Práctica 3

Greedy

Laura Gómez Garrido, Pedro Bonilla Nadal, Javier Sáez Maldonado, Daniel Pozo Escalona, Luis Ortega Andrés

2º Doble Grado Informática y Matemáticas

Índice

Problema.
Se pide diseñar e implementar un algoritmo greedy que solucione el problema de encontrar la triangulación mínima de un polígono convexo. Como salida, se deberá proporcionar el conjunto de aristas que forman la triangulación junto con el coste (perímetro de cada triángulo generado).

Conceptos.
Un polígono del plano se dice **convexo** si, para cada dos puntos cualesquiera del interior del polígono, el segmento que los une está contenido en el mismo.
Por la **triangulación de un polígono plano** entenderemos la división de éste en un conjunto de triángulos, con la restricción de que cada lado de un triángulo se reparta entre dos triángulos adyacentes (obviamente, sólo pertenece a uno si está en la frontera del polígono).
TODO: EJEMPLO POLIGONO TRIANGULADO

Explicación Algoritmo.

- Obtener todas las cuerdas (segmentos que unen nodos que se encuentran a 2 de distancia) del polígono.
- Seleccionar la cuerda de menor longitud.
- Eliminar el nodo aislado por dicha cuerda.
- Aplicar el algoritmo al polígono resultante.
- El caso base se cumple cuando el poligono tiene 4 nodos.

Circular Advance

```
auto circular_advance = [](auto& forwdIt, auto initValue,
    auto endValue, int n) {
   for(int i=0; i<n; i++)
   {
      forwdIt++;
      if(forwdIt == endValue)
         forwdIt = initValue;
   }

   return forwdIt;
};
```

Avanza un elemento (iterator or integer) como en una lista circular de n en n.

Find Min String

```
auto find_min_string = [&min_distance, &it_min, &min_string, &points](auto initial_point_it ) {
    auto p0 = initial_point_it ;
    auto p = p0;

    do {
      auto prev = p;

      circular_advance(p, points.begin(), points.end(), 2);
      if (min_distance > euclidean_distance(*prev, *p))
      {
          min_distance = euclidean_distance(*prev, *p);
            min_string = std::make_pair(*prev, *p);
            it_min = circular_advance(prev, points.begin(), points.end(), 1);
          }

     }while(p != p0);
};
```

Busca la cuerda de mínima longitud recorriendo los nodos hasta volver al nodo inicial.

Algoritmo

```
while(points.size() > 3)
{
   auto it = points.begin();
   // Unconditionally find minimum length string
       starting
   // at the first element
   find_min_string(it);

   // If the polygon's number of edges is even, we need
       to
   // do the same starting at the second one
   if(!(points.size()%2)) find_min_string(++it);

   sol.push_back(min_string);
   points.erase(it_min);
}
```