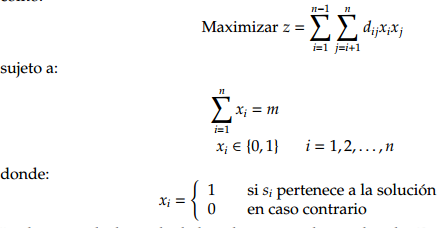
# Max Diversity problem.

***Práctica desarrollada por Iván García Campos.***

## Introducción.

Sea dado un conjunto S = {s1, s2, …, sn} de n elementos, en el que cada elemento si es un vector de tamaño k. Sea, asimismo, dij la distancia entre los elementos i y j. Si m < n es el tamaño del subconjunto que se busca el problema puede formularse como:

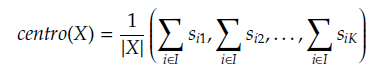


Se ha desarrollado:

1. Algoritmo Voraz Constructivo.
2. Algoritmo Voraz Destructivo.
3. Algoritmo Grasp con Búsqueda Local, eliminar un vértice, añadir uno de los vecinos.
4. Algoritmo Aleatorio con Búsqueda Local, eliminar un vértice, añadir uno de los vecinos.
5. Algoritmo Ramificación y poda, amplitud para m = 2.

## Algoritmo Voraz Constructivo. Apartado A.

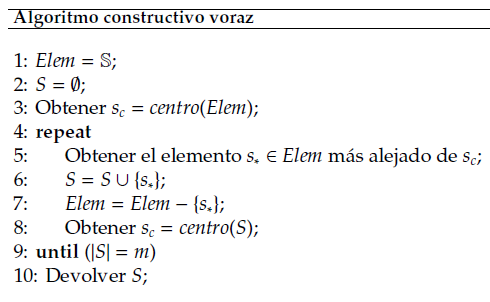
Se plantea la implementación de un algoritmo constructivo voraz partiendo de la solución vacía. Se calcula el centro de gravedad entre todos los nodos candidatos siguiendo la siguiente fórmula:



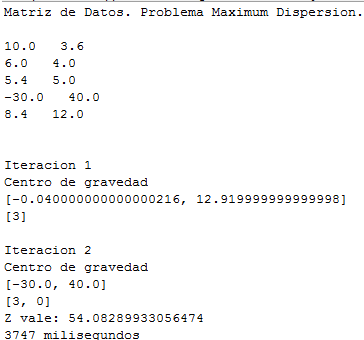
A continuación, se añade el vector con mayor distancia euclídea al centro de gravedad.

A partir de ahora, el centro de gravedad se calculará siempre sobre el nodo solución y se añadirá a la solución aquellos vectores con mayor distancia euclídea hasta tener una solución de tamaño m.

***Pseudocódigo.***



***Ejemplo sencillo de ejecución: m = 2***



Partimos de una solución cuyo centro de gravedad sea a partir de los vectores candidatos. El vector con distancia euclídea mayor es 3.

A partir de ahora, se calcula el centro de gravedad a partir de los vectores que estén en la solución, en este caso, mediante el vector 3. Buscamos el vector con mayor distancia euclidea. Se elige el vector 3.

Z vale 54.08.

## Algoritmo Grasp con Búsqueda Local Eliminar 1 Añadir 1.

El algoritmo diseñado sigue una estrategia similar al del punto 1. Se eligen iterativamente y partiendo de una solución vacía los LRC mejores candidatos para mejorar la diversidad.

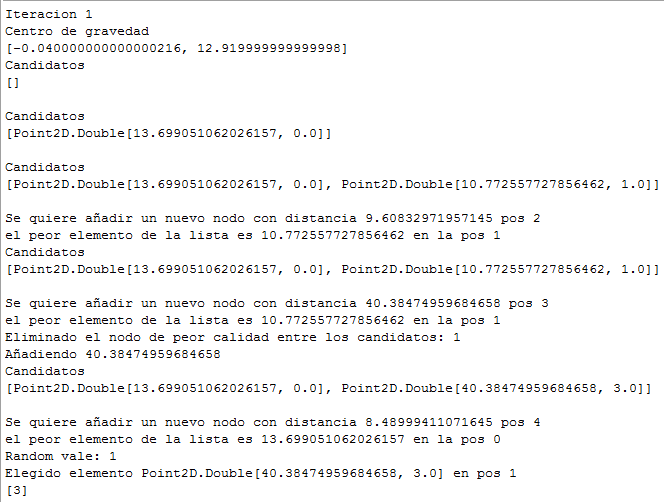
Se elige aleatoriamente una de ellas y se repite el proceso hasta m.

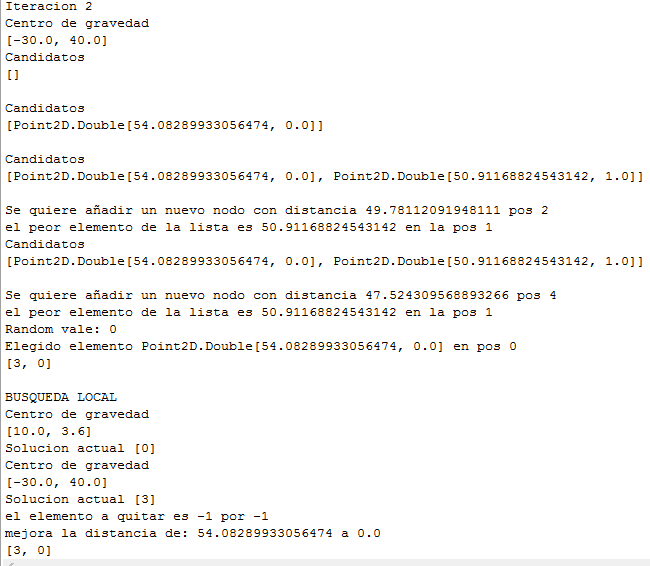
El centro de gravedad se calculara en la primera iteración a partir de los vectores candidatos (Todos los posibles). Cuando se haya elegido el primer vector de la solución, el centro de gravedad se calculará a partir de la solución. Igual que la estrategia voraz constructiva.

A continuación se aplicará una búsqueda local consistente en eliminar el vector que menor diversidad aporta y buscar entre los vectores que no están en la solución aquel que pueda mejorar la solución.

* Si se encontrase, se repite la búsqueda local sobre esta nueva solución.
* Si no se encontrase, termina el algoritmo.

***Ejemplo sencillo de ejecución: m = 2***





Se buscan los LRC mejores candidatos a introducir en la solución, en este caso el vector 3 y el 3. Se elige uno aleatoriamente, eligiendo el 3. Se repite la estrategia eligiendo el vector 0.

Se aplica una búsqueda local, que en este caso no mejora la solución actual.

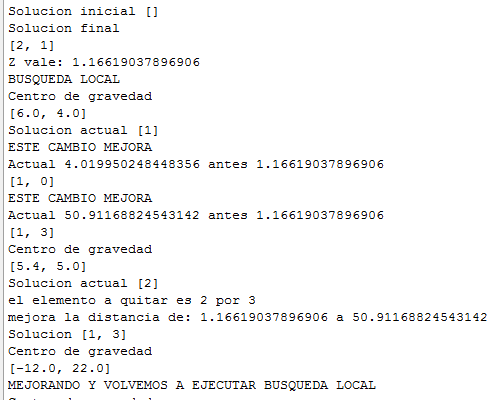
Solución final [0 3] con Z = 54.0829

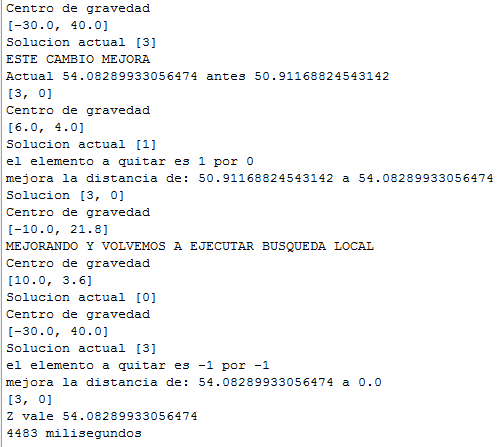
## Algoritmo Aleatorio con Búsqueda Local.

Se ha implementado un algoritmo que crea una solución aleatoria de tamaño m entre los nodos candidatos.

A esta solución, se le aplica una búsqueda local para encontrar posibles mejores soluciones. Si se encontrara, se aplica de nuevo la búsqueda local sobre esta nueva solución.

***Ejemplo sencillo de ejecución:***





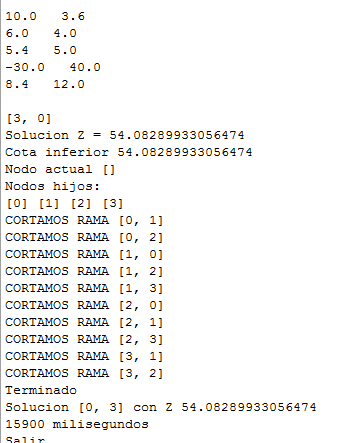
Se genera la solución aleatoria [2, 1] con Z = 1,16

Aplicando la búsqueda local, se intercambia el vector 2 por el 3 mejorando la dispersión a Z = 50,911. Se aplica de nuevo la búsqueda local intercambiando el vector 1 por el 0 y mejorando la dispersión a Z = 54,08. Se aplica de nuevo la búsqueda local sin éxito.

Solución final [0 3] con Z = 54.0829

## Algoritmo Ramificación y Poda.

***Ejemplo sencillo de ejecución: m = 2***



Partimos de una solución voraz con resultado Z = 54.08 y la guardamos como cota Inferior del árbol.

El nodo raíz del árbol contendrá la solución vacía y se irán rellenando los hijos con los vectores candidatos a introducir en la solución. 0, 1, 2, 3, 4. Si calcularCota de dicha solución < cotaInferior, dicha rama se podará y no se seguirá explorando. Si es mayor, se expanden los nodos hijos añadiendo otro posible candidato a la solución de la forma: [0 1] [0 2] [0 3] [0 4]

Se repetirá la estrategia hasta que se llegue a m en el tamaño de la solución.

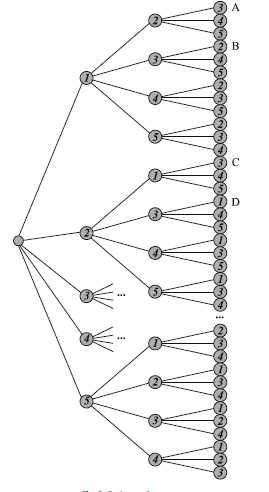
En este caso se han podado 10 ramas de las 20 posibles. Se elige la solución 0 3 como la que más aumenta la diversidad.

**NOTA.**

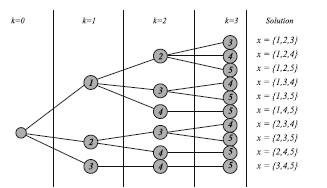
No se ha desarrollado correctamente el algoritmo calcular cotas que tiene la expresión:



Se ha logrado implementar un algoritmo parecido solo para m = 2. La generación del árbol de búsqueda expande todos los nodos de la forma



y no como se propone en la práctica:



## Tablas

**Algoritmo Voraz.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problema** | **M** | **Ejecución** | **Z** | **Subconjunto** | **CPU** |
| max-mean-div-15\_2.txt | 2 | 1 | 11.8592 | [8, 6] | 2,783 ms |
| max-mean-div-15\_2.txt | 3 | 2 | 25.7262 | [8, 6, 3] | 5,318 ms |
| max-mean-div-15\_2.txt | 4 | 3 | 48.4139 | [8, 6, 3, 10] | 3,278 ms |
| max-mean-div-15\_2.txt | 5 | 4 | 73.5619 | [8, 6, 3, 10, 1] | 5,862 ms |
| max-mean-div-15\_3.txt | 2 | 1 | 13.2732 | [11, 8] | 10,547 ms |
| max-mean-div-15\_3.txt | 3 | 2 | 30.3241 | [11, 8, 4] | 4,338 ms |
| max-mean-div-15\_3.txt | 4 | 3 | 59.7637 | [11, 8, 4, 10] | 4,652 ms |
| max-mean-div-15\_3.txt | 5 | 4 | 94.7487 | [11, 8, 4, 10, 13] | 4,016 ms |
| max-mean-div-20\_2.txt | 2 | 1 | 8.5103 | [17, 18] | 2,923 ms |
| max-mean-div-20\_2.txt | 3 | 2 | 21.9961 | [17, 18, 8] | 2,880 ms |
| max-mean-div-20\_2.txt | 4 | 3 | 39.5682 | [17, 18, 8, 2] | 3,535 ms |
| max-mean-div-20\_2.txt | 5 | 4 | 61.2393 | [17, 18, 8, 2, 12] | 3,063 ms |
| max-mean-div-20\_3.txt | 2 | 1 | 11.8003 | [12, 13] | 4,028 ms |
| max-mean-div-20\_3.txt | 3 | 2 | 30.8726 | [12, 13, 7] | 5,522 ms |
| max-mean-div-20\_3.txt | 4 | 3 | 56.53472 | [12, 13, 7, 2] | 2,871 ms |
| max-mean-div-20\_3.txt | 5 | 4 | 92.8297 | [12, 13, 7, 2, 16] | 3,703 ms |
| max-mean-div-30\_2.txt | 2 | 1 | 11.6571 | [8, 27] | 3,927 ms |
| max-mean-div-30\_2.txt | 3 | 2 | 28.9443 | [8, 27, 1] | 4,277 ms |
| max-mean-div-30\_2.txt | 4 | 3 | 52.7711 | [8, 27, 1, 10] | 5,340 ms |
| max-mean-div-30\_2.txt | 5 | 4 | 80.9102 | [8, 27, 1, 10, 12] | 2,938 ms |
| max-mean-div-30\_3.txt | 2 | 1 | 13.0737 | [16, 6] | 2,428 ms |
| max-mean-div-30\_3.txt | 3 | 2 | 33.8422 | [16, 6, 23] | 2,015 ms |
| max-mean-div-30\_3.txt | 4 | 3 | 63.5184 | [16, 6, 23, 13] | 3,164 ms |
| max-mean-div-30\_3.txt | 5 | 4 | 99.5088 | [16, 6, 23, 13, 14] | 2,858 ms |

**Algoritmo Voraz Destructivo**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problema** | **M** | **Ejecución** | **Z** | **Subconjunto** | **CPU** |
| max-mean-div-15\_2.txt | 2 | 1 | 11.8592 | [6, 8] | 3,998 ms |
| max-mean-div-15\_2.txt | 3 | 2 | 23.7964 | [1, 6, 8] | 3,593 ms |
| max-mean-div-15\_2.txt | 4 | 3 | 46.6528 | [1, 6, 8, 10] | 3,484 ms |
| max-mean-div-15\_2.txt | 5 | 4 | 73.5619 | [1, 3, 6, 8, 10] | 3,147 ms |
| max-mean-div-15\_3.txt | 2 | 1 | 13.2732 | [8, 11] | 3,779ms |
| max-mean-div-15\_3.txt | 3 | 2 | 30.3240 | [4, 8, 11] | 3,159 ms |
| max-mean-div-15\_3.txt | 4 | 3 | 58.7287 | [4, 8, 11, 13] | 4,966 ms |
| max-mean-div-15\_3.txt | 5 | 4 | 96.0858 | [3, 4, 8, 11, 13] | 3,519 ms |
| max-mean-div-20\_2.txt | 2 | 1 | 8.5103 | [17, 18] | 3,382 ms |
| max-mean-div-20\_2.txt | 3 | 2 | 21.9961 | [8, 17, 18] | 5,316 ms |
| max-mean-div-20\_2.txt | 4 | 3 | 39.5682 | [2, 8, 17, 18] | 1,972 ms |
| max-mean-div-20\_2.txt | 5 | 4 | 60.4301 | [2, 8, 10, 17, 18] | 1,733 ms |
| max-mean-div-20\_3.txt | 2 | 1 | 11.5909 | [2, 16] | 3,709 ms |
| max-mean-div-20\_3.txt | 3 | 2 | 29.1574 | [2, 12, 16] | 3,145 ms |
| max-mean-div-20\_3.txt | 4 | 3 | 56.6903 | [2, 12, 13, 16] | 2,295 ms |
| max-mean-div-20\_3.txt | 5 | 4 | 92.8297 | [2, 7, 12, 13, 16] | 2,052 ms |
| max-mean-div-30\_2.txt | 2 | 1 | 11.6571 | [8, 27] | 3,692 ms |
| max-mean-div-30\_2.txt | 3 | 2 | 28.9443 | [1, 8, 27] | 2,550 ms |
| max-mean-div-30\_2.txt | 4 | 3 | 52.7712 | [1, 8, 10, 27] | 2,644 ms |
| max-mean-div-30\_2.txt | 5 | 4 | 80.9102 | [1, 8, 10, 12, 27] | 2,330 ms |
| max-mean-div-30\_3.txt | 2 | 1 | 12.6137 | [5, 16] | 4,789 ms |
| max-mean-div-30\_3.txt | 3 | 2 | 34.2905 | [5, 16, 23] | 5,778 ms |
| max-mean-div-30\_3.txt | 4 | 3 | 63.7019 | [5, 13, 16, 23] | 2,809 ms |
| max-mean-div-30\_3.txt | 5 | 4 | 99.5920 | [5, 13, 14, 16, 23] | 2,960 ms |

**Algoritmo Grasp con búsqueda local eliminar uno, añadir uno.**

**LRC = 3.**

**Máximo de Búsquedas Locales = 10.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problema** | **M** | **Ejecución** | **Z** | **Subconjunto** | **CPU** |
| max-mean-div-15\_2.txt | 2 | 1 | 11.8592 | [6, 8] | 7,772 ms |
| max-mean-div-15\_2.txt | 3 | 2 | 27.3727 | [8, 6, 0] | 6,886 ms |
| max-mean-div-15\_2.txt | 4 | 3 | 49.8267 | [8, 6, 5, 0] | 8,069 ms |
| max-mean-div-15\_2.txt | 5 | 4 | 79.1295 | [8, 3, 6, 0, 5] | 9,623 ms |
| max-mean-div-15\_3.txt | 2 | 1 | 13.2732 | [11, 8] | 5,440 ms |
| max-mean-div-15\_3.txt | 3 | 2 | 31.8685 | [4, 11, 6] | 8,296 ms |
| max-mean-div-15\_3.txt | 4 | 3 | 59.7637 | [4, 10, 11, 8] | 4,770 ms |
| max-mean-div-15\_3.txt | 5 | 4 | 96.0858 | [11, 4, 8, 13, 3] | 7,644 ms |
| max-mean-div-20\_2.txt | 2 | 1 | 8.36900 | [2, 8] | 5,674 ms |
| max-mean-div-20\_2.txt | 3 | 2 | 21.9961 | [17, 8, 18] | 5,221 ms |
| max-mean-div-20\_2.txt | 4 | 3 | 40.0022 | [8, 18, 1, 2] | 7,403 ms |
| max-mean-div-20\_2.txt | 5 | 4 | 63.6516 | [18, 17, 8, 1, 13] | 9,107 ms |
| max-mean-div-20\_3.txt | 2 | 1 | 11.8003 | [12, 13] | 6,046 ms |
| max-mean-div-20\_3.txt | 3 | 2 | 30.8726 | [12, 13, 7] | 6,218 ms |
| max-mean-div-20\_3.txt | 4 | 3 | 56.6903 | [13, 12, 16, 2] | 5,230 ms |
| max-mean-div-20\_3.txt | 5 | 4 | 92.8297 | [13, 12, 16, 2, 7] | 9,688 ms |
| max-mean-div-30\_2.txt | 2 | 1 | 11.6571 | [27, 8] | 11,404 ms |
| max-mean-div-30\_2.txt | 3 | 2 | 28.9443 | [27, 8, 1] | 7,335 ms |
| max-mean-div-30\_2.txt | 4 | 3 | 52.7711 | [27, 1, 10, 8] | 9,343 ms |
| max-mean-div-30\_2.txt | 5 | 4 | 80.9102 | [27, 1, 10, 8, 12] | 9,942 ms |
| max-mean-div-30\_3.txt | 2 | 1 | 13.0737 | [16, 6] | 14,275 ms |
| max-mean-div-30\_3.txt | 3 | 2 | 34.2905 | [16, 23, 5] | 10,016 ms |
| max-mean-div-30\_3.txt | 4 | 3 | 63.7019 | [16, 23, 13, 5] | 11,561 ms |
| max-mean-div-30\_3.txt | 5 | 4 | 99.5920 | [23, 13, 16, 14, 5] | 12,599 ms |

**Algoritmo Búsqueda Local Sobre Algoritmo Aleatorio.**

**Máximo de Búsquedas Locales = 10.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problema** | **M** | **Ejecución** | **Z** | **Subconjunto** | **CPU** |
| max-mean-div-15\_2.txt | 2 | 1 | 11.8592 | [8, 6] | 9,098 ms |
| max-mean-div-15\_2.txt | 3 | 2 | 27.3727 | [8, 0, 6] | 3,826 ms |
| max-mean-div-15\_2.txt | 4 | 3 | 49.8266 | [5, 8, 6, 0] | 4,148 ms |
| max-mean-div-15\_2.txt | 5 | 4 | 79.1295 | [3, 6, 8, 5, 0] | 5,121 ms |
| max-mean-div-15\_3.txt | 2 | 1 | 12.1611 | [4, 10] | 6,505 ms |
| max-mean-div-15\_3.txt | 3 | 2 | 30.9867 | [8, 11, 3] | 5,163 ms |
| max-mean-div-15\_3.txt | 4 | 3 | 59.76376 | [11, 4, 8, 10] | 4,721 ms |
| max-mean-div-15\_3.txt | 5 | 4 | 96.0858 | [13, 3, 4, 8, 11] | 3,999 ms |
| max-mean-div-20\_2.txt | 2 | 1 | 8.5103 | [18, 17] | 4,174 ms |
| max-mean-div-20\_2.txt | 3 | 2 | 21.9961 | [8, 18, 17] | 5,746 ms |
| max-mean-div-20\_2.txt | 4 | 3 | 40.0022 | [1, 18, 8, 2] | 4,489 ms |
| max-mean-div-20\_2.txt | 5 | 4 | 63.6517 | [17, 1, 18, 13, 8] | 4,262 ms |
| max-mean-div-20\_3.txt | 2 | 1 | 11.5909 | [16, 2] | 4,262 ms |
| max-mean-div-20\_3.txt | 3 | 2 | 30.8726 | [13, 7, 12] | 4,723 ms |
| max-mean-div-20\_3.txt | 4 | 3 | 56.69031 | [2, 13, 12, 16] | 3,456 ms |
| max-mean-div-20\_3.txt | 5 | 4 | 92.8297 | [16, 2, 12, 13, 7] | 3,964 ms |
| max-mean-div-30\_2.txt | 2 | 1 | 11.6571 | [27, 8] | 3,054 ms |
| max-mean-div-30\_2.txt | 3 | 2 | 28.9443 | [1, 8, 27] | 3,579 ms |
| max-mean-div-30\_2.txt | 4 | 3 | 52.7712 | [27, 8, 1, 10] | 4,047 ms |
| max-mean-div-30\_2.txt | 5 | 4 | 80.9102 | [27, 1, 8, 10, 12] | 7,641 ms |
| max-mean-div-30\_3.txt | 2 | 1 | 13.0737 | [16, 6] | 4,819 ms |
| max-mean-div-30\_3.txt | 3 | 2 | 34.2905 | [16, 23, 5] | 4,009 ms |
| max-mean-div-30\_3.txt | 4 | 3 | 63.7019 | [5, 13, 16, 23] | 3,779 ms |
| max-mean-div-30\_3.txt | 5 | 4 | 98.5836 | [13, 16, 23, 3, 6] | 5,867 ms |

**Algoritmo Ramificación y Poda**

Aproximación Algoritmo Ramificación y Poda. M = 2.

La estrategia de poda es distinta a la planteada. En este caso, se decide la cota Inferior como la Z media entre todos los posibles nodos candidatos.

Ej:

1 -> Z = 10.3

2 -> Z = 9.4

3 -> Z = 8.4

4 -> Z = 40.2

5 -> Z = 11.3

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Problema** | **M** | **Ejecución** | **Z** | **Subconjunto** | **CPU** |
| max-mean-div-15\_2.txt | 2 | 1 | 11.8592 | [6, 8] | 5,038 ms |
| max-mean-div-15\_3.txt | 2 | 1 | 13.2732 | [8, 11] | 7,521 ms |
| max-mean-div-20\_2.txt | 2 | 1 | 8.5103 | [17, 18] | 6,463 ms |
| max-mean-div-20\_3.txt | 2 | 1 | 11.8003 | [12, 13] | 9,675 ms |
| max-mean-div-30\_2.txt | 2 | 1 | 11.6571 | [8, 27] | 10,354 ms |
| max-mean-div-30\_3.txt | 2 | 1 | 13.0737 | [6, 16] | 9,538 ms |