Unidad 09 Patrones algorítmicos

9

Contenidos analíticos

Parte III: Estructura de datos estáticos y almacenamiento lógico

Patrones de carga: en la definición. Secuencial, directa, ordenada
Patrones de recorrido: total, parcial, en ambas direcciones. Con corte de control, apareando dos o mas estructuras
Patrones de búsqueda: directa, binaria, lineal. Análisis de eficiencia
Patrones de ordenamiento: Con posición Única predecible, mediante métodos de ordenamiento. Análisis de eficiencia

Se desarrollan algunos patrones algorítmicos para datos simples, secuencias, estructuras lógicas y físicas.
El desarrollo es conceptual quedando para cada uno de ustedes adaptarlos según las características propias de cada caso.
A los efectos de una auto evaluación se recomienda en caso que sea posible generalizar, según corresponda según tipo de dato o criterio de seleccion
Los patrones de estructuras enlazadas se profundizan cuando abordemos esa unidad

Datos simples	
Dados dos identifiadores intercambiar valores	Dados tres valores mostrarlos ordenados

void intercambio (int &a, int &b)
{
// se asumen valores diferentes
 	
int c = a;
a = b;
b = c;
return;
}
Generalice para escalares
	void ordenar3(int & mr, int md, int mn)
	{
	/* poner en mr el mayor, mn el menor y en md el medio. Los tres valores son diferentes*/
	If (mr<md) intercambio(mr,md);
	If(mr<mn) intercambio(mr, mn);
	If(md<mn) intercambio(md, mn);
	return;
	}
	Generalice para escalares

Dados dos valores retornar el mayor	dados tres valores retornar el mayor

int mayorDeDos(int a, int b)
{
 	
return a>b?a:b;
}
//plantee otra solución
	Int mayorDeTres(int a; int b; int c)
	{
	int aux = mayorDeDos(a,b);
	return mayorDeDos(aux, c);
	}
	//plantee alternativas

buscar el máximo en un conjunto de 10 numeros enteros

Alternativa 1 valores > 0	Alternativa 2 sin restricción	Cual se le ocurre?

int maximo()
{
int v, myr = 0, i;
for(i=0; i<10; i++) {
cin>>v;
if(v>myr) myr= v;
}
return myr;
}
	int máximo()
	{
	Int v, myr, i;
	for(i=0, i<10; i++){
	cin>>v;
	if(i==0||v>myr) myr= v;
	}
	return myr;
	}
		Como imagina, por ejemplo una lectura anticipada asignándole ese primer valor al mayor y luego avanzar con el resto de las lecturas haciendo las evaluaciones? Obtendría el resultado correctamente?

En la propuesta anterior que cambios propone si:
El tipo de dato es float. Se puede generalizar por tipo de dato?
El lote evalua N valores y N es un parámetro
Para buscar un mínimo se puede generalizar por tipo de dato y criterio de selección?
El lote, para cualquiera de las opciones no es conocido a priori
Buscar el máximo o mínimo y la posición relativa del mismo dentro del conjunto de valores
Buscar el máximo o mínimo y el siguiente
Idem indicando posición relativa de ambos
Buscar máximo o mínimo y los dos siguientes
Idem indicando posición relativa de los tres valores
Dado un conjunto de float<> de cero buscar el máximo de los negativos y el mínimo positivo

Piense situaciones problemáticas que requieran utilización de estos patrones de datos simples

Con struct → suponga struct tr { int c1, char c2[10]}

Función que cargue los datos de un registro	Función que muestre una struct

tr crearReg(int a, char s[])
{
tr r;
r.c1 = a;
strcpy(r.c2,s);
return r;
}
	void mostrarReg(tr r)
	{
	cout<<r.c1<<" "<<r.c2<<endl;
	}
	/* la asignación externa de entrada y salida en una struct en miembro a miembro

Estructuras de colección de datos homogéneos

Vector	Archivo	Lista
Carga con asignación interna de estructuras con struct		

void cargar(tr v[], int N)
{
int i=0;
int a;
char s[N];
tr r;
while(i<N && la que sugiera){
cin>>a;
cin >> s;
v[i]=creaeReg(a,s); i++;
}
}
	void cargar(FILE* f)
	{
	int a;
	char s[N];
	tr r;
	while(la que sugiera){
	cin>>a;
	cin >> s;
	fwrite(&r, sizeof(r),1, f);
	}
	}
		void cargar(Nodo* l)
		{
		int a;
		char s[N];
		tr r;
		while(la que sugiera){
		cin>>a;
		cin >> s;
		insertarOrdenado(l,r);
		}
		}

Mostrar todos los registros por el dispositivo externo de salida recorriendo del primero al ultimo

```
void mostrarNat(tr v[], int N)
{
 int i=0;
 while(i<N ){
   mostrarReg(v[i]);
   i++;
  }
}
```

```
void mostrarNat(FILE * f)
{
 tr r;
 while(fread(&r, sizeof(r),1,f
){
   mostrarReg(r);

 }
}
```

```
void mostrarNat(Nodo *& l)
{
 tr r;
 while(!=NULL){
  r = pop(l);
  mostrarReg®;
  }
}
```

Mostrar todos los registros en orden inverso

```
void mostrarInv(tr v[], int N)
{
  int u = N-1;

  for( int i = u; i >= 0; i--){


      mostrarReg(v[i]);
  }
}
```

```
void mostrarInv(FILE* f)
{
  int u = cantReg(f);
  tr r;
  for( int i = u; i >= 0; i--) {
  fseek(f, sizeo(r)*i, SEEK_SET);
  fread(&r, sizeof( r), 1, f );
  mostrarReg(r);
 }
}
```

```
void mostrarInv(Nodo* l)
{
  Nodo* p = NULL;
  tr r;
while (l)
    push(p, pop(l));
  while (p){
.    r = pop(p);
     mostrarReg(r);
   }
}
```

Recorrido con corte de control

Vector

```
void corteControl(tr v[], int N)
{

   int i = 0;
   int control;
   while(i<N){
   control = v[i].c1;
   cout<< control<<endl;
     while(I<N && control == v[i].c1){
       cout<< v[i].c2;
       ……..
       I++;
      }
      Mostrar datos del subconjunto
    }
    Mostrar datos totales
 }
```

Archivo

```
void corteControl(FILE * F)
{
   tr r
   fread(&r, sizeof(r), 1,f);
   int control ;
   while(!feof(f)){
   control = r.c1;
   cout<< control<<endl;
     while(!feof(f) && control == v[i].c1){
       cout<< r.c2;
       ……..
       Fread(&r, sizeof(r), 1,f);
      }
      Mostrar datos del subconjunto
    }
    Mostrar datos totales
 }
```

El corte de control requiere al menos un campo que se repite y arma subconjuntos con los registros quepertenecen al mismo. Se produce un corte al leer un valor distinto y se muestran los datos del nuevo sub grupo. Este patron NO ordena, muestra sin repetir el campo que agrupa.
Cree que se puede aplicar este modelo con listas? Puede generalizar el patron?
Que pasa si cada subconjunto tiene en el mismo otro campo que se repite. Se puede hacer un doble corte de control? Se puede resolver con un modelo diferente al propuesto?

Apareo (enumere las precondiciones que deben tenerse en cuenta

Vector

```
void apareo(tr v1[], int N, tr v2[], int M)
{

  int i = 0;
  int j = 0;
  while(i<N&&j<M){
     if(v1[i].c1<v2[j].c1){
        procesar v1[i];
        i ++;
      else{
        procesar v2[j];
        j++;
       }
   }
   while(i<N){
     proceser v1[i];
     i++;
    }
    while(j<M){
     proceser v2[j];
     j++;
    }
 return;
}
```

Otro modelo

```
void apareo(tr v1[], int N, tr v2[], int M)
{
  int i = 0;
  int j = 0;
  while(i<N||j<M){
     if(j==N||i<N&&v1[i].c1<v2[j].c1){
        procesar v1[i];
        i ++;
      else{
        procesar v2[j];
        j++;
       }
   }
 return;
}
```

Archivo

```
void apareo(FILE* f1, FILE * f2)
{
  tr r1,r2;
  fread(&r1, sizeof(tr),1, f1);
  fread(&r2, sizeof(tr1,1,f2);
  while(! feof(f1)&&!foef(f2){
     if(r1.c1<r2.c1){
        procesar r1;
        fread(&r1, sizeof(tr),1, f1);
      else{
        procesar r2;
        fread(&r2, sizeof(tr),1, f2);
       }
   }
   while(!feof(f1)){
     proceser r1;
      fread(&r1, sizeof(tr),1, f1);
    }
    while(!feof(f2)){
     proceser r2;
     fread(&r2, sizeof(tr),1, f2);
    }
 return;
}
```

Puede adaptar el otro modelo para estruntura tipo archivo
Señale semejanzas y-o diferencias entre los modelos
Cual cree mas eficiente (defina las eficiencias que considere)
Que modelo le parece mas adecuado

Los modelos precedentes evaluan con un criterio creciente y por un solo campo, cree que es posible aparear por mas de un campo? Que precondiciones deberían cumplirse? Es posible cambiar el criterio de selección? Se puede establecer la propiedad de generalidad para el tipo de dato? Para el criterio de selección? Justifique sus respuestas
El modelo presentado no evalua la posibilidad de la igualdad en forma explicita. Como esta planteado, en caso de haber igualdades, que estructura se evalua primero? El valor igual cree que

se procesa en la próxima iteración? En caso que asi sea como puede evitarse procesar ese registro? Si hubiera repeticiones en la misma estructura indexada, como se puede procesar solo unrepresentante de ese subconjunto?Habiendo ejemplificado con estructuras del mismo tipo, preguntamos: Puede hacerlo con listas? Si el propósito es aparear dos y generar una tercera, que estructura generaría para optimizar eficiencia? Es posible hacerlo con estructuras de características diferentes? Se pueden aparear tres estructuras? Justifique. Se pueden aparear N estructuras? Los registros de las estructuras a aparear, tienen que ser del mismo tipo? Justifique. Puede pensar un modelo que solo utilice un ciclo de repetición?Se puede aparear por mas de un campo?. Podría conservar la algoritmia?Justifique.

Busqueda secuencial

Vector

```
int buscarSec(tr v[], int bus, int N){
    i = 0;
    while(i < N && v[i].c1 != bus){
            i++;
    }
    return i<N? i: -1;
}
```

Lista

```
Nodo * buscarSec(Nodo * l, int bus){

    while(l && l->info.c1 != bus){
        l = l->sgte;
    }
    return l != NULL&&l->info!=bus? l: NULL;
}
```

Se puede buscar por coincidencia de mas de un campo?Puede generalizar por tipo de dato y criterio? Justifique Puede optimizar el modelo? Justifique. Que opina de la aplicación de este patron en archivos?Que condición deben cumplir los datos para que el patron se pueda aplicar? Las expresiones dentro del while se pueden invertir sin alterar la selección y resultado?

Busqueda directa

Vector

```
tr búsquedaDir(tr v[], int indice){




    return v[indice];
}
```

Archivo

```
tr búsquedaDir(FILE * f, int indice){
    tr r;
    fseek(f, sizeof(tr)*índice, SEEK_SET);
    fread(&r, sizeof(tr), 1 f);
    return r;
}
```

Puede utilizar un patron similar en estructuras enlazadas
Que condicion debe cumplirse, en la estructura y la clave para aplicar este patron?

Busqueda binaria

Vector

```
Int busBin( tr v[], int aBuscar, int N, int &pri){
 int ult = N-1, pri = 0, m;

 while(pri>=ult)
      m = (pri u ult)/2;



      if (v[m]. ci == aBuscar) return m;
      if( aBuscar>v[m].c1) pri = m+1;
      else ult = m-1;
  }
 return -1;
}
```

Archivo

```
Int busBin( FILE*f, int aBuscar){
 int ult = CantReg(f)-1, pri = 0, m;
 tr r;
 while(pri>=ult)
      m = (pri u ult)/2;
      fseek((f, m*sizeof(r), SEEK_SET
      fread(&r, sizeof(r),1 f);
      if (r. ci == aBuscar) return m;
      if( aBuscar>v[m].c1) pri = m+1;
      else ult = m-1;
 }
 return -1;
}
```

Es aplicable el modelo en estructuras enlazadas? Justifique. Es posible buscar por dos campos? Que condición debe darse? Cambiaria lógica, algoritmia o ambas cosas? Justifique. Que criterio de ordenamiento supone la búsqueda desarrollada, puede generalizar criterios? Y tipo de dato? Que cambios haría para combinar búsqueda binaria por un campo con repetición con directa, binaria o secuencial por otro campo? Que condiciones deberían cumplirse para poder efectivizat esto, que vambios propondría? Como se puede implementar la búsqueda binaria con una función recursiva? Que piensa de la eficiencia en este modelo?

Metodo de ordenamiento

Vector

```
void ordenarVector(tr v[], int N) {// v el vector a ordenar N cantidad de componentes
        int i,j;
        tr aux;
        for(i = 1; i < N; i++){// pasos → 1..N-1
                for( j = 1; j <= N – i;j++){//comparaciones en cada paso → 1..N-i
                        if(v[j-1].c1 > v[j].c1{
                                aux = v[j];
                                v[j] = v[j -1];
                                v[j – 1] = aux;
                        }
                } // fin ciclo interno
        }// fin ciclo externo
return;
} // fin de la función
```

Ejemplo del concepto de modificar la lógica de selección manteniendo la algoritmia ordenando por dos campos

```
void ordenarVector(tr v[], int N) {// v el vector a ordenar N cantidad de componentes
        int i,j; tr aux;
        for(i = 1; i < N; i++){// pasos → 1..N-1
                for( j = 1; j <= N – i;j++){//comparaciones en cada paso → 1..N-i
                        if((v[j-1].c1 > v[j].c1)|(v[j-1].c1 == v[j].c1 &&v[j-1]-c2>v[j].c2){
                                aux = v[j];
                                v[j] = v[j -1];
                                v[j – 1] = aux;
                        }
                } // fin ciclo interno
        }// fin ciclo externo
return;
} // fin de la función
```

Ejemplo con plantilla

```
Template <typename T> void ordenarVector(tr v[], int N, int(criterio(T,T) {
        int i,j; T aux;
        for(i = 1; i < N; i++){
                for( j = 1; j <= N – i;j++){
                        if((criterio(v[j-1], v[j]==1{
                                aux = v[j];
                                v[j] = v[j -1];
                                v[j – 1] = aux;
                        }
                } // fin ciclo interno
```

```
        }// fin ciclo externo
return;
```

función de ordenamiento creciente del campo 1

```
int campo1creciente(tr a, tr b){
   return a.c1>b.c1?1:0;
}
```

Ejemplo invocacion

```
ordenarVector<tr>(v, n. campo1creciente)
```

Es aplicable este método para archivos o estructuras enlazadas?
Que entiende por carga directa con PUP posición única prececible?
En que tipo de estructura puede ser aplicable?
Que características debe tener la clave de búsqueda
Que método de ordenamiento es aplicable en listas ordenadas?
Es aplicable la combinación Busqueda binaria – binaria; búsqueda directa – directa; búsqueda binaria – directa; búsqueda binaria – secuencial?
Si en lo anterior algún modelo es factible ejemplifique.

Ejemplo de plantillas y punteros a funciones

Recorrer sin plantilla y criterio

```
void mostrar(int v[], int n){

   for(int i=0; i<n; i++){
      cout <<(v[i]<<endln;
   }

   return;
}
```

Recorrer con plantilla y puntero

```
template <typename T> void mostrar(T v[], int n),void (*ver(T))){
  for(int i=0; i<n; i++){
    ver(v[i];
  }
  return;
}

void ver_reg(TipoReg r){
 cout << r.c1.....<<endl;
 return;
}
mostrar<tipoReg>(v, 10,ver_reg);
```

Cargar sin repetir la clave

Vector sin orden

```
void cargarSinRepetir(tr v[], tr aCargar, int &N)
{

   Int pos = buscarSec(v, aCargar, N);
   If (pos == -1){

   v[N] = aCargar;
        N++;
  }
   return;
}
```

Vector ordenado

```
void cargarSinRepetir(tr v[], tr aCargar, int &N)
{
    int pri = 0, i = N;
    int pos = busBin(v, aCargar, N, pri);
    If (pos == -1){
       for( ;i>= pri; i--) v[i+1] = v[i¨];
       v[pri]= a Cargar;
      N++;
     }
  return;
}
```

Es aplicable este modelo en archivos?
De tener necesidad de hacerlo, que estrategia utilizaría?

Recuerde que en listas ordenadas existen las funciones insertarOrdenado y cragar sin repetir
Como diferencia tamaño lógico de físico, si el parámetro N es el tamaño lógico, que precondición debe establecer para no superar el tamaño físico? justidique

```
void mostrar(int v[], int n)
{

   for(int i=0; i<n; i++){
      cout <<(v[i]<<endln;
   }

   return;
}
```

```
template <typename T> void
mostrar(T v[], int n),void
(*ver(T)))
{
   for(int i=0; i<n; i++){
      ver(v[i];
   }

   return;
}

void ver_reg(TipoReg r){
 cout << r.c1…….…<<endl;
 return;
}
mostrar<tipoReg>(v, 10,ver_reg);
```

Observe la función ver dice void (*ver (T)) retorna el valor ausente (void) y es un puntero a una función que evalua un tipo de dato, genérico llamado aquí T. Aquí generaliza el tipo de dato, observe en la invocación que entre corchetes angulares se establece el tipo de dato particular a evaluar en este caso y ver_reg invoca a la función particular donde se indica que mostrar en este caso particular. Aquí se puede generalizar el criterio de selección de lo que busca mostrar. Aquí formulamos algunas preguntas porque la declaración es void (* ver(T)) ? podría ser void*ver(T)? que significado tienen los paréntesis? Defina la semántica de las dos declaraciones. Observe además la invocación, cual es la semántica de ese identificador? Es correcto que no tenga parámetros? Justifique su respuesta

Los patrones para estructuras enlazadas serán desarrollados luego de la exposición teorica y conceptual de los temas