ANÁLISIS

Se tiene un conjunto de n puntos $C=(p_1, p_2, \ldots, p_n)$, donde cada punto p_i viene identificado por sus coordenadas $p_i=(x_i, y_i)$.

Se desea encontrar un subconjunto de C que definan un polígono convexo cuyo número de vértices sea mínimo, y que además contenga en su interior a todos los demás puntos de C.

DISEÑO DEL ALGORITMO

La **idea general** de un posible método para resolver el problema tendría que seleccionar todos los puntos más externos del conjunto C. Para ello, podríamos comenzar por un punto que sepamos que es solución al problema, e ir incorporando a esta solución aquellos puntos que sean los más externos del conjunto. Por ejemplo, se podría comenzar por el primer punto p_i inferior (mínimo valor en coordenada y) y, entre todos los puntos que cumplan esta condición, seleccionar aquel punto que se encuentre más a la izquierda (mínimo valor en coordenada x). Así evitamos coger dos puntos en la misma horizontal cuando haya dos o más puntos con la misma coordenada y pero distinta coordenada x.

A continuación, deberíamos seleccionar algún otro punto *candidato* p_c que añadir a la solución. Podríamos escoger aquel punto que, trazando una recta desde el punto inicial seleccionado al *candidato*, todos los puntos del conjunto p_q queden a la izquierda del vector (p_i, p_c). Saber si un punto p_q queda a la izquierda o a la derecha del vector se puede realizar calculando el determinante de:

$$A=\begin{pmatrix} x_i & y_i & 1 \\ x_c & y_c & 1 \\ x_q & y_q & 1 \end{pmatrix}$$

Si $|A|<0$, el punto p_q se encuentra a la "derecha" del vector director. En caso de que $|A|> 0$, entonces se encuentra a la "izquierda". Finalmente, si $|A|= 0$ entonces el punto se encuentra en la misma recta. Por tanto, vamos a suponer que existe ya un procedimiento al que llamaremos **PosicionVectorDirector(pi, pc, pq)**, que devuelva si el punto pq queda a la izquierda del vector trazado de pi a pc, a su derecha o en el centro (la misma recta).

Para el caso de que exista algún punto p_q que quede exactamente en la misma recta definida por (p_i, p_c), entonces tendríamos que tanto p_q como p_c son puntos externos, y posibles candidatos a formar parte de la solución. Como queremos que el polígono tenga el mínimo número de vértices, sólo escogeríamos p_c si la norma del vector $||(p_i, p_c)||$ es más grande que la norma del vector $||(p_i, p_q)||$; es decir, si p_c está más lejos de p_i de lo que lo está p_q. Ya tendríamos el candidato seleccionado, y repetiríamos la misma operación tomando a p_c como punto inicial, hasta que el siguiente punto candidato a añadir fuese el mismo punto inicial que se añadió a la solución.

El diseño del algoritmo y sus componentes sería el siguiente:

- **Función objetivo**: Encontrar los puntos que forman un polígono convexo cuyo número de vértices es mínimo y que forman la envolvente convexa del conjunto de puntos C inicial.
- **Lista de candidatos:** Los puntos del conjunto inicial C.
- **Lista de candidatos utilizados:** Los puntos que ya han sido insertados en la solución.

- **Criterio de selección**: Sabiendo que tenemos un punto actual p_i, seleccionaremos como candidato aquel punto p_c que hace que todos los demás puntos del conjunto queden a la izquierda del vector director (p_i, p_c), y cuya norma $||(p_i, p_c)||$ sea máxima entre todos los puntos que cumplan esta condición.
- **Criterio de factibilidad**: El candidato seleccionado p_c se insertará en la solución siempre que sea uno de los puntos más externos del conjunto (todos los demás puntos del conjunto queden a la izquierda del vector director (p_i, p_c)) que no haya sido seleccionado previamente. Esto se cumple siempre salvo cuando ya hemos encontrado la solución, en cuyo caso p_c es el primer punto seleccionado al comienzo del algoritmo.
- **Criterio de solución**: El algoritmo parará cuando se escoja un candidato p_c que sea igual al primer punto de la solución del que se partió originalmente.

Con este diseño, adaptamos la plantilla Greedy para resolver el problema de la siguiente forma, donde S es la solución y S[0], S[1], …., S[k-1] son los puntos insertados en la solución, en el orden en el que fueron seleccionados:

ALGORITMO S= Greedy(C : Conjunto de puntos)

p_0 = Seleccionar el punto de C con menor valor de componente x, entre todos los puntos que tienen mínimo valor de componente y

```
S= {p0} // Insertar p0 en la solución
k ← 1
Repetir:
    NormaPiPc ← 0
    Para cada punto pi en C:
        pactual ← pi
        EsCandidato ← Verdadero
        Para cada punto pj en C:
            Si PosicionVectorDirector(S[k-1], pactual, pj) = Derecha :
                EsCandidato ← Falso
        Fin-Para
        Si EsCandidato y ||(pi, pactual)|| > NormaPiPc :
            NormaPiPc= ||(pi, pactual)||
            pc= pactual
        Fin-Si
    Fin-Para

    Si pc es distinto de p0 :
        Insertar pc en S
        k ← k+1
    Fin-Si
Mientras pc sea distinto de p0
Devolver S
```

DETALLES DE IMPLEMENTACIÓN

En la solución planteada en el código fuente:

- Un punto se representa con una estructura Punto conteniendo dos componentes x e y.

- El conjunto de puntos C={p_0 … p_n} se representa con un array.
- La solución se representa como array de índices de los puntos originales.
- Se implementa una función auxiliar `PosicionVectorDirector(const Punto &pi, const Punto &pj, const Punto &pq)` que indica la posición del punto pq con respecto al vector director (pi, pj). Indicará si esta posición es "derecha" (valor 1), en la misma recta (valor 0) o a la "izquierda" (valor -1).
- El criterio de selección se implementa en una función adicional `BuscarPunto(const Punto *puntos, const int ini, const int fin, const Punto pi)`, que devuelve el índice c del array de puntos "puntos", indexado de ini a fin, que cumple con que (pi, pc) deja a la izquierda a todos los demás puntos del conjunto y que, además, tiene ||(pi, pc)|| máxima.

Existen dos ficheros en la implementación:

- **GenerarPuntos.cpp:** Contiene código para generar un número de puntos 2D (dado como parámetro al programa), con coordenadas (x,y) por cada punto. El número de puntos y los puntos en sí se proporcionan por salida estándar. Un ejemplo de su uso:

  ./GenerarPuntos 100 > dataset100.txt

  Genera un fichero dataset100.txt con un conjunto de 100 puntos aleatorios.

- **Solucion.cpp**: Contiene:
  - Código para leer un conjunto de datos en el formato especificado por GenerarPuntos (función **LeerDatos**).
  - Código para calcular la posición de un punto pq con respecto al vector director dado por otro par de puntos (pi, pj) (Función **PosicionVectorDirector**).
  - Código para implementar el criterio de selección (Función **BuscarPunto**).
  - Código para implementar el algoritmo Greedy (Función **Greedy**).

  Adicionalmente, el programa principal realiza el siguiente flujo de acciones:
  1. Leer datos del conjunto de datos por consola (se guarda en variable **cPuntos**, de tamaño **n**).
  2. Ejecutar el algoritmo greedy para encontrar el conjunto de puntos no dominados, midiendo también su tiempo de ejecución.
  3. Mostrar por consola los resultados obtenidos, así como el tiempo de ejecució.

  Un ejemplo de uso de este programa sería:

  ./Solucion < dataset100.txt