

## TADESCALONADA.pdf



Anónimo



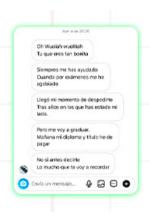
Análisis de Algoritmos y Estructuras de Datos



2º Grado en Ingeniería Informática



Escuela Superior de Ingeniería Universidad de Cádiz



Que no te escriban poemas de amor cuando terminen la carrera

(a nosotros por

(a nosotros pasa)

WUOLAH

Suerte nos pasa)







## No si antes decirte Lo mucho que te voy a recordar

## (a nosotros por suerte nos pasa)

## **TAD Escalonada**

// Una funcion f: R-r Es escalonada cuando consiste en una sucesion de funciones constantes definidas

//en subintervalos disjuntos y contiguos es decir f puede definirse mediante condiciones de la forma

// f(x) = yi si xi es menor o igual que z y es menor que xi+1 donde los valores yi son distintos para subinervalos

// adyacentes. Cada uno de los puntos (xi,yi) en lo que la funcion cambia de valor se llama salto o escalon

// IMAGEN

// Especificar un tad para las funciones escalonadas con un numero finito de saltos que incluya las sig operaciones

// - crear una funcion constante f(x) = yi definida a partir de xi

// - añadir un nuevo salto en el punto xi a una funcion, si ya existe un salto en las coordenadas x, se subsituye por el nuevo

// - eliminar el escalon definido en el subintervalo al que pertenece la coordenada x, el escalor anterior al

// eliminado se prolonga hasta el siguiente

// void eliminarSalto ( double xi, double yi);

// - calcular el valor de una funcion en un punto dado

// - calcular el valor minimo de una funcion escalonada

// - calculo el valor máximo de una funcion escalonada

// - hacer una traslacion de una funcion w unidades horizontales y z unidades verticales , siendo w y z ->R

// - destruir una funcion

// Definicion : Una funcion escalonada es una sucesion de funciones constantes definidas en los subintervalos disjuntos

// y contiguos, es decir f puede definirse mediante condiciones de la forma

// f(x) = yi, si xi es menor o igual que z y es menor que xi+1 donde los valores yi son distintos para subinervalos

// adyacentes. Cada uno de los puntos (xi,yi) en lo que la funcion cambia de valor se llama salto o escalon

//Especificaciones
// Escalonada();
// postCondicion : crea una funcion escalonada vacia ;
// void añadirSalto( double xi, double yi);
// postCondicion : añade un nuevo salto a la funcion escalonada, si ya existe un valor para la x, se sustituya



```
// postCondicion: elimina un salto de la funcion escalonada y se prolonga el escalon
anterior hasta el siguiente, si el
// punto no existe , no hace nada
// double valorPunto (double xi, double yi );
// postCondicion : devuelve el valor de la funcion en el punto (xi,yi), si el punto no existe
en la funcion devuelve
// un valor muy alto
// double valorMinimo();
// postCondicion : devuelve el minimo de la funcion, si la funcion esta vacia devuelve un
valor muy alto
// double valorMaximo();
// postCondicion : devuelve el maximo de la funcion, si la funcion esta vacia, devuelve un
valor muy bajo
// void translacion ( double w , double z);
// postCondicion: translada la funcion w unidades horizontales y z unidades verticales
// ~Escalonada();
// destruye la funcion escalonada;
typedef struct {
  double xi, yi;
  Salto(double a, double b) : xi(a), yi(b) {};
}salto;
class Escalonada{
public:
  explicit Escalonada(int numSaltos, double xi = 0 double yi = 0);
  void añadirSalto( double xi, double yi);
  void eliminarSalto (double xi, double yi);
  double valorPunto (double xi, double yi );
  double valorMinimo();
  double valorMaximo();
  void translacion (double w, double z);
  ~Escalonada();
private:
  int saltosMax, saltosActuales;
  Lista<salto> funcion();
  //double minimo , maximo ;
  Lista<salto>::posicion posicion;
};
Escalonada::Escalonada(int numSaltos, double xi = 0 double yi = 0){
  //minimo = max();
  //maximo = min();
```



```
saltosMax = numSaltos;
  saltosActuales =0;
  funcion.insertar(salto(xi,yi));
}
void Escalonada::añadirSalto( double xi, double yi){
  //modificar lo anterior
}
void Escalonada::eliminarSalto ( double xi, double yi){
  funcion.eliminar(buscar(xi,yi));
}
double Escalonada::valorPunto (double xi, double yi ){
}
double Escalonada::valorMinimo(){
// recorrer toda la lista y ver el valor del punto
double Escalonada::valorMaximo(){
  //
void Escalonada::translacion ( double w , double z){
// Sumar a todas las x w y a las y z
Escalonada::~Escalonada(){
  ~funcion();
}
```

