***PRE:***

***POST:***

***^***

***tpSolape calcularSolape(tpInter intervalo1, tpInter intervalo2) { ... }***

***PRE:***

***POST:***

***tpSolape maxSolFBruta(double inters[N][2], int n){ ... }***

***PRE:***

***POST:***

***void crearvind(double inters[N][2], tpInter indinters[N], int n){ ... }***

***PRE:***

***POST:***

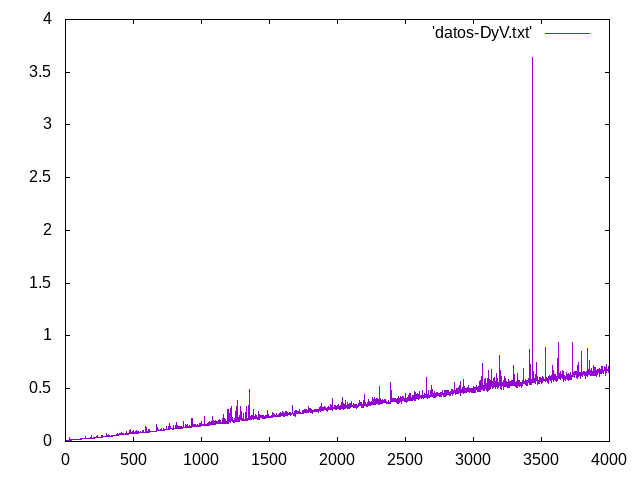
***void mergesortIndInters(tpInter indinters[N], int p, int f) ){ ... }***

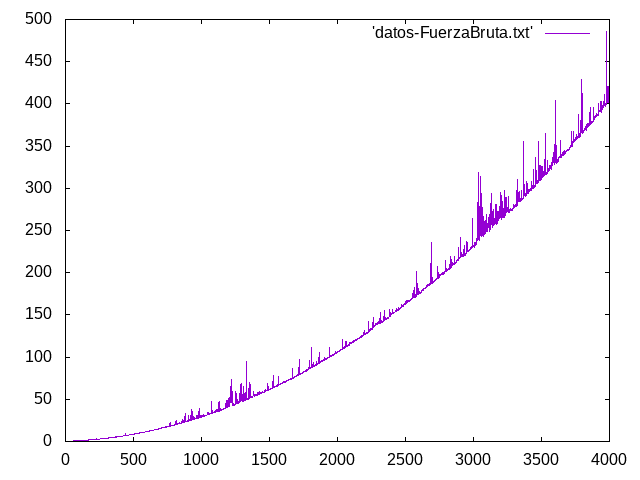
***PRE:***

***POST:***

***tpSolape maxSolDyV(tpInter indinters[N], int p, int f){ ... }***

***CONCLUSIONES:***

******Como se puede apreciar, el algoritmo de fuerza bruta se basa en el análisis de todos los casos posibles produciendo un incremento en el coste de tiempo a la vez que el tamaño del conjunto aumenta de forma exponencial.

******En cambio, el algoritmo de divide y vencerás, se basa en la división del problema principal en subproblemas más sencillos. Esta división en una cantidad ‘n’ de subproblemas y a su vez estos de una complejidad menor produce que al resolver un problema más simple este tenga un coste inferior al inicial. Este método hace que el tiempo se reduzca en gran parte ya que, aunque para los casos base el tiempo seguirá siendo constante(O(1)) este algoritmo tiene una cota superior asintótica de O(n·log(n)), por lo que se reduce su complejidad y producirá en este caso un coste temporal inferior al algoritmo de fuerza bruta que se caracteriza por un coste temporal de O(n2).