UNIVERSIDAD
DE SANTIAGO
DE CHILE

**Departamento de Matemática y Ciencia de la Computación**

# Laboratorio 4

## Dating On-Line

### Segundo Semestre 2016

Programación Avanzada 26106

Licenciatura en Ciencia de la Computación

Silvana Cerda Aravena
Alonso Maripi Vallejos
silvana.cerda@usach.cl / alonso.maripi@usach.c

# 1 Introducción

El usuario tiene un sistema que especifica una cantidad n de actividades las cuales deben ser calificadas, de esta forma se forma un poligono denominado "Diagrama Radical",estos puntos son ubicados en el centro del diagrama.

Se necesita un programa que calcule la mayor area ya que se necesita evaluar cual es el perfil con mas éxito.

# 2 Procedimiento

Se debe ingresar la cantidad de puntos que deseamos ingresar, luego de esto debemos dar los valores de cada punto para poder ubicarlos en el poligono base, con la ayuda de la liberia stdlib.h llamamos a la función qsort la cual ordena de menor a mayor los datos, una vez que tenemos el arreglo ordenado intercalamos los valores para calcular el area máxima del poligono formado.

## 2.1 Restricciones

Se debe ingresar datos de tipo entero, la cantidad de puntos debe ser superior o igual a 3 ya que necesitamos formar un polígono, para evitar problemas con los calculos del area se utilizo variables de tipo double.

# 3 Estructuras de Datos Utilizadas

Para este problema no se utilizaron estructuras de datos.

# 4 Algoritmo

**Algorithm** TotalArea(Area, Angle, Data[0 ... n-1]).

**Input**: $Area$ : The area of the radial diagram it's initializated in 0,
$Angle$ : the angle that exist betwen two consecutive points,
$Data[0...n-1]$ : the values of the n activities(0 - 100).
**Output**: $Area$ : The total area of the radial diagram.

```
1  Angle ← sin(Angle)
2  for i ← 0 to n − 2 do
3      Area ← Area + (0.5 ∗ Data[i] ∗ Data[i + 1] ∗ Angle)
4  Area ← Area + (0.5 ∗ Data[0] ∗ Data[n − 1] ∗ Angle)
5  return Area
```

**Algorithm** MaxArea(Data[0 ... n-1], Ndata[0 ... n-1]).

**Input**: $Data[0...n-1]$ : Array that contains the values of the n activities,
$Ndata[0...n-1]$ : An empty array of the same length that Data[n].
**Output**: $Ndata[0...n-1]$ : An array with the values of Data[0 ... n-1] sorted in away that allow to get a maximum area form the radial diagram.

```
1   if n >= 3
2   j ← 0
3   cont ← n − 1
4   Ndata[0] ← Data[0]
5   for i ← 0 to n − 1 do
6       if i % 2 != 0
7           Ndata[j] ← Data[i]
8           j ← j + 1
9       else
10          Ndata[cont] ← Data[i]
11          cont ← cont − 1
12  return Ndata
```

El programa se divide en dos parte importantes ya que se utiliza una implementacion del metodo de ordenamiento Quick Sort que esta presente en una libreria de C (stdlib.h).
Nota : Para calular el area del poligono generado utilizamos e implementamos razones trigonometricas.
Ahora, por un lado tenemos en la primera parte el algoritmo que calcula el area del poligono generado por los valores del diagrama radial, donde este recibe el arreglo que contiene los valores dichos anteriormente ya ordenados y reordenados de forma tal que al pasar por esta funcion se obtenga un area maxima(obiamente mayor o igual que si se calcularan los valores en el orden originalmente ingresados). Ademas, recibe el valor del angulo que generan los puntos entre si, pero este es ingresado como parametro de la funcion que le precede, su valor es recalculado mas adelante. Finalmente recibe el valor del area total que es inicializado en 0 desde el main.
En primera instancia el programa recalcula el valor del seno del angulo recibido como parametro(linea 1).Luego, recorre cada elemento del arreglo desde el elemento 0 hasta el n-1 en pares(linea 2-3) para calcular el area. Finalmente calcula el par que queda fuera del proceso for, el priemer elemento (Data[0]) con el ultimo(Data[n-1], linea 4). Al final se obtiene el area total maximizada del diagrama radial.

El segundo algoritmo es el que reordena los elementos del arreglo Data[0 ... n-1] (que en ese momento ya se encuentra ordendado de menor a mayor) en un nuevo arreglo auxiliar Ndata del mismo largo que el primero.

Siendo que el arreglo original ya se encuentra ordenado, el algoritmo sigue una funcion donde los elementos de este los ordena conforme a sus posiciones. Si un valor se encuentra en una posicion impar estos se guardan en el inicio del primer arreglo y por el contrario, si se encuentran en una posicion par, se guardan desde el final del arreglo hasta que se recorra y reordene por completo el primer arreglo. De esta forma, se obtiene una maximizacion del area original.

### 4.1 Análisis de Complejidad

El algoritmo propuesto tiene una complejidad de nlogn.

## 5 Implementación

Para solucionar el problema se eligio el lenguaje C.

```c
/*
 * FILE: ACM.c
 *
 * DESCRIPTION: A program that Maximize and calculate the area formed by a radial diagram.
 *
 * AUTHOR: Alonso Maripi Vallejos & Silvana Cerda Aravena
 *
 * LAST REVISED: Santiago de Chile, 28/11/2016
 *
 */

   #include <stdio.h>
   #include <stdlib.h>
   #include <sys/types.h>
   #include <unistd.h>
   #include <time.h>
   #include <math.h>

   double CalcA(double An, unsigned int Va1, unsigned int Va2){
       return (0.5*Va1*Va2*An);
   }

   double TotalArea(double area, double angle, unsigned int *data, unsigned int num){

       angle = sin(angle);

       for(int i = 0;(i + 1) < num;i++){
           area = area + CalcA(angle, data[i], data[i+1]);

       }
       area = area + CalcA(angle, data[0], data[num-1]);
   }

   int cmpfunc (const void * a, const void * b)
   {
       return ( *(int*)a - *(int*)b );
   }

   int main() {

       double area = 0;
       int izq, der;
       unsigned int *value,*value2, date;

       scanf("%u",&date);

       value = calloc(date,sizeof(unsigned int));
       value2 =calloc(date,sizeof(unsigned int));
       for(int i = 0;i < date;i++){

           scanf("%u",&value[i]);
           if(value[i]>100)
               value[i] = 100;
       }
       qsort(value, date,sizeof(unsigned int),cmpfunc);

       if(date>=3) { //si tenemos mas de 3 puntos
           int j=0;
           int cont = date-1;  // corresponde a la posicion penultima
           value2[0]=value[0]; // respaldo
           for  (int i= 0; i<date;i++){  //recorremos
               if (i%2!=0){ //si es distinto a modulo de 2 tomamos el valor
                       value2[j]=value[i]; //toman2 valor
                       j++; //avanzamos
               }
                 else {
                   value2[cont]=value[i];  //cuando se hayan acabado los distintos a modulo de 2 analizamos desde aca
                     cont--; //los enviamos al final de la lista
                 }
            }
       }
       area = TotalArea(area, 2*M_PI/date, value2, date);

       printf("%.3lf\n", area);
       return 0;
}
```

### 5.1 Plataforma Computacional

El programa fue ejecutado en 2 computadores con las siguientes características:

- **Computador n1**
- **Procesador:** Intel(R) Core(TM) i7-3520M 2.9 GHz
- **Memoria RAM:** 8192MB RAM
- **Sistema Operativo:** Ubuntu Gnome 16.04

- **Computador n2**
- **Procesador:** Inter(R) Core(TM) i3-4005U 1.70 GHz
- **Memoria RAM:** 6144MB RAM
- **Sistema Operativo:** Ubuntu 15

## 6 Resultados Experimentales

Los resultados del algoritmo son triviales y resuelven de manera exitosa el area máxima de los puntos ingresados.