Centro de Investigación en Matemáticas A.C.
Programación y Algoritmos 1 (2019)

Tarea 14 Envolvente Convexa

Marco Antonio Esquivel Basaldua

Un polígono convexo es un polígono cuyos ángulos interiores miden a lo más 180 grados o π radianes. Ejemplos de polígonos convexos son todos los triángulos y los polígonos regulares.

La envolvente convexa de un set de puntos es el polígono más pequeño en el cual cada punto en el set está ya sea en los límites del polígono formado o en su interior. Un ejemplo de una envolvente convexa se aprecia en la figura 1 en la que el polígono formado puede considerarse como una banda elástica que se cierra sobre un set de clavos que representan los puntos en el set.

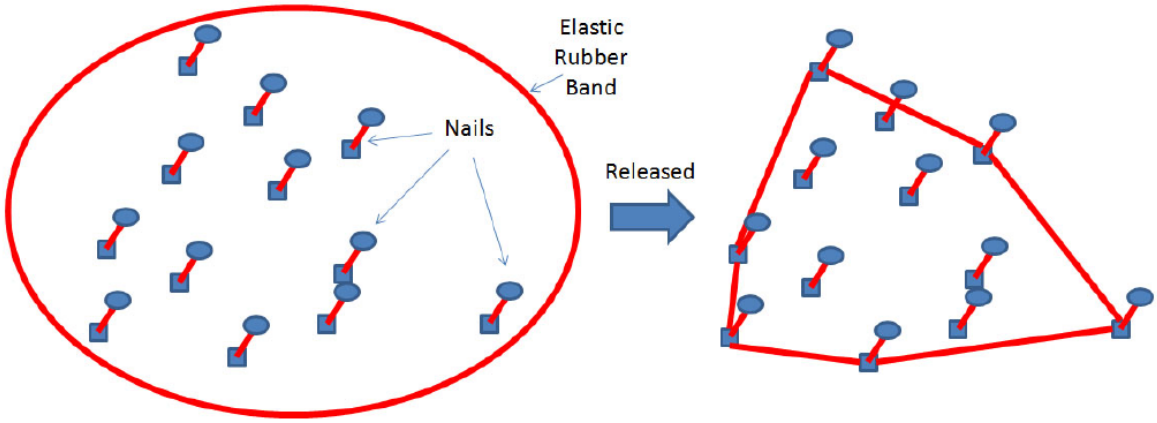

Figura 1. Envolvente convexa vista como una banda elástica.

Cada vértice en el polígono formado es un punto en el set por lo que un algoritmo para encontrar la envolvente convexa es esencialmente un algoritmo para decidir cuáles puntos en el set serán considerados como vértices en la envolvente convexa.

En este reporte se describen y presentan resultados aplicando los algoritmos de la Marcha de Jarvis y el Scan de Graham.

Marcha de Jarvis

Llamado así en nombre de R. A. Jarvis quien lo publicó en 1973, es un algoritmo con complejidad $O(hn)$, donde n es el número de puntos en el set y h es el número de puntos en la envolvente convexa. El peor caso en la complejidad de este algoritmo es $O(n^2)$ que resulta en el caso en que todos los puntos del set dado estén en la envolvente.

El algoritmo inicia con $i = 0$ y un punto p_0 que se sabe que está en la envolvente convexa, para ello se suele seleccionar el punto que esté mas hacia la izquierda o más abajo en coordenadas cartesianas, y selecciona p_{i+1} tal que el resto de los puntos queden hacia la derecha (o hacia arriba si se toma p_0 como el punto que está más abajo) de la línea $p_i p_{i+1}$. Este punto puede ser encontrado en un tiempo $O(n)$ comparando los ángulos polares de todos los puntos con respecto al punto p_i. El proceso se repite hasta que p_0 sea alcanzado desde el punto p_h.

Pseudocódigo

El siguiente pseudocódigo fue tomado de wikipedia.

```
jarvis(S)
  // S is the set of points
  // P will be the set of points which form the convex hull. Final set size is
i.
  pointOnHull = leftmost point in S // which is guaranteed to be part of the
CH(S)
  i = 0
  repeat
    P[i] = pointOnHull
    endpoint = S[0]      // initial endpoint for a candidate edge on the hull
    for j from 1 to |S|-1
      // endpoint == pointOnHull is a rare case and can happen only when j == 1
and a better endpoint has not yet been set for the loop
      if (endpoint == pointOnHull) or (S[j] is on left of line from P[i] to
endpoint)
          endpoint = S[j]   // found greater left turn, update endpoint
    i = i+1
    pointOnHull = endpoint
  until endpoint == P[0]      // wrapped around to first hull point
```

Scan de Graham

Publicado en 1972 por por Ronald Graham, es un algoritmo con complejidad $O(n\ log\ n)$ siendo n el total de puntos en el set. En un primer paso, este algoritmo encuentra el punto,$pivot$, cuya coordenada en el eje y es la menor. En caso de existir más de un punto con esa característica se toma el que tenga menor valor en su coordenada x. Esta operación tiene una complejidad $O(n)$.

En seguida los puntos en el set deben ser ordenados de forma creciente con respecto al ángulo formado por cada punto y el punto $pivot$ con respecto al eje x.

El algoritmo continúa considerando cada uno de los puntos ordenados en secuencia. Para cada punto, se determina si yendo de los dos puntos precedentes inmediatos a este punto representa girar a la izquierda o a la derecha. Si se gira a la derecha, la línea formada del segundo punto al punto final no forma parte de la envolvente convexa. La misma decisión es hecha para el set de el último punto y los dos puntos que preceden inmediatamente el último punto descartado. Este proceso se repite hasta encontrar un punto que genere un giro a la izquierda, el algoritmo avanza al próximo punto en el set de puntos ordenados menos todos los puntos que hayan sido descartados.

Pseudocódigo

El siguiente pseudocódigo fue tomado de wikipedia.

```
let points be the list of points
let stack = empty_stack()

find the lowest y-coordinate and leftmost point, called P0
sort points by polar angle with P0, if several points have the same polar angle
then only keep the farthest

for point in points:
    # pop the last point from the stack if we turn clockwise to reach this point
    while count stack > 1 and ccw(next_to_top(stack), top(stack), point) < 0:
        pop stack
    push point to stack
end
```

Resultados

Marcha de Jarvis

El algoritmo de la Marcha de Jarvis es presentado creando un set de n puntos creados de forma aleatoria en en espacio en dos dimensiones en los intervalos $[-10, 10]$ tanto para el eje x y el eje y. El número n es determinado por el usuario.

Se presentan los resultados para una ejecución con $n = 10$.

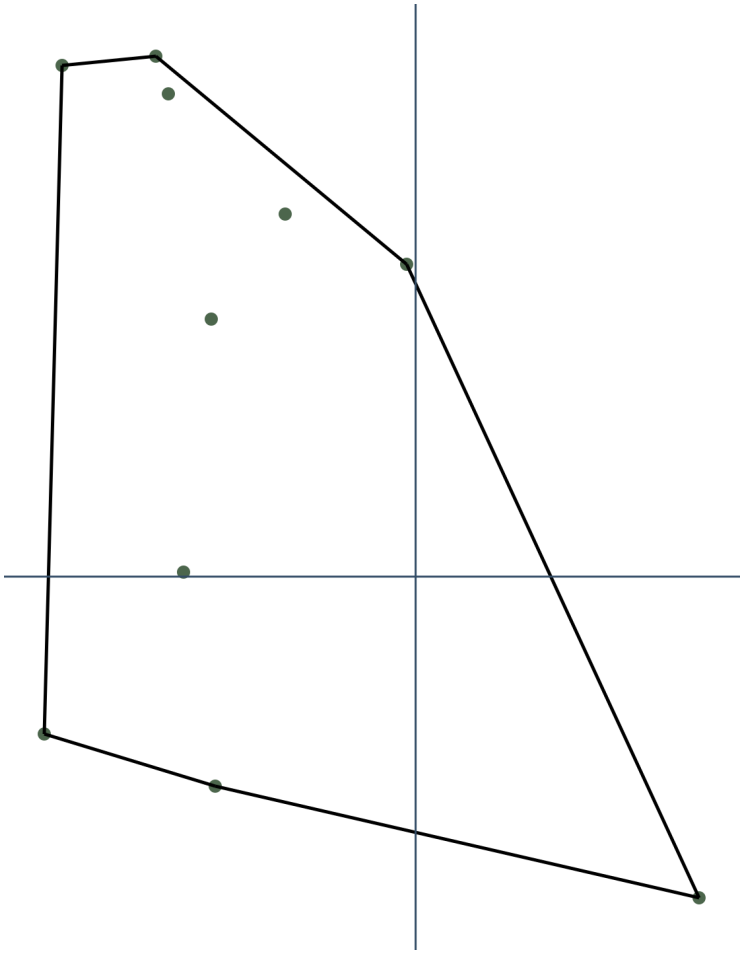

Figura 2. Marcha de Jarvis con n = 10. Las líneas perpendiculares representan los ejes coordenados.

El tiempo para esta ejecución fue de 0.109 milisegundos teniendo 6 vértices en la envolvente convexa siendo estos: $(5.38, -6.07)$, $(-0.16, 5.91)$, $(-4.91, 9.85)$, $(-6.69, 9.68)$, $(-7.02, -2.97)$, $(-3.79, -3.96)$.

Se muestran los resultados para $n = 30817$ con el objetivo de realizar una comparativa con el algoritmo del Scan de Graham que utiliza esta cantidad de puntos.

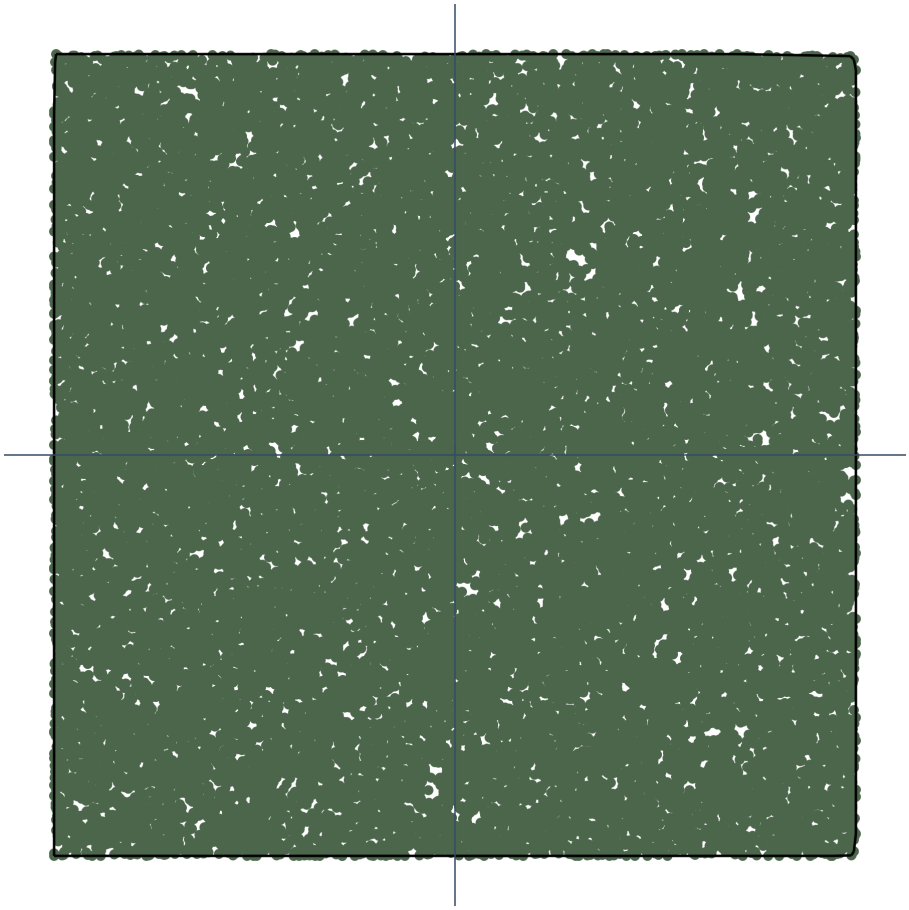

Figura 3. Marcha de Jarvis con n = 30817. Las líneas perpendiculares representan los ejes coordenados.

El tiempo de ejecución fue de 113.551 milisegundos, presentando 40 puntos en la envolvente convexa mismos que se pueden apreciar en el archivo *Convex Hull Points.txt* en la carpeta *Jarvis March*.

Scan de Graham

El algoritmo del Scan de Graham se realiza teniendo como entrada una imagen binaria *pgm* en la que los pixeles en blanco son los puntos que se quieren encerrar en la envolvente convexa. Para la imagen de entrada que se presenta en la figura 4 se obtiene como resultados la figura 5 donde las líneas en rojo representan el polígono de la envolvente convexa y los puntos en verde son los puntos pertenecientes a la envolvente.

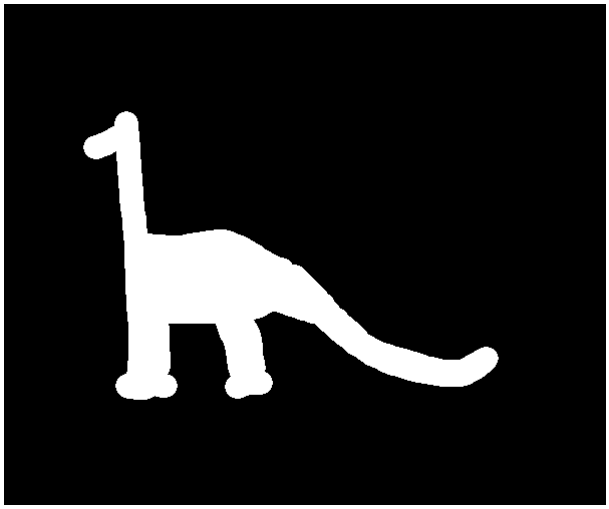

Figura 4. Figura utilizada para el Scan de Graham.

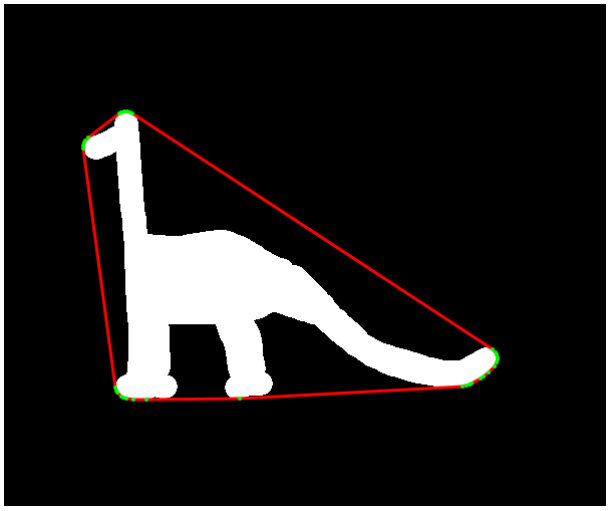

Figura 5. Envolvente convexa usando el Scan de Graham.

El tiempo en la ejecución es de 3.116 milisegundos. Presentando 34 puntos en la envolvente, mismos que se pueden visualizar en el archivo *Convex Hull Points.txt* en la carpeta *Graham*.

Comparación

Se puede observar que el Scan de Graham es aproximadamente 35 veces más rápido que la Marcha de Jarvis. Esto es debido a que el Scan de Graham no necesita analizar todos los puntos cada vez que que se encuentra un punto que pertenece a la envolvente conexa, tal como lo hace la Marcha de Jarvis. La complejidad en el Scan de Graham recae en el ordenamiento de los puntos.

Observaciones

Al realizar el algoritmo de la Marcha de Jarvis se logró solucionar el problema que se tenía en la tarea 11 en la que, al graficar con Cairo se obtenía una imagen con un efecto espejo con respecto a la imagen que se deseaba obtener.

Uno de los pasos esenciales en el Scan de Graham es encontrar el punto *pivot* para empezar a ordenar el resto de los puntos, esta búsqueda no fue necesaria en esta implementación ya que, de la forma en que se leen los pixeles en la imagen se sabe que el primer pixel considerado para pertenecer al se de puntos de interés es el *pivot* siendo este el que tiene menor valor en la coordenada y.