ANÁLISIS

Se tiene un conjunto de n puntos $C=(p_1, p_2, \ldots, p_n)$, donde cada punto tiene K coordenadas pi=($v_i[0]$, $v_i[1]$, …, $v_i[K-1]$). Por cuestiones de diseño, el conjunto de puntos se notará desde un punto inicial *ini* hasta un punto final *fin*, $C=(p_{ini}, p_{ini+1}, \ldots, p_{fin})$.

Se desea encontrar un subconjunto de C con los puntos no dominados. Denominaremos a una función **EsDominado(p_j, p_i)** que nos indique con valores verdadero/falso si el punto i es dominado por el punto j o no.

DISEÑO DEL ALGORITMO BÁSICO

La idea general del algoritmo sería comparar todos los puntos (p_i) con todos los demás (p_j) y comprobar si se cumple para al menos un p_j que EsDominado(p_j, p_i). Entonces p_i es dominado. Si no existe ningún p_j que haga que **EsDominado**(p_j, p_i) sea cierto, entonces p_i es no dominado.

El diseño de este algoritmo sería:

ALGORITMO S= BASICO($C=(p_{ini}, p_{ini+1}, \ldots, p_{fin})$: Conjunto de puntos):

```
    S ← Vacío
    Para cada punto pi, i=ini...fin:
        i_dominado ← falso
        Para cada punto pj, j=ini...fin:
            Si i != j :
                Si EsDominado(pj, pi):
                    marcar i_dominado ← cierto
        Fin-Para
        Si i_dominado es falso:
            Añadir pi a S
        Fin-Si
    Fin-Para
    Devolver S
```

DISEÑO DEL ALGORITMO DyV

- **División del problema en subproblemas.** El problema inicial $P=C=(p_1, p_2, \ldots, p_n)$ se divide en k=2 subproblemas P1, P2. El subproblema P1 sería $P1=(p_1, p_2, \ldots, p_{floor(n/2)})$ y el subproblema P2 sería $P2=(p_{floor(n/2)+1}, p_{floor(n/2)+2}, \ldots, p_n)$, de modo que ambos subproblemas P1 y P2 tendrían la misma naturaleza que el problema original P, tendrían aproximadamente el mismo tamaño n/2, y serían independientes entre sí y podrían resolverse por separado, dando lugar a las subsoluciones S1 de P1 y S2 de P2.
- **Caso base**: El caso base se dará cuando n<=2. En tal caso, se resuelve con el algoritmo básico.
- **Combinación de subsoluciones:** S1 es el conjunto de puntos no dominados de P1, y S2 es el conjunto de puntos no dominados de S2. La idea general para calcular la solución S al

problema P original consiste en comparar los puntos no dominados de S1 con los de S2 y viceversa, y obtener los puntos no dominados de ambos conjuntos.

El diseño del algoritmo DyV sería:

ALGORITMO S= DyV(C=(p_{ini}, p_{ini+1}, …, p_{fin}) : Conjunto de puntos):

```
    Si n<=2:
        Devolver S= BASICO( C )
    En otro caso:
        Dividir el conjunto de puntos C en:
            centro= floor((ini+fin)/2)
            P1=(p_ini, p_ini+1, …, p_centro-1)
            P2=(p_centro, p_centro+1, …, p_fin)
        S1= DyV(P1)
        S2= DyV(P2)
        S= COMBINAR(S1, S2)
        Devolver S
    Fin-En otro caso
```

El método combinar sería:

ALGORITMO S=COMBINAR(S1, S2)

```
    S ← vacío
    Para cada punto p_i en S1:
        i_dominado ← falso
        Para cada punto p_j en S2:
            Si EsDominado(p_j, p_i):
                marcar i_dominado ← verdadero
        Fin-Para
        Si i_dominado es falso:
            S ← S U {p_i}
        Fin-Si
    Fin-Para

    Para cada punto p_i en S2:
        i_dominado ← falso
        Para cada punto p_j en S1:
            Si EsDominado(p_j, p_i):
                marcar i_dominado ← verdadero
        Fin-Para
        Si i_dominado es falso:
            S ← S U {p_i}
        Fin-Si
    Fin-Para
```

DETALLES DE IMPLEMENTACIÓN

En la solución planteada en el código fuente:

- Dado que K se conoce a priori, se establece como constante global dentro de la solución.
- Un punto se representa con una estructura Punto conteniendo un array v de K valores.
- El conjunto de puntos C={c_{ini} … c_{fin-1}} Se representa con un array indexado de ini a fin-1 inclusive.
- Las soluciones se representan como arrays de índices de los puntos originales.
- Para generar los dos subproblemas P1 y P2 del algoritmo DyV para resolver el problema P no se crean dos nuevos arrays de puntos, sino que se reutilizan los puntos en P y se indexan P1=(p_1, p_2, …, $p_{centro-1}$) y P2=(p_{centro}, $p_{centro+1}$, …, p_{fin}), donde centro= floor((ini+fin)/2) por motivos de eficiencia.

Existen dos ficheros en la implementación:

- **GenerarPuntos.cpp:** Contiene código para generar un número de puntos (dado como parámetro al programa) con K=2 dimensiones. 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 dominancia entre dos puntos $p_2 > p_1$ (Función **EsDominado**).
  - Código para implementar el algoritmo básico (Función **BASICO**).
  - Código para implementar el algoritmo DyV (Función **DyV**).
  - Código para implementar la función de combinación de DyV (Función **fusionar**).

  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 básico para encontrar el conjunto de puntos no dominados, midiendo también su tiempo de ejecución.
  3. Ejecutar el algoritmo DyV para encontrar el conjunto de puntos no dominados, midiendo también su tiempo de ejecución.
  4. Mostrar por consola los resultados obtenidos por **BASICO** y **DyV**, así como sus tiempos de ejecución.

  Un ejemplo de uso de este programa sería:

  ./Solucion < dataset100.txt