github.com/nahuelArn

Resumen Algoritmos Taller

Algoritmos de Ordenacion

Insercion

```
const
  PosIni = 0;
Procedure ordenarVectorInsercion(var vec: v; dimL: integer);
var
    pos : integer;
    elemAct,i,b: integer;
begin
    pos:= 0;
  for i := 1 to (dimL-1) do
    begin
      elemAct := vec[i];
      pos := i - 1;
      while( (Pos > (PosIni-1) )and (vec[pos] > elemAct)) do  //si el anterior es mas grande que el segundo
        begin
          vec[pos+1] := vec[pos];
          pos := pos - 1;
        end;
        vec[pos+1] := elemAct;
    end;
end;
```

Seleccion

```
procedure ordenarVectorSeleccion(var v: vOrdenar; dimL: integer);
var a,b,i,min: integer;
begin
  for i:= 1 to (dimL-1) do
    begin
      a:= i;
      for b:= i+1 to dimL do
        begin
          if(v[b] < v[a])then
            begin
              a:= b;
            end;
          min:= v[a];
          v[a]:= v[i];
          v[i]:= min;
        end;
    end;
end;
```

github.com/nahuelArn

Algoritmos De Busquedas

de manera iterativa

Con Funcion

```
Function buscarNumero(numAbuscar: integer; v1: vPrimero; dimL: integer): Boolean;
Var
  ini,mitad,fin: integer;
  encontrado: Boolean;
Begin
  encontrado := false;
  ini := 1;
  fin := dimL;
  While ((ini <= fin) And (encontrado <> true)) Do
    Begin
      mitad := (ini+fin) Div 2;
      If (v1[mitad] =numAbuscar) Then
        encontrado := true
      Else
        If (numAbuscar < v1[mitad])Then
          fin := mitad-1
      Else
        If (numAbuscar> v1[mitad])Then
        ini := mitad+1;
    End;
  buscarNumero := encontrado;
End;
```

Encontre el Valor Buscado

Se tira para la izquierda

Se tira para la derecha

github.com/nahuelArn

Algoritmos De Busquedas

de manera recursiva

```
procedure busquedaDicotomicaRecursiva (v: vector ; ini: integer ; fin: integer ; valorBuscado: integer ; var pos: integer;var encontrado:Boolean);
var
  mid: integer;
begin
  if (ini > fin) then
    pos:= -1
  else
    begin
      mid:= (ini + fin) div 2;
      if (valorBuscado = v[mid]) then
        begin
          pos:= mid;
          encontrado:= true;
        end
      else
        begin
          if (valorBuscado < v[mid]) then
            busquedaDicotomicaRecursiva(v, ini, (mid - 1), valorBuscado, pos,encontrado)
          else
            busquedaDicotomicaRecursiva(v, (mid + 1), fin, valorBuscado, pos,encontrado);
        end;
    end;
end;
```

Impresion en Arboles

```
procedure imprimirPreOrden(a: arbol);
begin
  if(a <> nil)then
    begin
      Writeln('PreOrden: full izquierda, full Derecha(extremos)',a^.dato);
      imprimirPreOrden(a^.hi);
      imprimirPreOrden(a^.hd);
    end;
end;

procedure imprimirInOrder(a: arbol);
begin
  if(a <> nil)then
    begin
      imprimirInOrder(a^.hi);
      Writeln('In Order: de menor a mayor',a^.dato);
      imprimirInOrder(a^.hd);
    end;
end;
```

```
//espejo
procedure imprimirInOrder(a: arbol);
begin
  if(a <> nil)then
    begin
      imprimirInOrder(a^.hd);
      Writeln('In Order: de mayor a menor',a^.dato);
      imprimirInOrder(a^.hi);
    end;
end;


procedure imprimirPosOrder(a: arbol);
begin
  if(a <> nil)then
    begin
      imprimirPosOrder(a^.hi);
      imprimirPosOrder(a^.hd);
      Writeln('PosOrder: full derecha, full izquierda(extremos)',a^.dato);
    end;
end;
```

github.com/nahuelArn

Operaciones sobre Arboles (entre rangos)

```
function cantEntreRangos(a: arbol0; izquierda,derecha: integer): integer;
begin
  if(a = nil)then
    cantEntreRangos:= 0
  else
    begin
      if(a^.dato.isbn >= izquierda) and (a^.dato.isbn <= derecha)then
      //if(a^.dato.isbn > izquierda) and (a^.dato.isbn < derecha)then //sin los limites incluidos
        cantEntreRangos:= cantEntreRangos(a^.hi,izquierda,derecha) + cantEntreRangos(a^.hd,izquierda,derecha)+1
      else
        begin
          if(a^.dato.isbn > izquierda)then
            cantEntreRangos:= cantEntreRangos(a^.hi,izquierda,derecha)
          else
            begin
              if(a^.dato.isbn < derecha)then
                cantEntreRangos:= cantEntreRangos(a^.hd,izquierda,derecha)
            end;
        end;
    end;
end;
```

Mayores a X valor

```
procedure mayoresQue(a: arbol;x: integer; var cant: integer);
begin
  if(a <> nil)then
    begin
      if(a^.dato > x)then
        begin
          cant:= cant+1;
          mayoresQue(a^.hi,x,cant);
         mayoresQue(a^.hd,x,cant);
        end
      else
        mayoresQue(a^.hd,x,cant);
    end
end;
```

```
function mayoresQue (A: arbol; valor: integer) : integer;
begin
  if(a = nil)then
    mayoresQue:= 0
  else
    begin
      if (A^.dato.numero > valor) then
        mayoresQue := mayoresQue(A^.HI, valor) + 1 + mayoresQue(A^.HD, valor)
      else
        mayoresQue := mayoresQue(A^.HD, valor);
    end;
end;
```

Menores a X valor

```
procedure menoresQue(a: arbol;x: integer; var cant: integer);
begin
  if(a <> nil)then
    begin
      if(a^.dato < x)then
        begin
          cant:= cant+1;
          menoresQue(a^.hi,x,cant);
          menoresQue(a^.hd,x,cant);
        end
      else
        menoresQue(a^.hi,x,cant);
    end
end;
```

```
function menoresQue (A: arbol; valor: integer) : integer;
begin
  if(a = nil)then
    menoresQue:= 0
  else
    begin
      if (A^.dato.numero < valor) then
        menoresQue := menoresQue(A^.HI, valor) + 1 + menoresQue(A^.HD, valor)
      else
        menoresQue := menoresQue(A^.HI, valor);
    end;
end;
```

AgregarAtrasIneficiente

```
procedure agregarAtrasIneficiente(var L: lista; p: pelicula);
var
  nue,ant,act: lista;
begin
  new(nue);
  nue^.dato:= p;
  ant:= L;
  act:= L;
  While (act <> nil) do
    begin
      ant:= act;
      act:= act^.sig;
    end;
  if(ant = act)then
    L:= nue
  else
    ant^.sig:= nue;
  nue^.sig:= act;
end;
```

Arbol de listas

```
procedure inicializarListaArb2(var L: listaRepetidos);
begin
  L:= nil;
end;
//es la variacion del agregarAtras con un Ult, pero en este caso se tie
procedure agregarAtrasIneficiente(var L: listaRepetidos; p: prestamo);
var
  ant,act,nue: listaRepetidos;
begin
  new(nue);
  nue^.dato:= p;
  ant:= L;
  act:= L;
  While(act <> nil)do
    begin
      ant:= act;
      act:= act^.sig;
    end;
  if(ant = act)then
    L:= nue
  else
    ant^.sig:= nue; //entonces ant = era el ultimo en la lista
  nue^.sig:= act; //al nue . sig le asigno nil,
end;
```

```
prestamo = record
  isbn: integer;
  numSocio: integer;
  dia: rango31;
  mes: rango12;
  cantDiasPrestados: integer;
end;
```

```
arbol2 = ^a2; //arbol 2

a2 = record
  dato: listaRepetidos;
  hd: arbol2;
  hi: arbol2;
end;
```

Arbol de listas

```
//carga el arbol de listas
procedure cargarArbol2(var a2: arbol2; p: prestamo);
var
  aux: prestamo;
begin
  if(a2 = nil)then
    begin
      new(a2);
      inicializarListaArb2(a2^.dato);
      agregarAtrasIneficiente(a2^.dato,p);
      Writeln(); Writeln();
      Writeln('Que valor se esta guardando en el arbol2 ',a2^.dato^.dato.isbn);
      Writeln(); Writeln();
      a2^.hi:= nil;
      a2^.hd:= nil;
    end
  else
    begin
      aux:= a2^.dato^.dato;
      if(p.isbn = aux.isbn)then
        agregarAtrasIneficiente(a2^.dato,p)
      else
        begin
          if(p.isbn < aux.isbn)then
              cargarArbol2(a2^.hi,p)
          else
            cargarArbol2(a2^.hd,p);
        end;
    end;
end;
```

```
//carga de los arboles de forma iterativa
procedure cargarPrestamos(var a0: arbol0;var a2: arbol2);
var
  p: prestamo;
begin
  leerPrestamo(p);
  While(p.isbn <> -1)do
    begin
      cargarArbol0(a0,p);
      cargarArbol2(a2,p);
      leerPrestamo(p);
    end;
end;
```

Arbol de listas

```
//modulo Para los 2 arboles imprimir
procedure imprimirPrestamo(p: prestamo);
begin
  Writeln();
  Writeln('El isbn del arbol: ',p.isbn);
  Writeln('El numero de socio: ',p.numSocio);
  Writeln('El dia del prestamo del socio: ',p.dia);
  Writeln('El mes del prestamo ',p.mes);
  Writeln('La cantidad de dias prestado ',p.cantDiasPrestados);
  Writeln('-------------------------------------------------------------');
end;
//Modulo del Arbol normalito
procedure imprimirInOrder0(a0: arbol0);
begin
  if(a0 <> nil)then
    begin
      imprimirInOrder0(a0^.hi);
      imprimirPrestamo(a0^.dato);
      imprimirInOrder0(a0^.hd);
    end;
end;
//Modulos del Arbol de Listas (recorre una lista y va imprimiendo su contendio)
procedure imprimirLista(L: listaRepetidos);
begin
  While(L <> nil)do
    begin
      imprimirPrestamo(L^.dato);
      L:= L^.sig;
    end;
end;
//recorre el arbol, cuando llega a nil empieza el backtracking y va mandando la lista que esta en ese nodo a otro modulo
procedure imprimirInOrder2(a2: arbol2);
begin
  if(a2 <> nil)then
    begin
      imprimirInOrder2(a2^.hi);
      imprimirLista(a2^.dato);
      imprimirInOrder2(a2^.hd);
    end;
end;
```

Vector de Arboles

```
  dimF = 10;
type

  rango10 = 1..dimF;

  producto = record
    cod: integer;
    rubro: rango10;
    stock: integer;
    precioUnitario: real;
  end;

  arbol = ^nodo;

  nodo = record
    dato: producto;
    hi: arbol;
    hd: arbol;
  end;

  vectorRubros = array[rango10]of arbol;
```

```
procedure inicializarPuntero(var a: arbol);
begin
  a:= nil;
end;

procedure inicializarArboles(var v: vectorRubros);
var
  i: integer;
begin
  for i:= 1 to dimF do
    inicializarPuntero(v[i]);
end;
```

Vector de Arboles

github.com/nahuelArn

```
procedure leerProducto(var p: producto);
begin
  Writeln('Ingrese cod (-1 para cortar)');
  readln(p.cod);
  if(p.cod <> -1)then
    begin
      Writeln('Ingrese rubro ');
      readln(p.rubro);
      Writeln('Ingrese stock');
      readln(p.stock);
      Writeln('Ingrese precioUnitario');
      readln(p.precioUnitario);
    end;
end;

procedure agregarProductos(var a: arbol;p: producto);
begin
  if(a = nil)then
    begin
      new(a);
      a^.dato:= p;
      a^.hi:= nil;
      a^.hd:= nil;
    end
  else
    begin
      if(p.cod <= a^.dato.cod)then
        agregarProductos(a^.hi,p)
      else
        agregarProductos(a^.hd,p);
    end;
end;
```

Vector de Arboles

```
procedure cargarProductosEnRubros(var v: vectorRubros);
var p: producto;
begin
  leerProducto(p);
  While(p.cod <> -1)do
    begin
      agregarProductos(v[p.rubro],p);
      leerProducto(p);
    end;
end;
```

Vector de Arboles

github.com/nahuelArn

```
procedure imprimirNodo(p: producto);
begin
  Writeln('El cod es: ',p.cod);
  Writeln('El rubro es: ',p.rubro);
  Writeln('Ingrese stock',p.stock);
  Writeln('Ingrese precioUnitario',p.precioUnitari
end;

procedure imprimirArbol(a: arbol);
begin
  if(a <> nil)then
    begin
      imprimirArbol(a^.hi);
      imprimirNodo(a^.dato);
      imprimirArbol(a^.hd);
    end;
end;

procedure imprimirVector(v: vectorRubros);
var i: integer;
begin
  for i:= 1 to dimF do
    begin
      if(v[i] <> nil)then
        begin
          Writeln(); Writeln();
          Writeln('estas parado en el rubro ',i);
          imprimirArbol(v[i]); //rubro
          Writeln(); Writeln();
        end;
    end;
end;
```