Pontificia Universidad Católica de Chile
Departamento de Ciencia de la Computación
Estructuras de Datos y Algoritmos — IIC2133

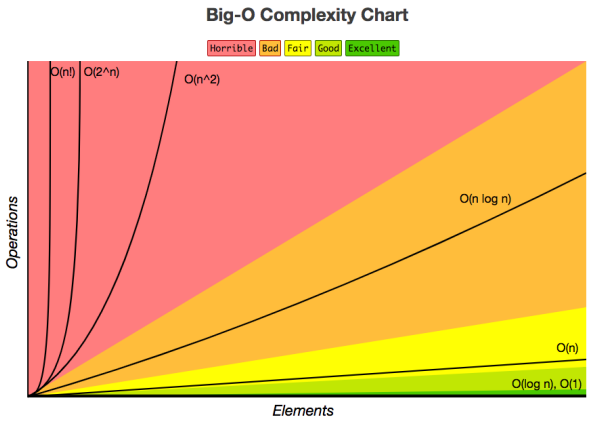
Tarea 2

13 de Agosto, 2018

Diego Iruretagoyena - 14619164

Introducción

En esta oportunidad, desarrollamos



Análisis

1. Análisis:

Debes mostrar a traves de graficos la complejidad de los distintos test con respecto a su tamano.

Especıficamente debes hacer un grafico donde el eje x corresponde a las dimensiones de la matriz (NM) y el eje y corresponde al tiempo total que tomo tu programa en resolverlo.

Tambien debes crear otro grafico con el mismo eje x, pero muestre en el eje y la cantidad

de veces que se deshace una asignacion (cantidad de veces que se ejecuta la lınea 8 del algoritmo).

Junto con los graficos debes comentar la complejidad que crees que tiene el algoritmo viendo los datos.

Optimizaciones

A medida que vamos conociendo el proceso, nos damos cuenta que hay veces que el algortimo podria evitar seguir un camino, dadas ciertas condiciones. Por ejemplo, puedo evitar pensar en opciones si

analisis de este tipo nos permiten acortar los tiempos de ejecución, ya que nos hacen no gastar recursos innecesariamente en opciones que no nos serán utiles. Existen varias formas de realizar estos ajustes al comportamiento de nuestro programa, siendo las principales las **podas**, las **heuristicas** y las **propagaciones**. A continuación explicaremos que es cada una de esas, pra luego dar ejemplos concretos de como podemos aplicarlo a nuestro problema.

explicando brevemente la forma de implementarlo y por que se espera que esto mejore la eficiencia del algoritmo.

1. Poda

Podas: una poda busca descartar combinaciones que no llegaran a resolver el problema independiente de que no esten incumpliendo alguna restriccion actualmente, por lo que una poda para hacer mas eficiente el algoritmo es que si se esta cumpliendo una restriccion de una fila todos los espacios de imanes horizontales restantes que estan en esa fila seran vacıos y si se esta cumpliendo una restriccion de una columna todos los espacios de imanes verticales restantes que estan en esa columna seran vacıos. En los dos casos permite ahorrar tiempo en revisar todas esas celdas que de antemano se sabe que no se puede poner nada ya que no se seguira cumpliendo con la restriccion correspondiente. Por ejemplo, se esta haciendo el test 6 y se ha hecho la primera asignacion, quedando de la siguiente manera:

Se puede ver que en la segunda columna ya se cumplio con la restriccion de positivos, por lo que si se asigna cualquier iman vertical se dejara de cumplir con esta restriccion. Esto permite ahorrarse revisar tres espacios verticales de inmediato. Si esto se hace durante todo el proceso permitirıa evitarse buscar en una gran cantidad de cambinaciones, lo que lleva a una gran reduccion en el tiempo que demora el proceso.

2. Heurística

Heurıstica: para desarrollar el problema de manera mas efectiva, se puede cambiar el orden en que se va armando el tablero. En lugar de empezar recorriendo cada fila y cada columna dentro de esta (o al reves), se le puede dar una prioridad a cada celda y de acuerdo a esto ir armando el tablero. Una buena manera de establecer prioridades es de acuerdo a lo acotado que esta su dominio, lo que se puede establecer por medio de las restricciones que tiene. Para realizar esto se pueden sumar las restricciones que hay para cada celda, es decir, filas positivas y negativas y columnas positivas y negativas. Para esto no importa el signo de la restriccion, ya que la heurıstica solo busca que primero se vean celdas que probablemente no terminaran vacıas. Por ejemplo, en el test 4 la solucion es de la siguiente forma:

3. Propagacion y una manera de propagar en este problema

Al ir viendo las prioridades, se tiene que la celda (0,0) tiene prioridad 6, la celda (1,0) tambien (fila +3 y columna -3) y la (2,0) tiene prioridad 4. Luego de que se le da una prioridad a cada celda, se realiza el mismo procedimiento de antes pero se revisan las celdas con mayor prioridad primero, hasta llegar a las que tengan prioridad 0. Esta heurıstica permite evitar de cierta manera poner celdas que con mayor probabilidad se tengan que sacar despues ya que la solucion es con ese espacio vacıo y que no se tenga que volver hasta la raız al hacer el backtracking. Propagacion: una propagacion bastante simple si se realiza el codigo revisando por celdas es que si se pone una celda positiva, la celda que acompa˜na en el espacio que comparten debe ser negativa y si se pone una negativa la que la acompa˜na en el espacio que comparten debe ser positiva. Esta propagacion es bastante obvia, ya que serıa sumamente ineficiente no hacerlo de esta manera ya que se revisarıa una celda que de antemano se sabrıa el valor que tiene que tener para cumplir con la restriccion de que dos celdas aleda˜nas deben tener carga contraria en caso de no estar vacıas.