Instituto tecnológico de Costa Rica

Campus San Carlos

Unidad de computación

Análisis de algoritmos, grupo #50

Proyecto #1

“Análisis de complejidad algorítmica”

Estudiantes:

Porras Morera Fabricio

Solís Mora Carlos

Profesora:

Valerio Solís Lorena

Fecha de entrega:

1 de octubre del 2022

Segundo semestre 2022

Introducción

En muchas ocasiones cuando se está realizando un proyecto programado de cualquier tipo, se presentan una serie de requerimientos variados, de los cuales se necesita entregar una solución comúnmente realizada en cualquiera de la amplia gamma de lenguajes de programación que existen a día de hoy, sin embargo, cuando se piensa en realizar una solución que satisface las exigencias del proyecto como tal, lo más normal es solo enfocarse en finalizar el proyecto y no se piensa fuera de la caja, sin tomar en cuenta una gran variedad de posibilidades.

Debido al efecto "túnel de visión" que generan las fechas límite en todo tipo de proyectos, ya no tan solo los programados, la verdadera eficiencia de las distintas soluciones que se plantean a lo largo del desarrollo del proyecto pasa a un segundo plano y la entrega del contenido se torna en lo principal. Esto que en un principio puede no sonar tan comprometedor o problemático dependiendo de cuál situación o contexto es en el que se habla, en el caso de la programación puede llegar a ser algo realmente grave, y esto es debido a que un algoritmo ineficiente puede conllevar a que todo un proyecto se desmorone, cuando ciertas situaciones que no se consideraban ocurran.

Este tipo de problemas es fundamental evitarlos a toda costa, debido a que, en un futuro, cuando se realicen proyectos con empresas en un contexto de competitividad en el mercado laboral, la entrega de un proyecto defectuoso puede significar consecuencias realmente graves, como lo pueden ser grandes pérdidas económicas e incluso el despido muchas de las personas implicadas.

Es por estas razones, que existen diferentes procesos de contención que se realizan para prevenir que se pueda estar entregando un algoritmo programado defectuoso, procesos que se engloban en un término conocido como análisis de algoritmos, comúnmente se realizan 2 análisis principales, el análisis empírico y el analítico.

El conocido como el análisis empírico de un algoritmo, es en el cual se obtiene el número de todas las comparaciones y asignaciones que realiza conforme a diferentes tamaños de los datos de entrada, y con ellos, se comparan el número de asignaciones y comparaciones

obtenidos de algunos de los casos elegidos con otros; el factor talla es fundamental en este proceso para determinar de forma aproximada, el comportamiento que está presentando ese algoritmo específico y así determinar su complejidad.

El proceso analítico, es uno mucho más profundo, en el cual se analiza línea por línea el comportamiento que presentará el código a analizar; para ello se utilizan procesos algebraicos analizados a mano, y gracias a este proceso se llega a una respuesta mucho más exacta de la complejidad que realmente tiene ese algoritmo.

Por último está el análisis gráfico, en el cual se representa de forma visual el comportamiento del algoritmo analizado en el software de elección, de esta forma, se podría decir que se ha realizado un análisis del algoritmo mucho más completo, y se pueden sacar conclusiones mucho más exactas de si ese algoritmo es uno eficiente, que se puede utilizar en un amplio rango de situaciones sin que entorpezca un proyecto entero, o si es uno que puede llegar a generar problemas bajo ciertas situaciones, y por ende gracias a esto, que se pueda descartar y buscar una solución mucho más completa que no vaya a generar problemas tanto a los ingenieros, como a las empresas que los contratan.

Análisis del Problema

La parte programada de este proyecto se basa principalmente en la resolución de un tipo de un problema no definido, que se conoce como el problema del cubrimiento mínimo de vértices, o vertex cover problem por su nombre en inglés. Para entregar una correcta solución al problema, se necesita obtener la cantidad mínima de vértices que cubren todos los arcos existentes en un grafo cualquiera.

Para obtener este mínimo de vértices, existen diferentes caminos a seguir en el desarrollo de los algoritmos. En el caso de este proyecto se deben de realizar 3 algoritmos diferentes que lleguen a la misma respuesta, pero de manera diferente, para así tener una varianza en la cantidad de comparaciones y asignaciones que estos realicen en cada ejecución del programa y conforme al tamaño del gráfico, y, por lo tanto, que tengan un diferente grado de eficacia.

El primer algoritmo por realizar se le conoce como el algoritmo de heurística voraz, o por su nombre en inglés, el greedy cover, el cual funciona de forma que este va obteniendo los vértices más prometedores en base a la cantidad de arcos que tiene asociados y luego elimina tanto el vértice, como los arcos del grafo para no poder volver a seleccionarlo nuevamente y así continuar correctamente; este proceso se repite indefinidamente hasta que no quede ningún arco por cubrir y se imprime la solución encontrada durante el proceso.

El segundo algoritmo que se debe realizar para la solución del vertex cover problem, es el algoritmo conocido como el algoritmo de fuerza bruta, o brute force por su nombre en inglés; este algoritmo tiene un nombre bastante descriptivo, ya que fuerza la solución exacta al problema comprobando cada una de las combinaciones posibles para cada uno de los vértices existentes en el grafo. La idea del algoritmo de fuerza bruta es siempre llegar a una solución perfecta incluso si se tiene que comprobar hasta el último de las posibilidades existentes; aunque, evidentemente, debido a la naturaleza no definida del problema que se trata de solucionar, las soluciones pueden llegar a ser más o menos fáciles, según el camino que vara a tomar el algoritmo, por lo que pueden existir algunos casos en los que las

respuestas que este algoritmo entregue sean engañosas, y deban ser obviadas para obtener las que de verdad conforman lo normal dentro de su comportamiento.

Finalmente, el tercer algoritmo a analizar se deja a elección de los desarrolladores del proyecto, este tercer algoritmo, por supuesto, debe de tener características diferenciales que generen otras asignaciones y comparaciones a las del algoritmo de fuerza bruta y el algoritmo voraz, para que se pueda clasificar si este tercer algoritmo es peor o mejor que los otros dos.

Una vez realizadas todas las propuestas de cada uno de los algoritmos que se utilizarán, se deberán medir la cantidad de comparaciones y asignaciones que estos realizan en base a grafos de distinto tamaño, la cantidad de líneas ejecutadas y su tiempo de ejecución, esto con el fin de poder aplicar aplicarle a los algoritmos un análisis empírico, donde se compare su rendimiento en los distintos tamaños de grafos a los que se ve sometido, y así dar una aproximación general del comportamiento y la complejidad que este podría tener.

Seguidamente, se deberá aplicar una medición analítica para cada uno de los algoritmos propuestos, esto con el fin de generar un acercamiento mucho más exacto del comportamiento y complejidad que tienen cada uno de estos algoritmos, para así poder compararos entre sí, y llegar a diferentes conclusiones con respecto a su rendimiento bajo qué contextos o circunstancias y dar un veredicto.

Como último requerimiento del proyecto, está el aplicar una medición gráfica para cada uno de los algoritmos propuestos, tomando como datos de entrada únicamente las líneas ejecutadas (dato que se conforma por la suma de la cantidad de asignaciones y comparaciones), para que, de esta forma, se pueda comparar el comportamiento indicado en la medición empírica de dicho algoritmo, y se pueda determinar si las conjeturas iniciales basadas en la etapa empírica estaban bien encaminadas o no.

Solución del Problema

Última solución realizada.

La última solución realizada en el apartado programado del proyecto es el método de generación de los arcos o aristas de manera aleatoria llamado “aristasAleatorios”, este método utiliza las clases “Aristas”, y la clase “Random” generada a partir de la librería de java.util.Random. El método se encarga básicamente de generar arcos entre los distintos vértices que le son entregados, los cuáles también fueron generados previamente de forma automática; el método crea un arreglo de tipo “Arista” en la cual se van agregando todas las aristas generadas de forma aleatoria entre los vértices, con la ayuda de la clase “Random”, y finalmente retorna este arreglo para posteriormente utilizarlo en la creación del grafo que se analizará por lo distintos algoritmos elegidos.

Algoritmo de Fuerza Bruta.

El primer algoritmo realizado es el algoritmo de fuerza bruta, este se obtuvo mediante la utilización de un código sacado de internet (Boham, 2017). Este algoritmo utiliza las clases “Aristas”, “Vértice” y “Grafo” para su correcto funcionamiento; adicionalmente, utiliza los métodos de “Combinations” y el método “removeEdges”.

La lógica que utiliza este algoritmo es la de crear un arreglo de tipo “Vértice” donde se guardarán todos los vértices que se encontraron como solución, y otro arreglo de tipo “arraylist<Vértice)>” el cual se utilizará en el método de “Combinations”. La idea es la de tomar cada vértice de uno por uno, y calcular todas sus posibles combinaciones en el método “Combinations”, cada una de estas combinaciones es reservada, para que el siguiente vértice analizado pueda determinar si al combinarse con dichas combinaciones es capaz de cubrir todas las aristas del grafo, y de no ser así, estas combinaciones también son almacenadas para el siguiente vértice en la lista; el proceso se repite indefinidamente hasta que se encuentra con un vértice que se combine correctamente con todas las posibilidades propuestas, y finalmente, se retornan dichos vértices que al combinarse lograron la respuesta perfecta.

A este algoritmo simplemente se le fue eliminado una gran cantidad de elementos como creaciones de variables y arreglos que no estaban siendo utilizados, para el proceso de optimización.

Algoritmo de Fuerza Bruta Optimizado

En el caso del segundo algoritmo realizado, es el elegido por los desarrolladores del proyecto, fue encontrado en el mismo código hallado en internet (Boham, 2017) y es el algoritmo llamado algoritmo de fuerza bruta optimizado, o por su nombre en inglés, optimized brute force. Este algoritmo utiliza las clases "Aristas", "Vértice" y "Grafo" para su correcto funcionamiento; pero en este caso, solo utiliza el método de "removeEdges", ya que el método de "Combinations" es reemplazado, debido a que el algoritmo utiliza un método de inicialización llamado "optimizedBruteForce", que recibe los distintos elementos del grafo a analizar por parámetro, y otro con el mismo nombre pero que no recibe nada, que se encarga de hacer el proceso del antiguo método de "Combinations" de una forma un poco más eficiente.

Básicamente la lógica es la de utilizar un método que recibe los elementos a analizar, y que sirva como arranque para el segundo método que no recibe nada, y se encarga de realizar las combinaciones para cada vértice, sin utilizar ese tedioso proceso de guardar cada uno de las combinaciones de cada uno de los vértices en un arreglo de tipo "arraylist<Vértice)>"; este algoritmo resuelve el problema disminuyendo la carga que conlleva generar cada combinación, con cambios que parecen muy minúsculos, pero que al momento de generar el resultado de todas las asignaciones y comparaciones del algoritmo, resultan en una mejoría suficiente con respecto al método de fuerza bruta común.

Así como en el caso del algoritmo de fuerza bruta, al código se le fue eliminado una gran de cantidad de creación de variables y arreglos que no se estaban ocupando para el correcto funcionamiento del algoritmo y simplemente estorbaban.

Algoritmo de Heurística Voraz.

Este tercer y último algoritmo es el algorimo de heurística voraz, el cual como en el caso de los anteriores 2 algoritmos, fue encontrado en el mismo código encontrado en internet (Boham, 2017). De igual manera, este método utiliza las clases "Aristas", "Vértice" y

“Grafo” para su correcto funcionamiento; y tampoco utiliza el método “Combinations”, ya que en su lugar se incluye el método “obtenerVérticeMáximo” y “hasVertex”.

La lógica detrás es en concepto, mucho más simple que las 2 anteriores. Se genera un ciclo en que se llama al método “obtenerVérticeMáximo”, este se encarga de obtener el vértice con el mayor grado de aristas asociadas a él, el cual es inmediatamente agregado a un arreglo de tipo “Vértice” para guardarlo como una de las soluciones, luego se llama al método “removeEdges” para eliminar ese vértice y todos las aristas asociadas a él y se vuelve a entrar a un nuevo ciclo; este proceso se repite hasta que ya no existan aristas a las que se deban de cubrir, y que por lo tanto, se haya llegado a una solución satisfactoria de la cantidad mínima de vértices para cubrir todas las aristas existentes de un grafo específico de manera voraz.

Como en todos los casos, no existían realmente muchos cambios posibles para la optimización de los ya existentes códigos, más allá del hecho de eliminar las variables y arreglos que no estaban siendo utilizados por el algoritmo voraz, por lo que una vez hecho esto, se habría finalizado con las soluciones propuestas para la utilización en el proyecto, y estarían los códigos ya preparados para comenzar las mediciones de los diferentes requerimientos propuestos en el mismo ya mencionado.

Análisis de Resultados

A continuación, se realizará un análisis de los datos obtenidos en la etapa de medición a empírica, analítica y gráfica. Se llevará a cabo una comparación del rendimiento de los 3 algoritmos elegidos en todos los métodos mencionados, para poder así llegar diferentes conclusiones al respecto, y dar un reporte claro de los resultados obtenidos a lo largo del desarrollo de este proyecto.

Resultados Finales

Antes de entrar en detalle con el juicio de cada uno de los 3 algoritmos realizados en el proyecto, se pasará a realizar unas especificaciones generales de los resultados finalmente obtenidos. Primeramente, el proyecto se encuentra completado prácticamente en su totalidad, el código es capaz de crear grafos con vértices y aristas generados de forma totalmente aleatoria, y los 3 algoritmos realizados para realizar el proceso de cubrimiento de vértices de un grafo en específico cumplen con su función satisfactoriamente.

El código genera el conteo de comparaciones y asignaciones necesario para la realización de las diferentes mediciones requeridas correctamente, además, cada vez que se genera una ejecución del programa, una vez se inicializa cualquiera de los 3 algoritmos clave, se calcula el tiempo de ejecución que ese algoritmo en específico tardó en realizar el proceso de cubrimiento de vértices, y se realiza su cálculo con una precisión de 3 decimales para mayor exactitud.

Se podría decir que la creación de las aristas se encuentra en un estado un tanto defectuoso, puesto que su generación no se realiza de forma no direccionada, así que debido a que los requerimientos del trabajo dictaminan que esto era necesario para su completa realización, este aspecto en específico del programa es el factor que conlleva a que no se tenga la totalidad del proyecto; sin embargo, más allá de este aspecto, el código realizado cumple notablemente los requerimientos impuestos al principio del trabajo.

Por último, si se tuviera que mencionar un aspecto a mejorar del código obviando el apartado defectuoso ya mencionado, esto sería una tarea realmente complicada ya que en términos generales el algoritmo funciona a la perfección, en un principio, se tenían muchas variables y arreglos inicializados que no se necesitaban para que el funcionamiento del

algoritmo fuera el correcto, sin embargo luego de varias revisiones, este aspecto se optimizó completamente y ya no es un defecto del código, por lo que si se pudiera mejorar algo, quizás sería el poner un nuevo algoritmo de cubrimiento de vértices con una complejidad distinta a las que ya se tienen para tener una mayor variedad.

Mediciones Empíricas y Su Respectivo Factor de Crecimiento

Medición empírica

Nombre del algoritmo #1: Heurística Voraz

Operaciones	5 v 4 a	5 v 8 a	10 v 9 a	10 v 18 a	20 v 19 a	20 v 38 a	40 v 80 a	50 v 120 a	50 v 150 a
Asignaciones	229	498	1.096	2.314	4.997	12.272	68.889	186.264	228.794
Comparaciones	167	373	932	1.979	4.645	11.370	66.542	181.057	222.205
Cantidad de líneas ejecutadas	396	871	2.028	4.293	9.642	23.642	135.431	367.321	450.999
Tiempo de ejecución	0,001 mili	0,001 mili	0,001 mili	0,001 mili	0,001 mili	0,004 mili	0,008 mili	0,015 mili	0,034 mili
Cantidad de líneas del código	Nota = Todas las funciones pertenecientes a la clase Graph no serán contadas, pues todos los algoritmos los comparten. obtenerVerticeMaximo = 18 hasVertex = 6 greedyCover = 39 Total = 63								

Nombre del algoritmo #2: Fuerza bruta optimizado

Operaciones	5 v 4 a	5 v 8 a	10 v 9 a	10 v 18 a	20 v 19 a	20 v 38 a	40 v 80 a	50 v 120 a	50 v 150 a
Asignaciones	12.759	24.735	486.821.294	834.703.296	NULL	NULL	NULL	NULL	NULL
Comparaciones	8.892	17.180	334.358.402	528.706.898	NULL	NULL	NULL	NULL	NULL
Cantidad de líneas ejecutadas	21.651	41.915	821.179.696	1.363.410.194	NULL	NULL	NULL	NULL	NULL

Tiempo de ejecución	0,003 mili	0,004 mili	1,985 mili	5,173 mili	NULL	NULL	NULL	NULL	NULL
Cantidad de líneas del código	Nota = Todas las funciones pertenecientes a la clase Graph no serán contadas, pues todos los algoritmos los comparten. Tampoco se cuentan las líneas donde se suman las asignaciones y comparaciones. removerAristas = 11 optimizedBruteForce (sin parámetros) = 53 optimizedBruteForce (con parámetros) = 47 Total = 111								

Nombre del algoritmo #3: Fuerza bruta

Operaciones	5 v 4 a	5 v 8 a	10 v 9 a	10 v 18 a	20 v 19 a	20 v 38 a	40 v 80 a	50 v 120 a	50 v 150 a
Asignaciones	22.289	36.739	1.169.753.572	1.848.371.160	NULL	NULL	NULL	NULL	NULL
Comparaciones	17.342	28.235	1.009.262.164	1.545.990.452	NULL	NULL	NULL	NULL	NULL
Cantidad de líneas ejecutadas	39.631	64.974	2.179.015.736	3.394.361.612	NULL	NULL	NULL	NULL	NULL
Tiempo de ejecución	0,002 mili	0,002 mili	6,770 mili	9,324 mili	NULL	NULL	NULL	NULL	NULL
Cantidad de líneas del código	Nota = todas las funciones pertenecientes a la clase Graph no serán contadas, pues todos los algoritmos los comparten. Tampoco se cuentan las líneas donde se suman las asignaciones y comparaciones. removerAristas = 11 combinaciones = 20 bruteForce = 76								

Determinar el factor de crecimiento

NOTA: Por cantidad de datos, tiempo de ejecución siempre va a ser el mejor (omega) por lo que se está comparando igualmente en busca del mejor en asignaciones, comparaciones y líneas ejecutadas.

Nombre del algoritmo #1: Heurística Voraz

Talla por arcos	Factor talla	Factor Asig	Factor comparaciones	Factor cantidad de líneas ejecutadas	Factor tiempo de ejecución
120 a 150	1,25	228.794/186.264=1,228	222.205/181.057=1,227	450.999/367.321=1,227	0,034/0,015=2,266
19 a 38	2	12.272/4.997=2,455	11.370/4.645=2,447	23.642/9.642=2,451	0,004/0,001=4
80 a 120	1,5	186.264/68.889=2,70	181.057/66.542=2,720	367.321/135.431=2,712	0,015/0,008=1,875
38 a 80	2,10	68.889/12.272=5,613	66.542/11.370=5,852	135.431/23.642=5,728	0,008/0,004=2
8 a 120	15	186.264/498=374,024	181.057/373=485,407	367.321/871=421,723	0,015/0,001=15
8 a 150	18,75	228.794/498=459,425	222.205/373=595,723	450.999/871=517,794	0,034/0,001=34
4 a 150	37,5	228.794/229=999,100	222.205/167=1.330,568	450.999/396=1.138,886	0,034/0,001=34

Usar la notación que más se ajuste a los resultados obtenidos y su respectiva clasificación	
Clasificación del comportamiento de las **asignaciones**	**$\Omega(n^3)$**
Clasificación del comportamiento de las **comparaciones**	**$O(n^3)$**
Clasificación del comportamiento de las **líneas ejecutadas**	**$\theta(n^3)$**
Clasificación del comportamiento en el **tiempo de ejecución**	**$\Omega(n)$**

NOTA: Por cantidad de datos, tiempo de ejecución siempre va a ser el mejor (omega) por lo que se está comparando igualmente en busca del mejor en asignaciones, comparaciones y líneas ejecutadas. (Todas las tablas)

Nombre del algoritmo #2: Fuerza bruta optimizado

Talla por arcos	Factor talla	Factor Asig	Factor comparaciones	Factor cantidad de líneas ejecutadas	Factor tiempo de ejecución
4 a 8	2	24.735/12.759=1,938	17.180/8.892= 1,932	41.915/21.651=1,935	0,004/0,003= 1,333
8 a 9	1,125	486.821.294/24.735=19.681,475	334.358.402/17.180= 19.462,072	821.179.696/41.915=19.591,547	1,985/0,004= 496,25
4 a 9	2,25	486.821.294/12.759=38.155,129	334.358.402/8.892= 37.602,159	821.179.696/21.651=37.928,026	1,985/0,003= 661,666
9 a 18	2	834.703.296/486.821.294=1,714	528.706.898/334.358.402= 1,581	1.363.410.194/821.179.696=1,660	5,173/1,985= 2,606

8 a 18	2,25	834.703.296/24.735=33.745,837	528.706.898/17.180= 30.774,557	1.363.410.194/41.915=32.527,977	5,173/0,004= 1.293,25
4 a 18	4,5	834.703.296/12.759=65.420,745	528.706.898/8.892= 59.458,715	1.363.410.194/21.651=62.972,158	5,173/0,003= 1.724,333

Usar la notación que más se ajuste a los resultados obtenidos y su respectiva clasificación	
Clasificación del comportamiento de las **asignaciones**	**O(n!)**
Clasificación del comportamiento de las **comparaciones**	**Θ(n!)**
Clasificación del comportamiento de las **líneas ejecutadas**	**Ω(n!)**
Clasificación del comportamiento en el **tiempo de ejecución**	**Ω(2^n)**

Nombre del algoritmo #3: Fuerza bruta

Talla por arcos	Factor talla	Factor Asig	Factor comparaciones	Factor cantidad de líneas ejecutadas	Factor tiempo de ejecución
4 a 8	2	36.739/22.289= 1,648	28.235/17.342= 1,628	64.974/39.631= 1,639	0,002/0,002=1
8 a 9	1,125	1.169.753.572/36.739= 31.839,559	1.009.262.164/28.235= 35.745,073	2.179.015.736/64.974= 33.536,733	6,770/0,002= 3.385
4 a 9	2,25	1.169.753.572/22.289= 52.481,204	1.009.262.164/17.342= 58.197,564	2.179.015.736/39.631= 54.982,607	6,770/0,002= 3.385
9 a 18	2	1.848.371.160/1.169.753.572= 1,580	1.545.990.452/1.009.262.164= 1,531	3.394.361.612/2.179.015.736= 1,557	9,324/0,002= 4.662
8 a 18	2,25	1.848.371.160/36.739= 50.310,872	1.545.990.452/28.235= 54.754,398	3.394.361.612/64.974= 52.241,844	9,324/0,002= 4.662
4 a 18	4,5	1.848.371.160/22.289= 82.927,505	1.545.990.452/17.342= 89.147,183	3.394.361.612/39.631= 85.649,153	9,324/0,002= 4.662

Usar la notación que más se ajuste a los resultados obtenidos y su respectiva clasificación	
Clasificación del comportamiento de las **asignaciones**	**Ω(n!)**
Clasificación del comportamiento de las **comparaciones**	**O(n!)**
Clasificación del comportamiento de las **líneas ejecutadas**	**Θ(n!)**
Clasificación del comportamiento en el **tiempo de ejecución**	**Ω(n)**

Medición Analítica

Medición analítica

NOTA: después de los algoritmos, están los métodos extras

NOTA: Los que tienen letra color rojo, representa métodos externos

Nombre del algoritmo de #1: Heurística Voraz

1	ArrayList<Aristas> aristas = grafo.obtenerAristas();	1+ 3n+4 = 3n+5
2	ArrayList<Vertice> vértices = grafo.obtenerVertices();	1+3n+4 = 3n+5
3	cover = new ArrayList<Vertice> ();	1
4	while(!aristas.isEmpty()) {	n+1
5	Vertice vertex = obtenerVerticeMaximo(vertices, aristas);	$(1 + 8+8n+ 12n^2)$ n = $12n^3+8n^2+9n$
6	cover.add(vertex);	1(n) = n
7	vertices.remove(vertex);	1(n) = n
8	this.removerAristas(vertex, aristas)	$(15n + 5)$ (n) = $15n^2+5n$
9	}	
	Suma total:	$12n^3+23n^2+14n+21$
	Clasificación en notación O Grande	$O(n^3)$

Nombre del algoritmo de #2: Fuerza bruta optimizado

1	ArrayList<Vertice> v = new ArrayList<Vertice> ();	1
2	for(int i = 0; i < this.k; i++) {	1+(n+1) + (n) = 2n+2
3	v.add(new Vertice(i));	1(n) = n
4	}	
5	boolean[] visit = new boolean[grafo.obtenerVertices().size()];	1+(3n+4)= 3n+5
6	for(int i = 0; i < grafo.obtenerVertices().size(); i++) {	1+(n+1) + (n) = 2n+2
7	visit[i] = false;	1(n) = n
8	}	0
9	this.min = k + 1;	1
10	cover = new ArrayList<Vertice>();	1
11	this.fuerzaBrutaOptimizado(grafo.obtenerVertices(), this.k, v, -1, visit);	$36n^3n!+41n^2n!+32nn!$
12	if (!this.cover.isEmpty()) {	1
13	//Solución	0
14	}	0
15	else {	0

16	//No solución	0
17	}	0
	Suma total:	$36n^3n!+41n^2n!+32nn!+9n+13$
	Clasificación en notación O Grande	$O(n! * n^3)$

Nombre del algoritmo de #3: Fuerza bruta

1	posibleCovers = new ArrayList<ArrayList<Vertice>> ();	1
2	ArrayList<Vertice> v = new ArrayList<Vertice> ();	1
3	for(int i = 0; i < this.k; i++) {	1+(n+1) + (n) = 2n+2
4	v.add(new Vertice(i));	1(n) = n
5	}	0
6	boolean[] visit = new boolean[grafo.obtenerVertices().size()];	1+3n+4 = 3n + 5
7	for(int i = 0; i < grafo.obtenerVertices(). size(); i++) {	(1+(n+1) + (n))+3n+4 = 5n+6
8	visit[i] = false;	1(n) = n
9	}	0
10	this.combinaciones(grafo.obtenerVertices(), this.k, v, -1, visit);	$(10n^2n!+13nn!)+ 3n+4=$ $10n^2n!+13nn!+3n+4$
11	ArrayList<Aristas> aristas = grafo.obtenerAristas();	1 + 3n+4 = 3n+5
12	cover = new ArrayList<Vertice> ();	1
13	int min = this.k + 1;	1
14	int coverIndex = 0;	1
15	for (int i = 0; i < posibleCovers.size(); i++) {	1+(n+1) + (n) = 2n+2
16	for (int j = 0; j < this.k; j++) {	$(1+(n+1) + (n))$ n = $2n^2+2n$
17	this.removerAristas(posibleCovers.get(i).get(j), aristas);	$(15n + 5)(n)(n) =15n^3+5n^2$
18	if (aristas.isEmpty()) {	$1(n)(n) = n^2$
19	if ((j + 1) < min) {	$1(n)(n) = n^2$
20	min = (j + 1);	$1(n)(n) = n^2$
21	coverIndex = i;	$1(n)(n) = n^2$
22	}	0
23	}	0
24	}	0
25	aristas = grafo.obtenerAristas();	$(1+3n+4)n = 3n^2+5n$
26	}	0
27	if (min <= k) {	1
28	for (int i = 0; i < min; i++){	1+(n+1) + (n) = 2n+2
29	cover.add(posibleCovers.get(coverIndex).get(i));	1(n) = n
30	}	0
31	}	0
32	else{	0
33	//No solución	0
34	}	0

	Suma total:	$15n^3+14n^2+10n^2n!+13nn!+30n+31$
	Clasificación en notación O Grande	$O(n! * n^2)$

METODOS EXTRAS:

Método obtenerAristas

1	ArrayList<Aristas> aristas = new ArrayList<Aristas> ();	1
2	for (Aristas arista : this.aristas) {	1+(n+1) + (n) = 2n+2
3	aristas.add(arista);	1(n) = n
4	}	0
5	return aristas;	1
	Suma total:	3n+4

Método obtenerVertices

1	ArrayList<Vertice> vertices = new ArrayList<Vertice> ();	1
2	for (Vertice vertex : this.vertices) {	1+(n+1) + (n) = 2n+2
3	vertices.add(vertex);	1(n) = n
4	}	0
5	return vertices;	1
	Suma total:	3n+4

Método getVertice

1	return vertice;	1
	Suma total:	1

Método hasVertex

1		1 parámetro
2	If(this.vertice1.getVertice() == vertex.getVertice() || this.vertice2.getVertice() == vertex.getVertice())	4+2 (4 llamadas a getVertice 1 paso cada una + las 2 comparaciones)
3	return true;	1
4	else	0
5	return false;	1
	Suma total:	9

Método obtenerVerticeMaximo

1		2 parámetros

2	Map<Vertice, Integer> map = new HashMap<Vertice,Integer>();	1
3	Int max = 0;	1
4	Vertice maxVertice = new Vertice ();	1
5	for(Vertice vertice : vertices) {	1+(n+1) + (n) = 2n+2
6	Map,put(vertice,0);	1(n)=n
7	for(Aristas arista : aristas) {	(1+(n+1) + (n)) n= (2n+2) (n)=$2n^2$+2n
8	if(arista.hasVertex(vertice)){	9(n)(n) = $9n^2$
9	map.put(vertice,map.get(vertice)+1);	1(n)(n) = n^2
10	}}	0
11	if(map.get(vertice) > max) {	1(n) = n
12	max = map.get(vertice);	1(n) = n
13	maxVertice = vertice;	1(n) = n
14	}}	0
15	Return maxVertice;	1
	Suma total:	8+8n+ 12n^2

Método removerAristas

		2 parámetros
1	ArrayList<Aristas> removerArista = new ArrayList<Aristas>();	1
2	for (int i = 0; i < listaAristas.size(); i++) {	1+(n+1) + (n) = 2n+2
3	if ((listaAristas.get(i)).hasVertex(vertice)){	9(n) = 9n
4	removerArista.add(listaAristas.get(i));	1(n) = n
5	}	0
6	}	0
7	for (int i = 0; i < removerArista.size(); i++){	1+(n+1) + (n) = 2n+2
8	listaAristas.remove(removerArista.get(i));	1(n) = n
9	}	0
10	}	0
	Suma total:	15n + 5

Método combinaciones

1		5 parámetros
2	if (actualTot >= k-1) {	1
3	ArrayList<Vertice> temp = new ArrayList<Vertice> ();	1
4	for (int i = 0; i < actual.size(); i++) {	1+(n+1) + (n) = 2n+2
5	temp.add(actual.get(i));	1(n) = n
6	}	0
7	posibleCovers.add(temp);	1
8	return;	1
9	}	0
10	for (int i = 0; i < vertices.size(); i++) {	1+(n+1) + (n) = 2n+2
11	if(!visit[i]) {	1(n) = n
12	actual.set(++actualTot, vertices.get(i));	1(n) = n
13	visit[i] = true;	1(n) = n

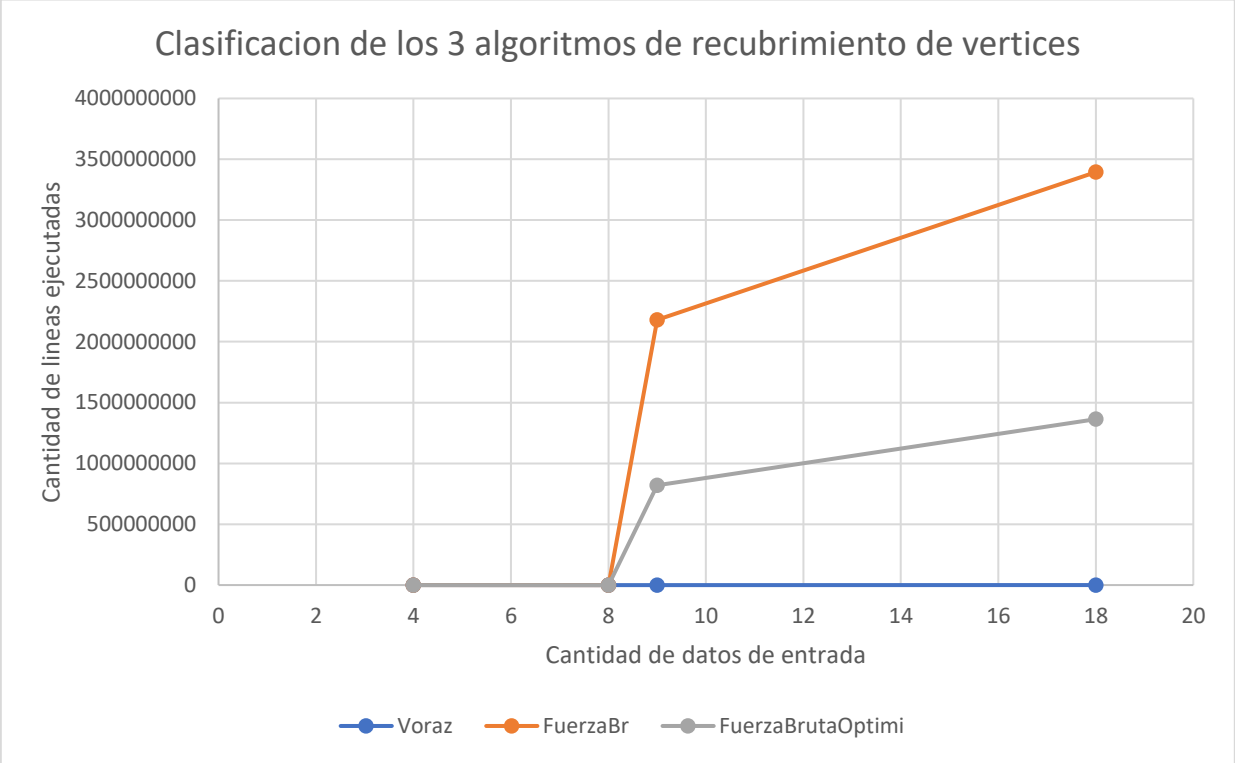
14	combinaciones(vertices, k, actual, actualTot, visit);	n!(n)
15	visit[i] = false;	1(n) = n
16	actualTot--;	1(n) = n
17	}	0
18	}	0
	Suma total:	n!*n(10n+13)=$10n^2n!+13nn!$

MÉTODO fuerzaBrutaOptimizado (No es el algoritmo en sí, es un método utilizado por el algoritmo)

1		5 parámetros
2	ArrayList<Aristas> aristas = grafo.obtenerAristas();	1+3n+4 = 3n+5
3	if (actualTot < k-1) {	1
4	if (actualTot < min) {	1
5	for (int j = 0; j < actual.size(); j++) {	1+(n+1) + (n) = 2n+2
6	this.removerAristas(actual.get(j), aristas);	(15n + 5)(n) = $15n^2$+5n
7	if (aristas.isEmpty()) {	1(n) = n
8	if ((j + 1) < this.min) {	1(n) = n
9	this.min = (j + 1);	1(n) = n
10	for (int i = 0; i <= j; i++) {	(1+(n+1) + (n))n = $2n^2$+2n
11	cover.add(actual.get(i));	1(n)(n) = n^2
12	}	0
13	}	0
14	}	0
15	}	0
16	}	0
17	aristas = grafo.obtenerAristas();	1+3n+4 = 3n+5
18	}	0
19	else if (actualTot == k-1 && actualTot < min) {	2
20	for (int j = 0; j < actual.size(); j++) {	1+(n+1) + (n) = 2n+2
21	this.removerAristas(actual.get(j), aristas);	(15n + 5)n = $15n^2$+5n
22	if (aristas.isEmpty()) {	1(n) = n
23	if ((j + 1) < this.min) {	1(n) = n
24	this.min = (j + 1);	1(n) = n
25	cover.clear();	1(n) = n
26	for (int i = 0; i <= j; i++) {	(1+(n+1) + (n))n = $2n^2$+2n
27	cover.add(actual.get(i));	1(n)(n) = n^2
28	}	0
29	}	0
30	}	0
31	}	0
32	aristas = grafo.obtenerAristas();	1+3n+4 = 3n+5
33	return;	1
34	}	0

35	else {	0
36	return;	1
37	}	0
38	for (int i = 0; i < vertices.size(); i++) {	1+(n+1) + (n) = 2n+2
39	if(!visit[i]) {	1(n) = n
40	actual.set(++actualTot, vertices.get(i));	1(n) = n
41	visit[i] = true;	1(n) = n
42	fuerzaBrutaOptimizado(vertices, k, actual, actualTot, visit);	n!(n)
43	visit[i] = false;	1(n) = n
44	actualTot--;	1(n) = n
45	}	0
46	}	0
	Suma total:	n!*n ($36n^2+41n+32$) = $36n^3n!+41n^2n!+32nn!$

Medición gráfica:



Comentario del grafico:

Como se puede observar en el gráfico, el algoritmo voraz el cual cuenta con una complejidad temporal de $O(n^3)$ tiene una cantidad de líneas ejecutadas mucho menor a las de sus competidores; fuerza bruta y fuerza bruta optimizado, donde ambos cuentan con una complejidad temporal de O(n!).

Debido a que la complejidad de n^3 presenta un crecimiento de cantidad de líneas ejecutadas, mucho menor a una complejidad factorial, se genera una diferencia abismal en cuanto a la cantidad de líneas ejecutadas por la cantidad de datos de entrada, esto hace que la gráfica del algoritmo voraz parezca que es lineal, lo cual no es así, ya que, en comparación a los otros dos algoritmos, su pendiente es mínima y genera esta sensación representada en el gráfico.

Por último, a pesar de que el algoritmo fuerza bruta optimizado nos haya dado un peor de los casos mayor en comparación al algoritmo de fuerza bruta ordinario, se puede apreciar que este segundo presenta una mayor cantidad de líneas ejecutadas por datos de entrada. Lo cual hace ver que la complejidad no determina totalmente el rendimiento de los algoritmos, ya que dependiendo del contexto del programa en el que se ejecuten, esto puede variar enormemente y es por eso que se debe realizar una medición empírica para llegar a conclusiones exactas en el caso que se está tratando en un proyecto determinado.

Análisis de todos los algoritmos

A partir de los datos obtenidos, rápidamente se puede inferir que el algoritmo voraz es con diferencia el algoritmo que trabaja con el mejor rendimiento de todos, sus características de utilizar una llamada a la función “obtenerVérticeMáximo” más sus propios procesos lo llevan a tener una complejidad de n^3, siendo esta una complejidad realmente buena para problemas no determinísticos que suelen ser exponenciales; esto en definitiva, en comparación con los otros 2 algoritmos, hace que el algoritmo de heurística voraz genere un promedio de asignaciones, comparaciones y por ende, de líneas ejecutadas mucho menor en cada una de las ejecuciones del programa, lo que le da en definitiva el título del algoritmo con el mejor rendimiento entre todos los utilizados en el programa.

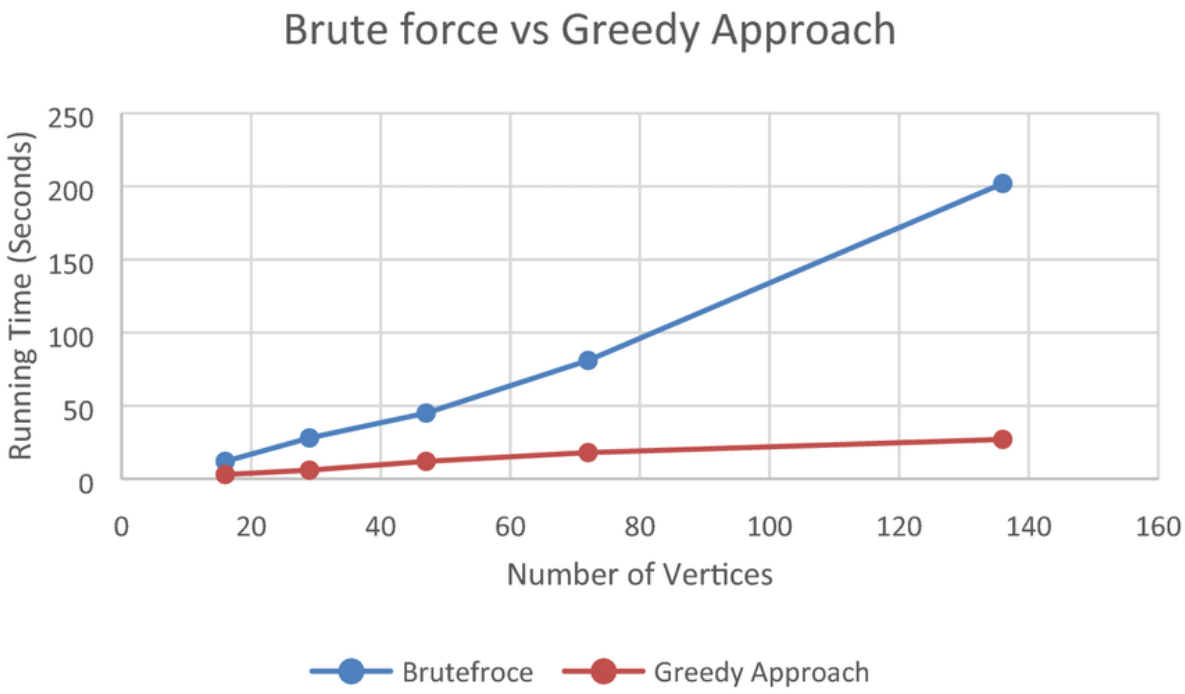
Como segundo mejor algoritmo de los analizados en cuanto a su rendimiento, está el algoritmo de fuerza bruta optimizado, este algoritmo al utilizar llamadas recursivas dentro de un ciclo hace que su complejidad se eleve a la de las factoriales, haciendo de este algoritmo uno realmente poco eficiente y los datos obtenidos lo demuestran, ya que después de realizar varios testeos, el algoritmo no da respuesta a partir de los 13 de vértices y 18 aristas. En general el crecimiento de sus líneas ejecutadas es lo que se podría esperar de un algoritmo factorial, sin embargo, se puede ver que este conteo es considerablemente menor al promedio,

si consideramos que el algoritmo de fuerza bruta siempre realiza un poco más de líneas ejecutadas en cada ejecución. A pesar de que todo apunta a que en unas condiciones normales, el algoritmo de fuerza bruta optimizado es mejor que el algoritmo de fuerza bruta, este aun así cuenta con una complejidad más elevada, por lo que si se va al peor de los casos en ambos algoritmos, el fuerza bruta optimizado tendrá el peor rendimiento de los dos y por lo tanto no siempre el juicio aplicado en este análisis, lastimosamente no siempre será el mismo, y en algunas ocasiones muy asoladas, el por algoritmo de los tres debe de ser el algoritmo de fuerza bruta optimizado.

El tercer algoritmo, y por lo tanto el peor de todos es el llamado algoritmo de fuerza bruta, el hecho de que este algoritmo utiliza el método de combinations, hace que el algoritmo compruebe todas y cada una de las posibles soluciones que se podrían obtener, por lo que en la mayoría de los casos en los que se ejecute, se llegarán a valores muy cercanos al peor de sus casos el cual es de complejidad factorial también; es por esta razón que casi siempre es el algoritmo más ineficiente de los 3 presentados en el proyecto. El algoritmo de fuerza bruta optimizado cuenta con más herramientas para no llegar al peor de sus casos, y gracias a esto es que por lo general nunca supera en ineficiencia al algoritmo de fuerza bruta clásico. Para poner un poco más en perspectiva el hecho de que este algoritmo es por lo general el peor de todos, está el hecho de que este algoritmo deja de dar respuesta a partir de los 11 vértices y 18 aristas, que es considerablemente peor al margen de error de 13 vértices y 18 aristas que permite el algoritmo de fuerza bruta optimizado; entonces por esta razón y todas las demás mencionadas es que se da como veredicto que el algoritmo de fuerza bruta es el peor de todos.

Comparación con otros algoritmos similares

Figura 1.



Nota. Adaptado de *A novel approach for agricultural decision making using graph coloring*, por K. Subramanian et all, 2020, ResearchGate(https://www.researchgate.net/publication/337787551_A_novel_approach_for_agricultural_decision_making_using_graph_coloring).

En la Figura 1 se puede observar una medición gráfica en base al tiempo de ejecución de un algoritmo de fuerza bruta y un algoritmo de heurística voraz. Al observar el comportamiento que tienen los datos de esta figura, rápidamente se puede comprobar que en definitiva los valores obtenidos en la medición empírica son correctos, ya que mientras más datos de entrada se tenga, más será la diferencia entre ambos algoritmos y por lo tanto, el algoritmo de heurística voraz se acercará más y más al eje X, dando esa sensación de que se trata de un algoritmo prácticamente logarítmico, cuando en realidad no es así y solo es una ilusión generada por la exagerada diferencia de valores entre ambos algoritmos. En el caso del algoritmo de fuerza bruta optimizado utilizado en este proyecto sería un comportamiento realmente similar al de la imagen de Subramanian et all (2020), puesto que al igual que el algoritmo de fuerza bruta, este cuenta con una complejidad factorial, por lo que no importa con cual algoritmo se compare el algoritmo de heurística voraz, mientras más datos se traten de representar gráficamente, este simplemente tenderá a acercarse al eje X.

Conclusiones

Se concluye que el algoritmo de heurística voraz utilizado en este proyecto, es el mejor de los 3 algoritmos comparados, tanto empíricamente, ya que su conteo de asignaciones y comparaciones por ejecución del programa es en todas las ocasiones mucho más eficiente que los otros 2 algoritmos analizados, como analíticamente, ya que cuenta con una complejidad de n^3, la cual resulta ser mucho mejor que la complejidad factorial de sus competidores.

A su vez, se concluye que como segundo mejor algoritmo el merecedor por los datos obtenidos en las diferentes mediciones realizadas es el algoritmo de fuerza bruta optimizado, ya que empíricamente genera resultados de un mejor rendimiento en comparación al algoritmo de fuerza bruta clásico, lo que empíricamente lo colocaría como el segundo mejor algoritmo analizado. Pero, por otra parte, analíticamente este algoritmo es peor que el de fuerza bruta clásico, ya que cuenta con una factorial adicional multiplicando una de las constantes en su haber, elemento que lo lleva a albergar un peor de los casos mayor.

Por último, el tercer algoritmo y por lo tanto, que se concluye como el peor algoritmo de todos, es el algoritmo de fuerza bruta, ya que a pesar de que analíticamente, goza de tener una complejidad más baja que el algoritmo de fuerza bruta optimizado, los resultados obtenidos en la medición empírica de este proyecto son en todos los casos ampliamente superiores, y por lo tanto, demuestran que su rendimiento es peor, así que a efectos de los datos obtenidos y las sensaciones generales al ejecutar programa, se decide que el algoritmo de fuerza buta clásico es el peor de los 3 analizados.

Recomendaciones

Para ampliar los horizontes del alcance del tema tratado en este trabajo, se recomienda realizar un análisis de estos mismos tipos de algoritmos, adaptándolos para que funcionen con árboles binarios e incluso de niveles aún mayores; de esta forma se podrá tener un conocimiento mucho más amplio sobre el tema y se podrá analizar el rendimiento de los distintos algoritmos existentes bajo diferentes contextos.

De igual forma, una manera muy buena de enriquecer aún más el contenido del tema tratado en este trabajo sería el incluir algunos algoritmos de aproximación, los cuales no fueron elegidos en este caso pero que sin duda sería interesante conocer su rendimiento según el tamaño de los grafos utilizados, así como para compararlos con los ejemplos ya mostrados en este proyecto.

A su vez, sería interesante explorar la posibilidad de que otros campos de estudio puedan aportar al análisis de algoritmos de optimización como lo son los utilizados en este trabajo. Un ejemplo de esto puede ser el campo de la economía, la cual podría aplicar este tipo de algoritmos en numerosos contextos en los cuales se necesite la mejor opción , la cuál puede ser el determinar el mejor valor de distintos materiales para la fabricación de un producto, o las mejores rutas de comercio, por lo que se recomienda adaptar algoritmos de complejidades similares para su utilización en otros contextos y así extender el conocimiento sobre este tema.

Por último, si se está trabajando sobre un proyecto de características similares a las del tratado en este caso, al momento de tomar los datos resultantes del rendimiento de los distintos algoritmos utilizados como lo pueden ser el número de comparaciones y asignaciones, o incluso el tiempo de ejecución, se recomienda que se tomen la mayor cantidad de muestras posibles por algoritmo, ya que al tener una gran cantidad de información sobre la cual trabajar, el análisis de las distintas complejidades, el factor de talla, el factor de crecimiento e incluso la medición gráfica de los algoritmos analizados serán procesos mucho más sencillos. Es por esto que es importante recordar que mientras más ejemplos se tenga sobre del funcionamiento de algo en específico, más es la evidencia que respalda cualquier tipo de afirmación realizada, ya que, al fin y al cabo, en la sociedad actual, la información es poder.

Bitácora y minutas del trabajo

Número de actividad	Fecha y hora	Lugar o medio	Participantes	Asuntos a tratar	Acuerdos y responsables	Asuntos pendientes
1	Miércoles 31 de agosto a las 12:10 p.m.	Instituto Tecnológico de Costa Rica.	Fabricio Porras Morera y Carlos Solís Mora.	Búsqueda de códigos funcionales de Fuerza bruta, Voraz y el adicional.	Se acordó que ambos responsables, Fabricio Porras y Carlos Solís buscarían en internet sobre los distintos códigos necesarios.	Queda pendiente encontrar todos los códigos. ¿Completado? Si(X) No()
2	Lunes 5 de septiembre a las 6 p.m.	Discord y Whatsapp.	Fabricio Porras Morera y Carlos Solís Mora.	Analizar el funcionamiento de los códigos encontrados.	Se acordó que ambos responsables, Fabricio Porras y Carlos Solís consultarían a la profesora acerca de si los códigos elegidos podrían ser utilizados, sobre todo el caso del tercer código de Fuerza Bruta Optimizado.	Consultar a la profesora acerca de los códigos encontrados en internet. ¿Completado? Si(X) No()
3	Miércoles 7 de septiembre a las 11:50 a.m.	Instituto Tecnológico de Costa Rica.	Fabricio Porras Morera y Carlos Solís Mora.	Modificación de los códigos encontrados.	Se acordó que ambos responsables, Fabricio Porras y Carlos Solís, se encargarían de modificar el código para que este contabilice correctamente las asignaciones y comparaciones realizadas por cada código encontrado, además de programar una manera de que los grafos se	Modificar el código para que contabilice tanto asignaciones como comparaciones, además de que los grafos se realicen de manera aleatoria y automática. ¿Completado? Si() No(X)

					creen de manera aleatoria.	
4	Sábado 10 de septiembre a las 3:00 a.m.	Discord y Whatsapp.	Fabricio Porras Morera y Carlos Solís Mora.	Método para que los vértices del grafo se entreguen de forma aleatoria, así como el método para que los arcos se realicen de forma aleatoria.	Se acordó que los responsables Fabricio Porras y Carlos Solís, realizarían los métodos para que la construcción del grafo sea de forma aleatoria.	Modificar el código para que contabilice tanto asignaciones como comparaciones. ¿Completado? Si(X) No()
5	Jueves 15 de septiembre a las 11:00 p.m.	Discord y Whatsapp.	Fabricio Porras Morera y Carlos Solís Mora.	Conteo de asignaciones y comparaciones, traducción del código.	Se acordó que los responsables Fabricio Porras y Carlos Solís, realizarían el conteo de comparaciones asignaciones, igualmente, se comenzó a traducir el código ya que era estaba en inglés en su totalidad.	Revisión del conteo de asignaciones y comparaciones realizadas en el código. ¿Completado? Si(X) No()
6	Domingo 18 de septiembre a las 7:00 p.m.	Discord y Whatsapp.	Fabricio Porras Morera y Carlos Solís Mora.	Revisión de conteo de asignaciones y comparaciones, finalización de la traducción del código.	Se acordó que los responsables Fabricio Porras y Carlos Solís, revisarían el conteo de asignaciones y comparaciones previamente realizado, también el finalizar la traducción del código en su totalidad	Comenzar el registro de asignaciones y comparaciones según el tamaño del gráfico para cada una de los métodos conseguidos. ¿Completado? Si(X) No()
7	Viernes 23 de septiembre a las 9:00 p.m.	Discord y Whatsapp.	Fabricio Porras Morera y Carlos Solís Mora.	Registro de asignaciones y comparaciones para cada uno	Se acordó que los responsables Fabricio Porras y Carlos Solís, realizarían un	Queda pendiente hacerle algunas consultas a la profesora

				de los métodos conseguidos.	primer registro de las asignaciones y comparaciones para cada uno de los métodos conseguidos.	acerca del factor talla conseguido en el proceso de registro, ya que no calza con los valores conseguidos. ¿Completado? Si(X) No()
8	Lunes 26 de septiembre a las 3:00 a.m.	Discord y Whatsapp.	Fabricio Porras Morera y Carlos Solís Mora.	Consultas sobre dilemas acerca del factor talla conseguido en el proceso de registro de asignaciones y comparaciones.	Se acordó que los responsables Fabricio Porras y Carlos Solís, realizarían diferentes consultas para clarificar el por qué de los factores tallas conseguidos en el proceso de registro.	Ejecutar de forma perfecta el análisis empírico de cada uno de los métodos conseguidos. ¿Completado? Si(X) No()
9	Martes 27 de septiembre a las 11 p.m.	Discord y Whatsapp.	Fabricio Porras Morera y Carlos Solís Mora.	Finalización de medición empírica, comienzo de la medición analítica	Se acordó que los responsables Fabricio Porras y Carlos Solís, realizarían la medición empírica en su totalidad, además se completó la medición analítica del método de Heurística Voraz	Finalizar la medición analítica de los métodos de Fuerza bruta, y Fuerza Bruta optimizado. ¿Completado? Si(X) No()
10	Jueves 29 de septiembre a las 12:30 a.m.	Discord y Whatsapp.	Fabricio Porras Morera y Carlos Solís Mora.	Finalización de medición analítica	Se acordó que los responsables Fabricio Porras y Carlos Solís, terminarían la medición analítica finalmente.	Realizar uno de los gráficos requeridos, y realizar la clasificación de Omega, Theta y O grande de las asignaciones, comparaciones, líneas

						ejecutadas y tiempo de ejecución de cada uno de los algoritmos ¿Completado? Si(X) No()
11	Jueves 30 de septiembre a las 12:30 a.m.	Discord y Whatsapp.	Fabricio Porras Morera y Carlos Solís Mora.	Finalización de medición gráfica, clasificación Omega, theta y O grande los distintos valores analizados por algoritmo, y finalmente la documentación escrita del proyecto.	Se acordó que los responsables Fabricio Porras y Carlos Solís, terminarían los últimos pendientes para finalizar el proyecto.	Dar los retoques finales para la finalización del proyecto ¿Completado? Si(X) No()

Referencias

Boham, D. (2017). *java-vertex-cover*. Recuperado de: https://github.com/bonhamdaniel/java-vertex-cover

Samufo. (2020). *How to Create a Random Graph Using Random Edge Generation in Java?*. Recuperado de: https://www.geeksforgeeks.org/how-to-create-a-random-graph-using-random-edge-generation-in-java/

TofuBeer. (2013). *Displaying seconds in 3 decimal places. – java*. Recuperado de: https://stackoverflow.com/questions/10593834/displaying-seconds-in-3-decimal-places-java

Subramanian, K., & Bhanu, D., & Subramanian, B. (2020). Brute force vs Greedy Approach [Imagen]. ResearchGate. https://www.researchgate.net/publication/337787551_A_novel_approach_for_agricultural_decision_making_using_graph_coloring

RUBRICA PARA EVALUAR EL PROCESO DEL

TRABAJO EN EQUIPO

COEVALUACIÓN

CRITERIO	1 Deficiente 0 pts.	2 Requiere mejorar 1 pt.	3 Bueno 2 pts.	4 Excelente 3pts.	Observaciones o comentarios
Participación 30%	Ausencia en aportar ideas en la toma de decisiones de forma grupal.	Proporcionan ideas difusas o confusas en la discusión del grupo y hace lo se le pide.	Proporcionan ideas útiles en la discusión del grupo, y cumple con lo programado	Proporcionan ideas útiles en la discusión del grupo y evalúa alternativas con base a la viabilidad, enriqueciendo la participación del grupo en la toma de decisiones.	
Actitud 20%	El estudiante muestra crítica en público el trabajo de sus compañeros de equipo incluso justifica sus carencias en los errores de sus pares y dificultades en la realización del proyecto, lo que desfavorece en mantener la unión en el equipo.	Dentro de las actividades por cumplir en el equipo, muestran una actitud positiva ante el trabajo en equipo y proyecto, aunque muestra despreocupa en la unión en el equipo	Sus actitudes son positivas ante el trabajo en equipo y proyecto, lo que se muestran al colaborar y mantener la unión en el equipo.	Sus actitudes son positivas ante el trabajo en equipo y proyecto y buscan mantener la unión en el equipo, promoviendo una sana convivencia.	
Responsabilidad 10%	Incumplen con los roles asignados de forma individual y esto perjudica con el compromiso con el trabajo.	Asumen roles determinados por el equipo, aunque su participación es regular en el desempeño de su equipo.	Asumen roles y colabora en la realización, demostrando una participación buena en el desempeño de su responsabilidad en el equipo.	Asumen eficientemente sus roles y temas de los cuales se hace cargo, demostrando una participación clave en el desempeño de su equipo y evidencia una colaboración con otros.	
Resolución de conflictos 10%	En situaciones de desacuerdos o conflicto, muestran una limitada escucha con respecto a otras opiniones o acepta sugerencias, por lo que carece de propuestas alternativas y le cuesta aceptar el consenso o la solución.	En situaciones de desacuerdos o conflicto, escuchan de forma limitada otras opiniones y acepta sugerencias, pero sin proponer alternativas para aceptar el consenso.	En situaciones de desacuerdos o conflicto, escuchan otras opiniones y acepta sugerencias.	En situaciones de desacuerdos o conflicto, al escuchar otras opiniones y acepta sugerencias, propone alternativas para la solución de forma colaborativa y promueve el consenso.	
Seguimiento del tema 20%	Se desconcentra o realiza actividades fuera del tema o actividad, lo que incumplen con el tiempo programado.	Se mantienen en el tema o actividad algunas veces, del tiempo programado.	Se mantienen en el tema o actividad la mayor parte del tiempo programado.	Se mantienen en el tema o actividad en el tiempo programado y revisan con cautela sus avances y progreso.	
Uso del tiempo 10%	Ausencia de presentar los productos realizados por cada miembro del equipo provocando que otros asuman sus responsabilidades para cumplir con los tiempos establecidos.	Tiende a demorarse en la presentación de sus productos como miembro del equipo en las fechas establecida, según perjudicando el cumplimiento de la meta esperada.	Presenta cada uno de los miembros los productos realizados al ser responsable con los tiempos establecidos.	Cada miembro es organizado y presentan sus productos a tiempo, más bien colaboran con otros miembros que muestra atraso o requieren apoyo, para cumplir con las fechas establecidas.	

CRITERIO	1 Deficiente 0 pts.	2 Requiere mejorar 1 pt.	3 Bueno 2 pts.	4 Excelente 3pts.	Observaciones o comentarios
Participación 30%	Ausencia en aportar ideas en la toma de decisiones de forma grupal.	Proporcionan ideas difusas o confusas en la discusión del grupo y hace lo se le pide.	Proporcionan ideas útiles en la discusión del grupo, y cumple con lo programado	Proporcionan ideas útiles en la discusión del grupo y evalúa alternativas con base a la viabilidad, enriqueciendo la participación del grupo en la toma de decisiones.	
Actitud 20%	El estudiante muestra crítica en público el trabajo de sus compañeros de equipo, incluso justifica sus carencias en los errores de sus pares y dificultades en la realización del proyecto, lo que desfavorece en mantener la unión en el equipo.	Dentro de las actividades por cumplir en el equipo, muestran una actitud positiva ante el trabajo en equipo y proyecto, aunque muestra despreocupa en la unión en el equipo	Sus actitudes son positivas ante el trabajo en equipo y proyecto, lo que se muestran al colaborar y mantener la unión en el equipo.	Sus actitudes son positivas ante el trabajo en equipo y proyecto y buscan mantener la unión en el equipo, promoviendo una sana convivencia.	
Responsabilidad 10%	Incumplen con los roles asignados de forma individual y esto perjudica con el compromiso con el trabajo.	Asumen roles determinados por el equipo, aunque su participación es regular en el desempeño de su equipo.	Asumen roles y colabora en la realización, demostrando una participación buena en el desempeño de su responsabilidad en el equipo.	Asumen eficientemente sus roles y temas de los cuales se hace cargo, demostrando una participación clave en el desempeño de su equipo y evidencia una colaboración con otros.	
Resolución de conflictos 10%	En situaciones de desacuerdos o conflicto, muestran una limitada escucha con respecto a otras opiniones o acepta sugerencias, por lo que carece de propuestas alternativas y le cuesta aceptar el consenso o la solución.	En situaciones de desacuerdos o conflicto, escuchan de forma limitada otras opiniones y acepta sugerencias, pero sin proponer alternativas para aceptar el consenso.	En situaciones de desacuerdos o conflicto, escuchan otras opiniones y acepta sugerencias.	En situaciones de desacuerdos o conflicto, al escuchar otras opiniones y acepta sugerencias, propone alternativas para la solución de forma colaborativa y promueve el consenso.	
Seguimiento del tema 20%	Se desconcentra o realiza actividades fuera del tema o actividad, lo que incumplen con el tiempo programado.	Se mantienen en el tema o actividad algunas veces, del tiempo programado.	Se mantienen en el tema o actividad la mayor parte del tiempo programado.	Se mantienen en el tema o actividad en el tiempo programado y revisan con cautela sus avances y progreso.	
Uso del tiempo 10%	Ausencia de presentar los productos realizados por cada miembro del equipo provocando que otros asuman sus responsabilidades para cumplir con los tiempos establecidos.	Tiende a demorarse en la presentación de sus productos como miembro del equipo en las fechas establecida, según perjudicando el cumplimiento de la meta esperada.	Presenta cada uno de los miembros los productos realizados al ser responsable con los tiempos establecidos.	Cada miembro es organizado y presentan sus productos a tiempo, más bien colaboran con otros miembros que muestra atraso o requieren apoyo, para cumplir con las fechas establecidas.	

Nombres de estudiantes y firma:	Nota ponderada en el subgrupo:
1. Fabricio Porras Morera	17
2. Carlos Solís Mora	17

Autoevaluación

Nombre del Estudiante: Fabricio Porras Morera					**Observaciones o comentarios**
CRITERIO	**1 Deficiente 0 pts.**	**2 Requiere mejorar 1 pt.**	**3 Bueno 2 pts.**	**4 Excelente 3pts.**	
Participación 30%	Ausencia en aportar ideas en la toma de decisiones de forma grupal.	Proporcionan ideas difusas o confusas en la discusión del grupo y hace lo se le pide.	Proporcionan ideas útiles en la discusión del grupo, y cumple con lo programado	Proporcionan ideas útiles en la discusión del grupo y evalúa alternativas con base a la viabilidad, enriqueciendo la participación del grupo en la toma de decisiones.	
Actitud 20%	El estudiante muestra crítica en público el trabajo de sus compañeros de equipo, incluso justifica sus carencias en los errores de sus pares y dificultades en la realización del proyecto, lo que desfavorece en mantener la unión en el equipo.	Dentro de las actividades por cumplir en el equipo, muestran una actitud positiva ante el trabajo en equipo y proyecto, aunque muestra despreocupa en la unión en el equipo	Sus actitudes son positivas ante el trabajo en equipo y proyecto, lo que se muestran al colaborar y mantener la unión en el equipo.	Sus actitudes son positivas ante el trabajo en equipo y proyecto y buscan mantener la unión en el equipo, promoviendo una sana convivencia.	
Responsabilidad 10%	Incumplen con los roles asignados de forma individual y esto perjudica con el compromiso con el trabajo.	Asumen roles determinados por el equipo, aunque su participación es regular en el desempeño de su equipo.	Asumen roles y colabora en la realización, demostrando una participación buena en el desempeño de su responsabilidad en el equipo.	Asumen eficientemente sus roles y temas de los cuales se hace cargo, demostrando una participación clave en el desempeño de su equipo y evidencia una colaboración con otros.	
Resolución de conflictos 10%	En situaciones de desacuerdos o conflicto, muestran una limitada escucha con respecto a otras opiniones o acepta sugerencias, por lo que carece de propuestas alternativas y le cuesta aceptar el consenso o la solución.	En situaciones de desacuerdos o conflicto, escuchan de forma limitada otras opiniones y acepta sugerencias, pero sin proponer alternativas para aceptar el consenso.	En situaciones de desacuerdos o conflicto, escuchan otras opiniones y acepta sugerencias.	En situaciones de desacuerdos o conflicto, al escuchar otras opiniones y acepta sugerencias, propone alternativas para la solución de forma colaborativa y promueve el consenso.	
Seguimiento del tema 20%	Se desconcentra o realiza actividades fuera del tema o actividad, lo que incumplen con el tiempo programado.	Se mantienen en el tema o actividad algunas veces, del tiempo programado.	Se mantienen en el tema o actividad la mayor parte del tiempo programado.	Se mantienen en el tema o actividad en el tiempo programado y revisan con cautela sus avances y progreso.	
Uso del tiempo 10%	Ausencia de presentar los productos realizados por cada miembro del equipo provocando que otros asuman sus responsabilidades para cumplir con los tiempos establecidos.	Tiende a demorarse en la presentación de sus productos como miembro del equipo en las fechas establecida, según perjudicando el cumplimiento de la meta esperada.	Presenta cada uno de los miembros los productos realizados al ser responsable con los tiempos establecidos.	Cada miembro es organizado y presentan sus productos a tiempo, más bien colaboran con otros miembros que muestra atraso o requieren apoyo, para cumplir con las fechas establecidas.	

Autoevaluación

Nombre del Estudiante: Carlos Solís Mora					**Observaciones o comentarios**
CRITERIO	**1 Deficiente 0 pts.**	**2 Requiere mejorar 1 pt.**	**3 Bueno 2 pts.**	**4 Excelente 3pts.**	
Participación 30%	Ausencia en aportar ideas en la toma de decisiones de forma grupal.	Proporcionan ideas difusas o confusas en la discusión del grupo y hace lo se le pide.	Proporcionan ideas útiles en la discusión del grupo, y cumple con lo programado	Proporcionan ideas útiles en la discusión del grupo y evalúa alternativas con base a la viabilidad, enriqueciendo la participación del grupo en la toma de decisiones.	
Actitud 20%	El estudiante muestra crítica en público el trabajo de sus compañeros de equipo, incluso justifica sus carencias en los errores de sus pares y dificultades en la realización del proyecto, lo que desfavorece en mantener la unión en el equipo.	Dentro de las actividades por cumplir en el equipo, muestran una actitud positiva ante el trabajo en equipo y proyecto, aunque muestra despreocupa en la unión en el equipo	Sus actitudes son positivas ante el trabajo en equipo y proyecto, lo que se muestran al colaborar y mantener la unión en el equipo.	Sus actitudes son positivas ante el trabajo en equipo y proyecto y buscan mantener la unión en el equipo, promoviendo una sana convivencia.	
Responsabilidad 10%	Incumplen con los roles asignados de forma individual y esto perjudica con el compromiso con el trabajo.	Asumen roles determinados por el equipo, aunque su participación es regular en el desempeño de su equipo.	Asumen roles y colabora en la realización, demostrando una participación buena en el desempeño de su responsabilidad en el equipo.	Asumen eficientemente sus roles y temas de los cuales se hace cargo, demostrando una participación clave en el desempeño de su equipo y evidencia una colaboración con otros.	
Resolución de conflictos 10%	En situaciones de desacuerdos o conflicto, muestran una limitada escucha con respecto a otras opiniones o acepta sugerencias, por lo que carece de propuestas alternativas y le cuesta aceptar el consenso o la solución.	En situaciones de desacuerdos o conflicto, escuchan de forma limitada otras opiniones y acepta sugerencias, pero sin proponer alternativas para aceptar el consenso.	En situaciones de desacuerdos o conflicto, escuchan otras opiniones y acepta sugerencias.	En situaciones de desacuerdos o conflicto, al escuchar otras opiniones y acepta sugerencias, propone alternativas para la solución de forma colaborativa y promueve el consenso.	
Seguimiento del tema 20%	Se desconcentra o realiza actividades fuera del tema o actividad, lo que incumplen con el tiempo programado.	Se mantienen en el tema o actividad algunas veces, del tiempo programado.	Se mantienen en el tema o actividad la mayor parte del tiempo programado.	Se mantienen en el tema o actividad en el tiempo programado y revisan con cautela sus avances y progreso.	
Uso del tiempo 10%	Ausencia de presentar los productos realizados por cada miembro del equipo provocando que otros asuman sus responsabilidades para cumplir con los tiempos establecidos.	Tiende a demorarse en la presentación de sus productos como miembro del equipo en las fechas establecida, según perjudicando el cumplimiento de la meta esperada.	Presenta cada uno de los miembros los productos realizados al ser responsable con los tiempos establecidos.	Cada miembro es organizado y presentan sus productos a tiempo, más bien colaboran con otros miembros que muestra atraso o requieren apoyo, para cumplir con las fechas establecidas.	