

# ESTUDIO VORACES

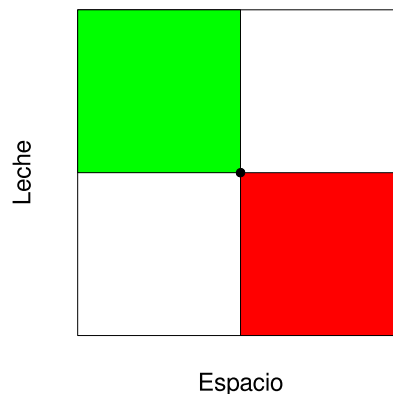
## OPTIMALIDAD DE LA SOLUCIÓN

En ambos casos, independientemente de a la respuesta que alcancemos, estamos sujetos a los problemas que trae el problema de la mochila donde no se puede fraccionar, ya que al ser un algoritmo voraz, si cumplimos las restricciones, tomaremos el elemento, independientemente de si ello nos lleva a la mejor solución o no. Es decir, para asegurar la solución óptima, tendríamos que ver todas las combinaciones para poder saber si existe una solución mejor.

## APARTADO A

El objetivo es maximizar la producción de leche teniendo como limitante el espacio. Es decir, queremos aquellas vacas que más leche den y menos espacio ocupen. Una primera idea sería seguir este criterio estrictamente.

Para verlo gráficamente, podríamos representar 2 ejes, Espacio y Leche donde cada punto sería una vaca:

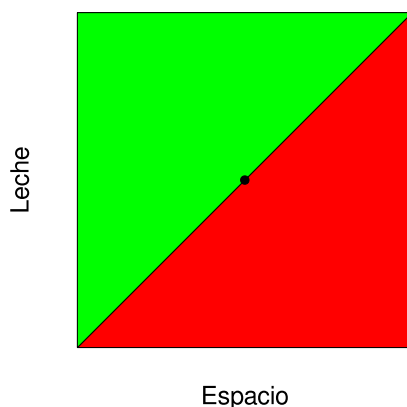


Dado el dibujo, el plano se divide en diferentes zonas respecto a cada vaca. Por encima de ella, las vacas que tienen una mayor producción de leche, debajo las que menos, a su izquierda, las que tienen menor consumo de espacio, y la derecha las que más. Resulta de especial interés hacer la intersección de estas regiones.

Este gráfico representa un diagrama de Venn. Además nos es de especial utilidad ya que ordena las vacas en el plano atendiendo a los dos parámetros a estudiar.

Nos quedaríamos con aquellas que están arriba a la izquierda, en la zona verde ya que son aquellas que cumplen ambas condiciones. El problema de esto es que tenemos 2 regiones, abajo izquierda y arriba derecha, las blancas, donde no sabríamos cuál sería mejor, ya que cumplen una condición, pero la otra no, estas vacas tendrían el mismo valor. Las de la zona roja serían aquellas que tenemos asegurado que son peores ya que no cumplen ninguna.

Para resolver el empate, podemos usar otra idea, calcular un ratio beneficio frente a coste. El beneficio sería la producción de leche y el coste, el espacio. La división quedaría:

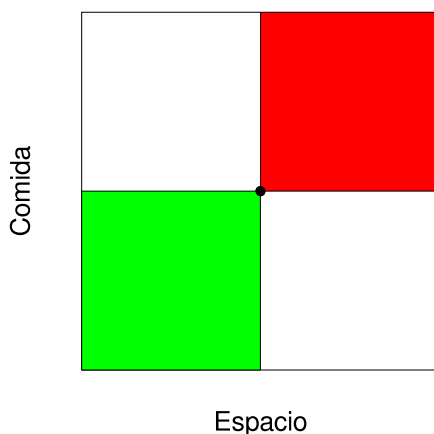


El ratio se puede interpretar como la pendiente de la recta entre el origen y la vaca. Como vemos, esto sí divide el plano en 2 regiones, además coincidentes con la primera división. Al dividir sólo en 2 zonas es más óptimo que el primer caso. En este caso, las vacas con el mismo valor son aquellas que están en la misma recta.

La pendiente de la recta es dependiente de cada vaca o valor asignado a esa vaca, se priorizan aquellas de mayor valor, es decir, las de mayor pendiente.

## APARTADO B

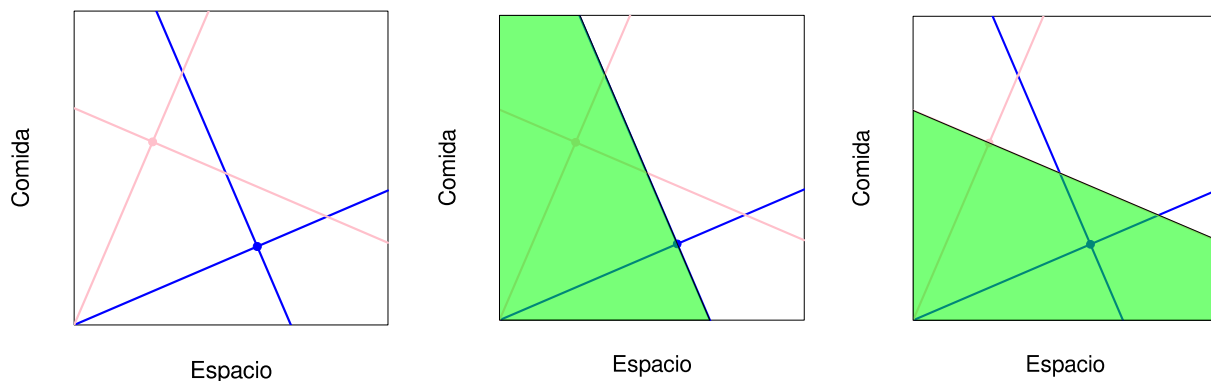
En este caso buscamos minimizar el consumo de comida con el limitante del espacio. Siguiendo la representación gráfica, ahora podemos representar las vacas en los ejes Comida y Espacio, donde si seguimos el criterio de elección dado, veremos un problema como en el primer caso:



Como vemos, ahora las regiones están reflejadas horizontalmente por lo que si hacemos el ratio nos equivocaríamos ya que podría darse el caso de coger vacas de la zona roja pensando que son mejores. Ahora no tenemos un beneficio, todo son gastos. Para arreglar la situación tenemos 2 posibilidades:

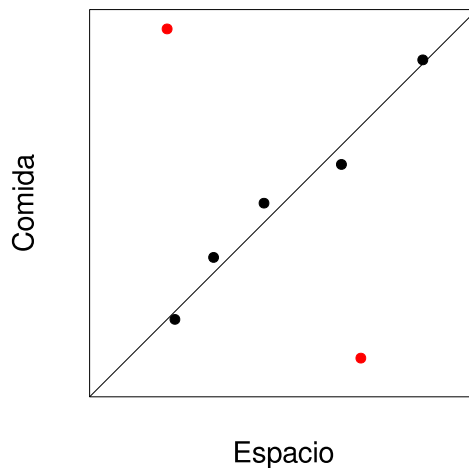
1. Buscar una recta.

Al igual que en el primer caso, podríamos buscar una recta que nos divida el espacio haciendo que coincidan las zonas como deben. Una posibilidad sería tomar la perpendicular por la vaca a la recta del primer caso y tomaríamos aquellas que estén abajo a la izquierda, como el dibujo de arriba dice, pero esto trae un problema:



Dado este caso, si tomamos como referencia la vaca azul, vemos que ella piensa que no tiene preferencia ya que la rosa está por debajo a la izquierda de su perpendicular (en la zona verde). Pero si tomamos como referencia la rosa vemos que ocurre lo mismo, hay ambigüedad.

Tendríamos que encontrar una recta que nos sirva para todas las vacas. Esto podría hacerse mediante una media de todas las vacas, pero esto trae otros problemas:

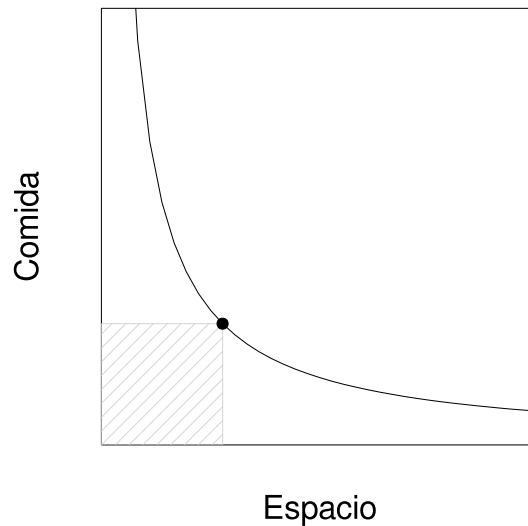


Cuanto más dispersas sean las vacas, peor será la aproximación que usemos. Además, esto hace que la ordenación de 2 vacas en concreto dependa de todo el resto de las vacas, cosa que no queremos. Necesitamos que la ordenación sea independiente y lo más completa posible.

No parece que podamos encontrar una recta que nos sirva. De todas formas, cabe mencionar que esta idea se podría interpretar como una suma pesada, donde el resultado es el coste, y los sumandos son pares  $\text{Parámetro} \times \text{Coste del Parámetro}$ , y los posibles parámetros son espacio y coste. La relación entre los costes de los parámetros sería la pendiente de la recta que pasa por el origen. Estaríamos intentando darle un mismo valor a ambos parámetros.

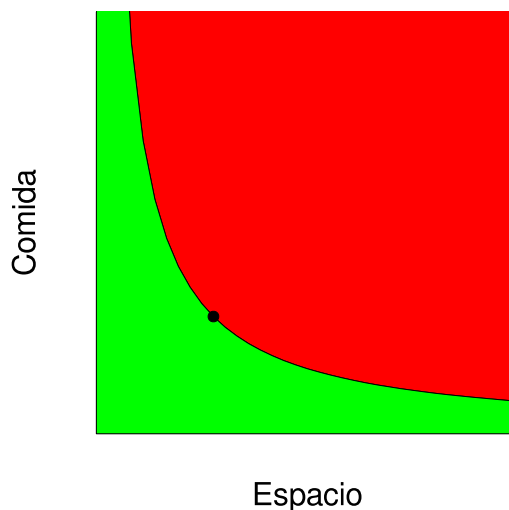
## 2. Buscar otra solución.

Tenemos que solucionar el desempate visto en la primera solución propuesta. La idea que exploraremos es usar áreas, calculadas como  $\text{Comida} \times \text{Espacio}$ , de esta forma obtenemos:



Con este parámetro resolvemos el empate. Aquellas vacas que tengan menor área se priorizarán. De esta forma, somos capaces de dividir el plano en dos regiones que contienen completamente a las iniciales y divide las ambigüedades. Además, esta división es independiente de otras vacas.

La línea representada divide el espacio al igual que antes de esta forma:



Aquellas vacas que coincidan en la línea tienen el mismo coste. Se escogen las que estén por debajo a la izquierda de la línea.

La línea depende de cada vaca o coste asignado a esa vaca, se priorizan aquellas de menor coste, es decir, las de menor área.

Volvemos a recalcar que aunque hayamos conseguido unos parámetros de ordenación que separan el plano en dos regiones haciendo que el número de vacas que se consideren iguales sean infinitamente menor ya que ahora es una curva en vez de dos planos, siguen pudiendo existir infinitas vacas iguales. Además, se pueden obtener diferentes criterios que sigan cumpliendo estas necesidades y podrían ser igual de buenos o mejores.

De todas formas, una ordenación, aunque se quede cerca, no es capaz de resolver todo el problema y deberíamos estudiar todas las posibilidades para comprobar si existen mejores soluciones.

## CÁLCULO DE COMPLEJIDAD

Bajo la suposición de que todos los métodos y operaciones que no se razonen tienen un coste constante.

Todos los métodos de la carpeta vaca tienen una complejidad constante  $O(1)$  ya que no cuentan con bucles.

El main cuenta con diversas partes: la llamada a la lectura de datos y las simulaciones.

La lectura del fichero de datos (Fichero.java) tiene un bucle que se repite tantas veces como líneas tenga el fichero. Complejidad  $O(n)$ , donde  $n$  es el número de líneas del fichero, que si no hay errores, coincide con el número de vacas disponibles.

Las simulaciones están representadas por un bucle for, tantas iteraciones como simulaciones queramos hacer, en este caso 2, una por cada apartado y atendiendo a un criterio de ordenación diferente. Para cada simulación se hace:

1. Ordenación de las vacas, la llamada a quickSortC, este algoritmo tiene una complejidad de  $O(n \times \log(n))$  en el caso promedio.
2. Ejecución de run, que tiene un bucle while que como mucho hará tantas iteraciones como vacas disponibles haya. Complejidad es  $O(n)$ , donde  $n$  es el número de vacas disponibles.
3. Muestra de resultados, el toString de la clase Solución, cuenta con un bucle que recorre todas las diferentes vacas seleccionadas. La complejidad es  $O(m)$ , donde  $m$  es el número de vacas seleccionadas, que en el peor de los casos es igual que el número de vacas disponibles, en otro caso, es menor.

Por tanto, en total se quedaría:

$$O(n) + 2 (O(n \times \log(n)) + O(n) + O(\min(n, m)))$$

Que se reduciría a:

$$O(n \times \log(n))$$

Esta complejidad viene por el algoritmo de ordenación.

## AUTORES

- Noelia Díaz-Alejo Alejo
- Samuel Espejo Gil