if(r1.nl<r2.nl){ fwrite(&r1;sizeof(r2),1,A3); fread(&r1;sizeof(r1),1,A1);}
 else fwrite(&r2;sizeof(r2),1,A3); fread(&r2;sizeof(r2),1,A2);}
}
// terminar el que no se agoto
while(!feof(A1)){
fwrite(&r1;sizeof(r2),1,A3);
fread(&r1;sizeof(r1),1,A1);
};

// terminar el que no se agoto
while(!feof(A2)){
fwrite(&r2;sizeof(r2),1,A3);
fread(&r2;sizeof(r1),1,A2);
}
```