Instituto tecnológico de Costa Rica

Campus San Carlos

Unidad de computación

Análisis de algoritmos, grupo #50

Proyecto #2

“Estrategias de diseño”

Estudiantes:

Porras Morera Fabricio

Solís Mora Carlos

Profesora:

Valerio Solís Lorena

Fecha de entrega:

23 de noviembre del 2022

Segundo semestre 2022

Tabla de contenidos

Introducción

El azar está presente en nuestras vidas en todo tipo de cosas y contextos, tan común, que por lo general no nos detenemos a pensar que muchas de las cosas que nos ocurren día con día cuentan con una probabilidad implícita de que ocurran; desde la posibilidad de ganar la lotería, hasta el si va a llover el día de hoy cuentan con un punto de aleatoriedad que puede llegar a ser analizado, en el caso del clima, el informe meteorológico de cualquier noticiero determinado se basa en el análisis de los datos para poder llegar a la conclusión de que existe un margen de que llueva o no. Con este concepto en mente, es que el aplicar los algoritmos al análisis de grandes cantidades de posibilidades toma un rol protagónico, ya que las grandes cantidades de información que no pueda procesar la mente humana, si lo puede llegar a hacer una computadora.

En este proyecto se estarán analizando la manera en la que los programadores a lo largo del mundo, lidian con un problema en el cual se necesita dar la mejor de las soluciones, la cual por supuesto que presenta mejores o peores métodos para llegar a ella, pero que al fin y al cabo utilizan el análisis de los datos de manera diferente para llegar a una aproximación más o menos exacta, y es el uso de los algoritmos probabilísticos.

Estos algoritmos, primeramente, deberán cumplir con una serie de condiciones iniciales para poder ser utilizados, y posterior a ello, se deberá realizar un análisis de su eficiencia y comportamiento. Para ello se aplicará un análisis empírico sobre los 3 algoritmos generados, donde se obtendrán la cantidad de asignaciones y comparaciones que estos están realizando en cada ejecución del programa. Con esto se determinará el factor de talla, el cual es fundamental en este proceso para determinar de forma aproximada, el comportamiento que está presentando ese algoritmo específico y así determinar su complejidad.

Seguidamente se realizará también un análisis gráfico del comportamiento de cada algoritmo, en el cual se representa de forma visual el comportamiento del algoritmo analizado en el software de elección, y se podrán 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 se entorpezca el funcionamiento de un computador, 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.

Análisis del problema

En el caso de este proyecto, se plantea el problema de realizar 3 diferentes tipos de algoritmos probabilísticos, los cuáles por supuesto presentarán un diferente conteo de asignaciones y comparaciones, el cual deberá de ser procesado, así como analizado posteriormente a su creación.

Estos algoritmos deberán analizar una lista de alimentos la cual el usuario deberá elegir a su gusto, y una vez hecho esto, deberá llegar a la mejor combinación de alimentos, la cuál satisfaga un mínimo de calorías y un máximo de peso también impuesto por el propio usuario. Debido a que los 3 algoritmos deben trabajar de manera diferente, y el nivel de complejidad en su creación debe de ser distinto, se buscan y se plantean diferentes cosas dentro del proyecto para cada uno de estos.

Primeramente se el hecho de crear un algoritmo que presente una solución de tipo voraz, con la cual ya se había trabajado previamente en el curso. El algoritmo generado debe de ser capaz de conseguir una muy buena solución en un margen de error determinado. Debido a las características que presentan los algoritmos de este tipo, no se espera que siempre se llegue a la mejor solución posible, puesto que este algoritmo prioriza la eficiencia sobre la perfección de la respuesta.

Por otro lado, también se plantea la utilización de un tipo de algoritmo muy comúnmente utilizado en este tipo de problemas, que son los algoritmos de tipo backtracking. Estos algoritmos trabajan de una manera que se podría decir contraria o antagónica, a los algoritmos de tipo voraz, puesto que los algoritmos backtracking priorizan la perfección de la respuesta por encima de la eficiencia del código, muy similar a como es la naturaleza de los algoritmos de fuerza bruta, solo que no llegando a ese nivel. Debido a que los algoritmos

backtracking presentan estas características, se esperan conteos de asignaciones y comparaciones bastante elevados.

Por último está el tercer tipo de algoritmo propuesto para solucionar el problema planteado, el cual es el algoritmo de tipo genético. Estos algoritmos toman los conceptos aplicados por la famosa teoría de la evolución de Charles Darwin y los aplican en la programación, para realizar un descarte de muchas posibilidades y solo quedarse con las mejores combinaciones. Estos algoritmos utilizan conceptos como lo son el de la mutación, el cruce de dos padres, y la regla del más fuerte o el mejor adaptado, lo que permite que a lo largo de la creación de varias generaciones, solo vayan quedando en la población elegida los individuos más capaces de sobrevivir, o puesto en palabras de lo que se necesita para este proyecto, los individuos que más se acercan a la mejor solución al problema planteado.

Solución del problema

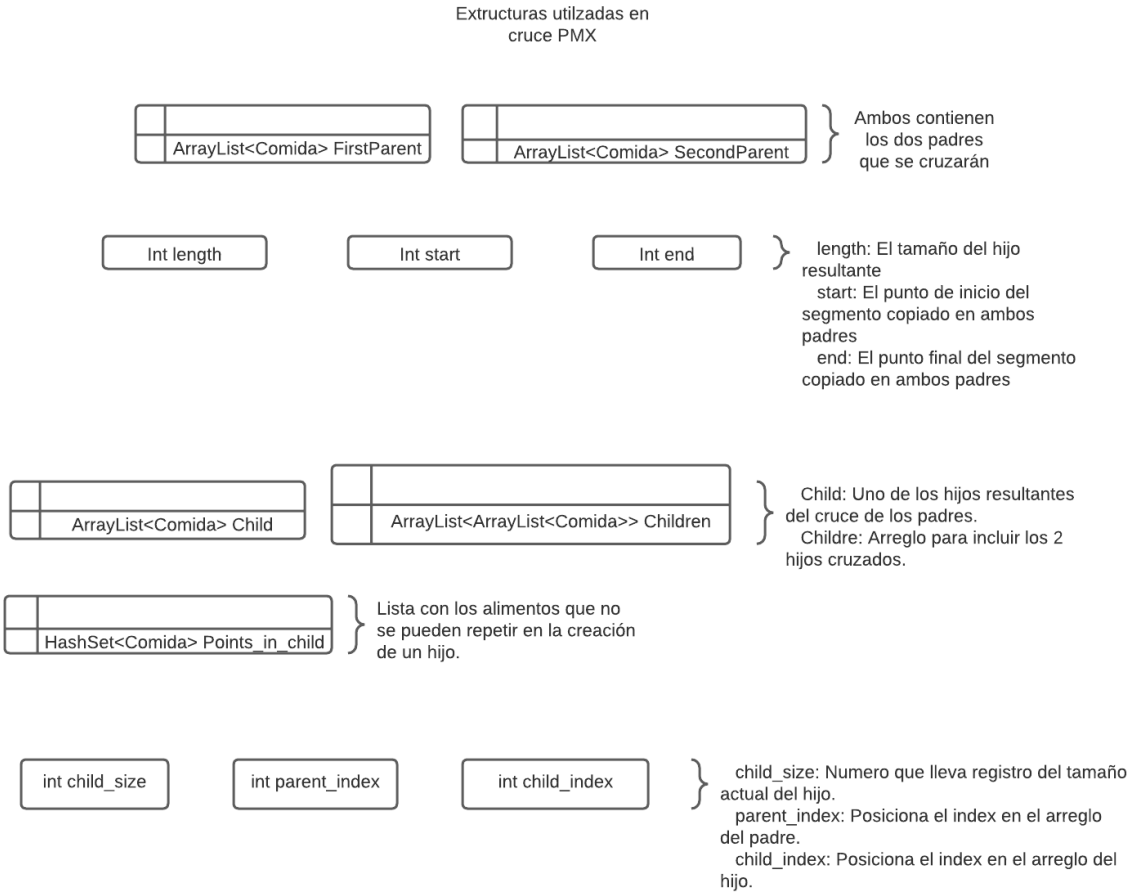
Estructuras

Estructuras Utilizadas en el algoritmo voraz

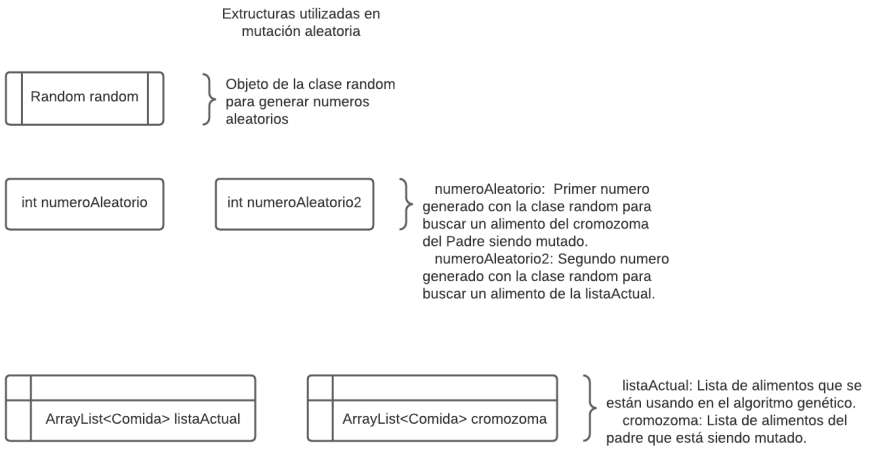
Estructuras Utilizadas en el algoritmo backtracking

Al igual que el algoritmo voraz, el backtracking solo utiliza los ArrayList como estructura de almacenamientos de las instancia tipo ‘Comida’

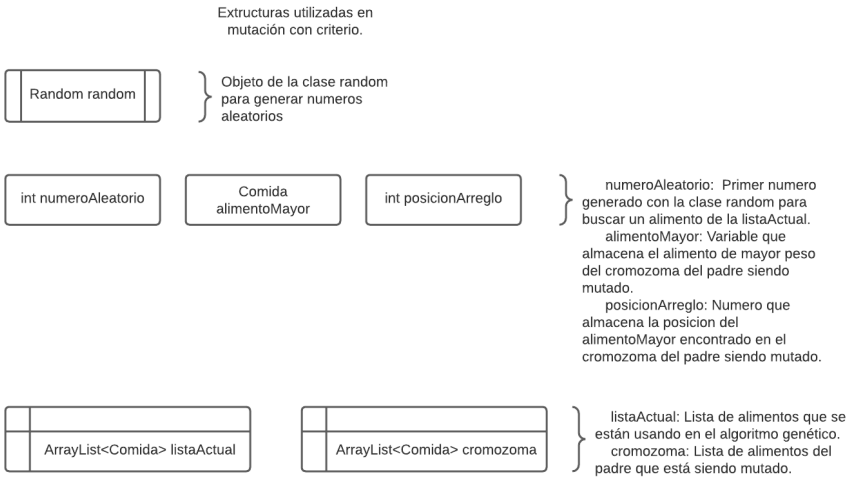
Estructuras Utilizadas en el Cruce PMX



Estructuras Utilizadas en la Mutación Aleatoria



Estructuras Utilizadas en la Mutación con Criterio



Diagramas de flujo

Diagrama de Flujo del Algoritmo Voraz

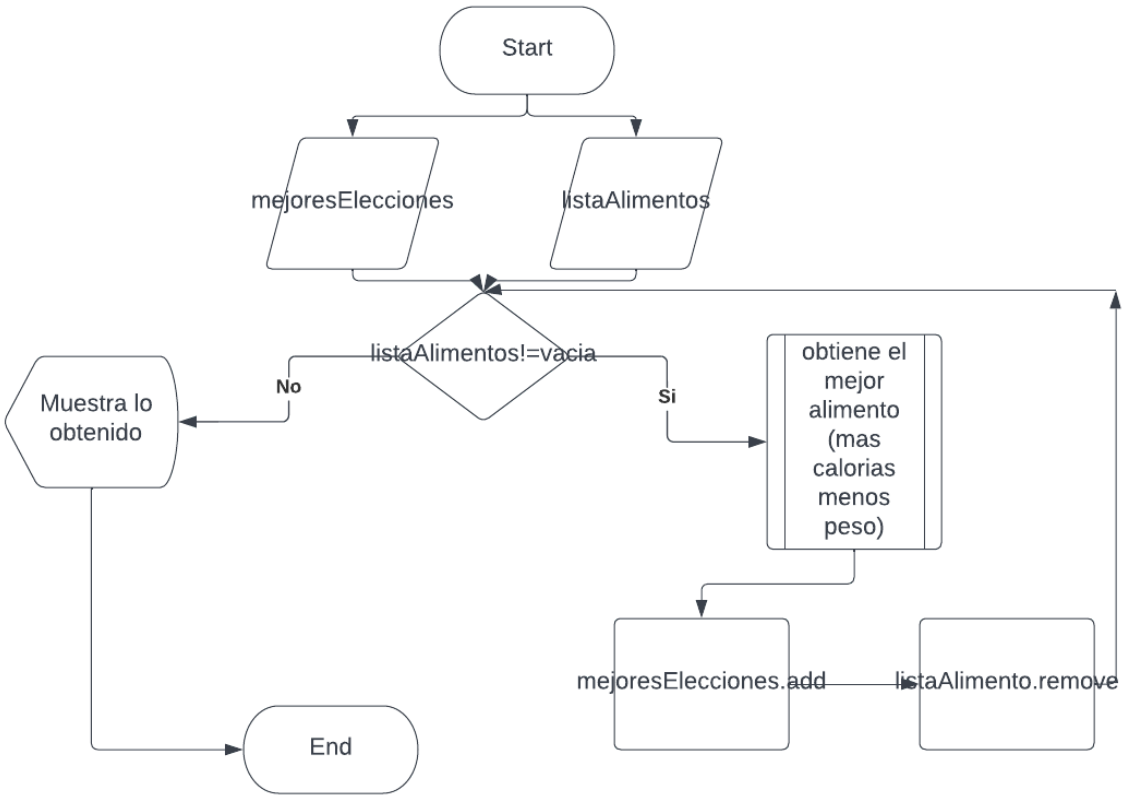


Diagrama de Flujo del Algoritmo BackTracking

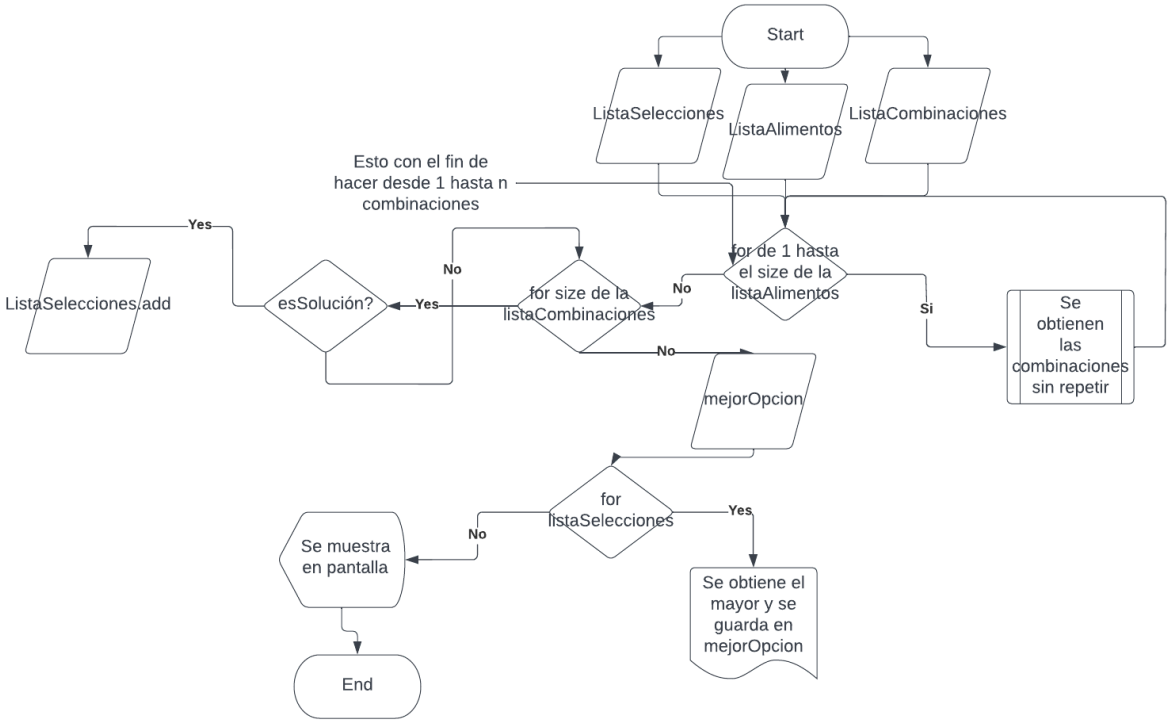
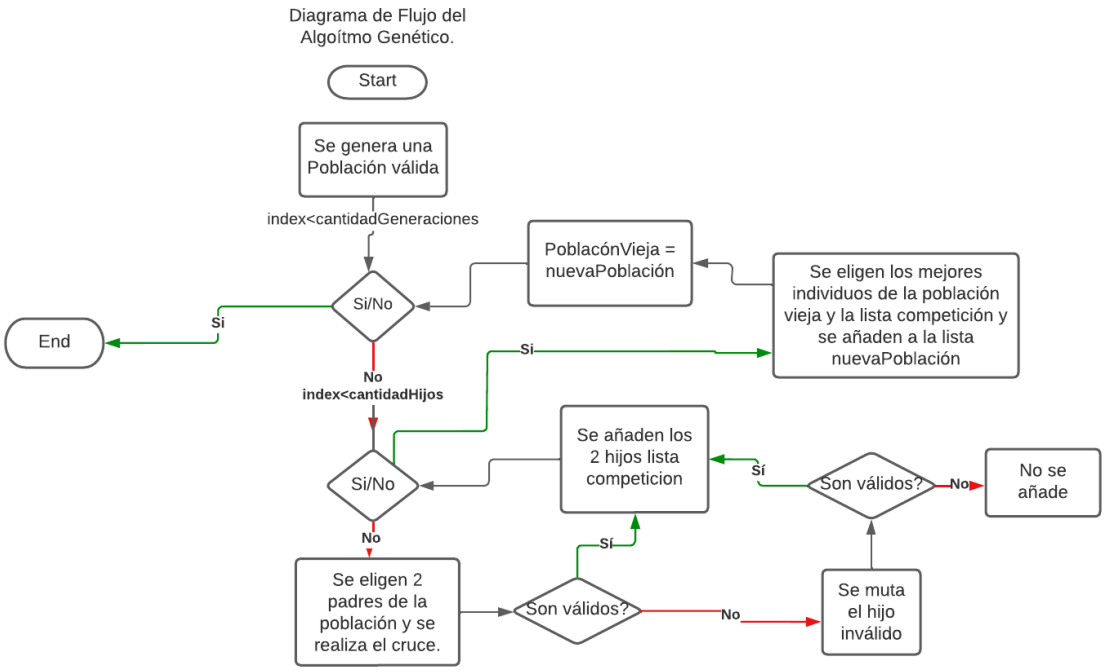


Diagrama de Flujo del Algoritmo Genético



Descripción de los Algoritmos Realizados

Estrategía voraz aplicada

La estrategia voraz aplicada para obtener la lista de alimentos más óptima es el recorrer toda la lista y obtener el alimento que tiene la mayor cantidad de calorías en el menor peso posible, y ese va a ser nuestro candidato electo el cual se agrega de primero a la lista de alimentos seleccionados, reiteradamente se vuelve a buscar el siguiente elemento y así hasta que uno sobrepase el límite de peso, por lo que se elimina y se intenta con el siguiente para ver si alguno aún es admisible y respeta los márgenes preestablecidos.

Estrategia backtracking aplicada

En el algoritmo backtracking lo que se utilizó fue un método de combinaciones sin repetir elementos gracias a (Gul, 2022) lo cual mejora significativamente el algoritmo en sí

puesto que no se hacen combinaciones innecesarias las cuales tienen diferente orden pero siguen teniendo los mismos valores. Luego de obtener este método, lo adaptamos para que funcionara con ArrayList y la clase Comida, entre otras modificaciones para que haga todas las combinaciones desde 1 hasta la cantidad de alimentos en la lista. Luego verifica cada elemento para ver si es solución, con solución se refiere a que respete los márgenes establecidos.

Por último, al tener una lista con alimentos que sí cumplan con los criterios, se busca entre todos el que tenga mayor calorías y menor peso y este será el mejor candidato entre todas las combinaciones posibles.

Función fitness

La función fitness aplicada a cada uno de los individuos pertenecientes a las diferentes generaciones realizadas se basa en el hecho de conseguir el mejor resultado, en el menor número de alimentos posibles. La fórmula puesto a niveles prácticos, es restarle al valor del total de calorías de cada individuo, el valor total del peso que este presenta, y ese número, dividirlo por la cantidad de alimentos con las que cuenta y le generaron esos valores. De esta manera, se está representando el hecho de que mientras más sea el peso conseguido, la resta sobre su nivel de calorías va a ser más grande, y por lo tanto el puntaje va a ser peor, y la división entre su número de alimentos permite representar el hecho de que, mientras más alimentos tenga esa combinación, menor será el puntaje final. Por lo que el mejor resultado posible se logrará obteniendo la mayor cantidad de calorías, menos la menor cantidad de peso, entre la menor cantidad de alimentos.

Tipo de cruce

Para el caso de este proyecto se utilizó el tipo de cruce conocido como el cruce PMX, por sus siglas en inglés, las cuales significan “Partially Mapped Crossover”. Se eligió realizar el cruce de los distintos padres que formarían parte de las poblaciones a lo largo del

recorrido del algoritmo genético con este tipo de cruce, ya que es conocido por ser el tipo de cruce que no genera descendientes con valores repetidos bajo ningún contexto posible. Es sin lugar a dudas uno de los cruces más seguros al momento de realizar soluciones al tipo de problema que se plantea en este proyecto, por lo que al momento de deliberar cuál cruce se elegiría, se decidió tomar la opción más conservadora y realizar una apuesta segura utilizando el cruce PMX.

Tipo de mutación no aleatorio

El tipo de mutación no aleatorio, se hizo basándose en el criterio de seleccionar el alimento de mayor peso dentro del conjunto de alimentos de un individuo, para luego cambiarlo por otro alimento de la lista de alimentos que está utilizando el programa en ese momento y ver si la puntuación aplicada mejora. Debido a que la función fitness utilizada en este programa disminuye la puntuación mayormente, debido a la resta que aplica el peso de los alimentos sobre el total de calorías, con esta mutación se trata de conseguir disminuir en la medida de lo posible ese principal factor que es el que afecta el puntaje final.

Análisis de resultados

A continuación se presenta un análisis de los datos obtenidos en la etapa de medición empírica y gráfica de los algoritmos. Se llevará a cabo una comparación del rendimientos de los 3 algoritmos que utilizan diferentes estrategias de diseño como lo serían la estrategia voraz, backtracking y genética, respectivamente. Con el fin de poder obtener conclusiones al respecto, y dar un reporte claro de los resultados obtenidos a lo largo del desarrollo de este proyecto.

Resultados Finales

Previamente 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 casi completado en su totalidad, el código de cada algoritmo es capaz de obtener de manera eficaz y respetando el encapsulamiento de su estrategia de diseño la lista de los alimentos que cumplen con los requisitos de calorias y peso ya establecidos con anterioridad.

Cabe recalcar que el proyecto solo cuenta con un tipo de cruce el cual sería el PMX, que contiene cada uno de sus apartados correspondientes de los cuales son; generar una población inicial de manera aleatoria y que cumpla con los criterios establecidos, igualmente cuenta con una función fitness que se encarga de obtener los individuos más prometedores y por último los dos tipos mutación, una cambia un alimento de manera aleatoria por otro al azar sin que se encuentre repetido y la otra selecciona el alimento con más peso y lo cambia por uno aleatorio sin que se encuentre repetido igualmente. Únicamente este cruce se logró realizar por falta de tiempo y debido a que no se logró encontrar otro que funcione de manera eficaz y que no cause problemas de duplicación de alimentos al momento de cruzar los padres.

Cada uno de los algoritmos 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 entra en cualquiera de los 3 algoritmos clave, se calcula el tiempo de ejecución que ese algoritmo en específico tardó en realizar el proceso de encontrar la lista de alimentos que cumplen los requisitos, y se realiza su cálculo con una precisión de 3 decimales para mayor exactitud y en milisegundos.

El proyecto carece de la medición empírica en el conteo de memoria total consumida debido a falta de conocimiento del tema y complejidad al momento de investigar sobre el peso de algunas variables, de las cuales se pueden destacar; el peso de un ArrayList indefinido e igualmente el peso de una variable tipo Hash entre otras.

El proyecto cuenta con una parte visual, la cual especifica el algoritmo ejecutado, cuánto fue el tiempo transcurrido desde que se ingresó en la estrategia de diseño, la cantidad de asignaciones y comparaciones que se realizó hasta que se encuentra la solución y a su vez la solución en sí.

Por último, si se quisiera mencionar algún aspecto a mejorar del código aparte de las únicas dos partes incompletas mencionadas anteriormente, sería muy complicado ya que en términos generales los algoritmos funcionan a la perfección utilizando su patrón de diseño adecuadamente y se llevó un orden adecuado para que no hayan ejecuciones innecesarias, variables sin usar y que todo esté documentado internamente para el entendimiento correcto en su completitud de cualquier persona externo a los desarrolladores del proyecto.

Mediciones empíricas y su respectivo factor de crecimiento

Medición empírica

Nombre del algoritmo #1: <u>Algoritmo Voraz</u>

Tamaño	**Asignaciones**	**Comparaciones**	**Total líneas ejecutadas**	**Tiempo**
6	141	124	265	0,000 milisegundos
8	214	196	410	0,000 milisegundos
12	401	385	786	0,000 milisegundos
16	628	629	1.257	0,000 milisegundos

Nombre del algoritmo #2: <u>Algoritmo Backtracking</u>

Tamaño	**Asignaciones**	**Comparaciones**	**Total líneas ejecutadas**	**Tiempo**

6	6.227	3.525	9.752	0,001 milisegundos
8	38.689	22.633	61.322	0,000 milisegundos
12	1.524.479	1.052.233	2.576.712	0,022 milisegundos
16	36.784.010	25.435.299	62.219.239	0,281 milisegundos

Nombre del algoritmo #3: <u>Algoritmo Genético (Cruce PMX)</u>

Tamaño	**Asignaciones**	**Comparaciones**	**Total de líneas ejecutadas**	**Tiempo**
6	269.294	170.848	440.142	0,062 milisegundos
8	10.125.605	6.538.919	16.664.524	0,372 milisegundos

12	39.897.390	43.120.976	83.018.366	0,886 milisegundos
16	42.091.496	48.783.103	90.874.599	0,898 milisegundos

Determinar el factor de crecimiento

Nombre del algoritmo #1: Algoritmo Voraz

Cantidad alimentos	**Factor talla**	**Factor asignaciones**	**Factor comparaciones**	**Factor total líneas ejecutadas**	**Factor tiempo**
8/6	1,33	214/141=1,51	196/124=1,58	410/265=1,54	0
12/6	2	401/141=2,84	385/124=3,10	786/265=2,96	0
16/6	2,66	628/141=4,45	629/124=5,07	1.257/265=4,74	0

16/8	2	628/214=2,93	629/196=3,20	1.257/410=3,06	0

Análisis de comportamiento	**Clasificación en notación O grande**
Según el factor de talla y comparándolo con los demás factores, se ve que adapta un comportamiento lineal, pero un poco más que eso.	O(n*log(n))

Nombre del algoritmo #2: <u>Algoritmo Backtracking</u>

Cantidad alimentos	**Factor talla**	**Factor asignaciones**	**Factor comparaciones**	**Factor total líneas ejecutadas**	**Factor tiempo**
8/6	1,33	38.689/6.227=6,21	22.633/3.525=6,42	61.322/9.752=6,28	0
12/6	2	1.524.479/6.227=244,81	1.052.233/3.525=298,50	2.576.712/9.752=264,22	22
16/6	2,66	36.784.010/6.227=5.907,18	25.435.299/3.525=7.215,68	62.219.239/9.752=6.380,15	281
16/8	2	36.784.010/38.689=950,76	25.435.299/22.633=1.123,81	62.219.239/61.322=1.014,63	0

Análisis de comportamiento	**Clasificación en notación O grande**
Según el factor de talla y comparándolo con los demás factores, se ve que mantiene un crecimiento excesivamente grande por lo que tiene que ser => n!	O(n^n)

Nombre del algoritmo #3: <u>Algoritmo Genético (Cruce PMX)</u>

Cantidad de alimentos.	**Factor talla.**	**Factor asignaciones.**	**Factor comparaciones.**	**Factor total de líneas ejecutadas.**	**Factor tiempo.**
8/6	1,33	10.125.605/269.294= 37,55	6.538.919/170.848= 38,27	16.664.524/440.142= 37,86	0,372/0,062= 6
12/6	2	39.897.390/269.294= 148,15	43.120.976/170.848= 252,39	83.010.366/440.142= 188,59	0,886/0,062= 14,29
16/6	2,66	42.091.496/269.294= 156,12	48.783.103/170.848= 285,53	90.874.599/440.142= 206,46	0,898/0,062= 14,48

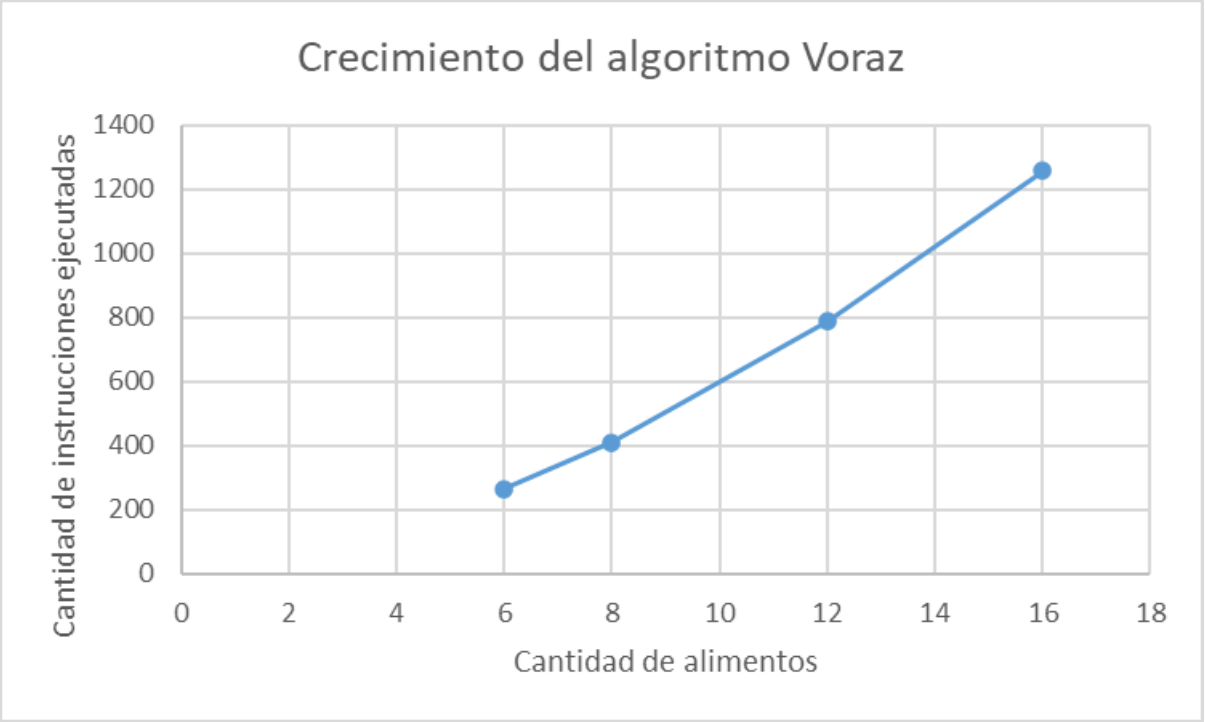
16/8	2	42.091.496/10.125.605= 5,15	48.783.103/6.538.919= 7,46	90.874.599/16.664.524= 5,45	0,898/0,372= 2,41

Análisis de comportamiento	**Clasificación en notación O grande**
Según el factor de talla y comparándolo con los demás factores, se ve que mantiene un crecimiento excesivamente grande al igual que el algoritmo backtracking pero este crecimiento será menor a futuro que el backtracking por lo que no puede ser igual la clasificación que el anterior.	O(n!)

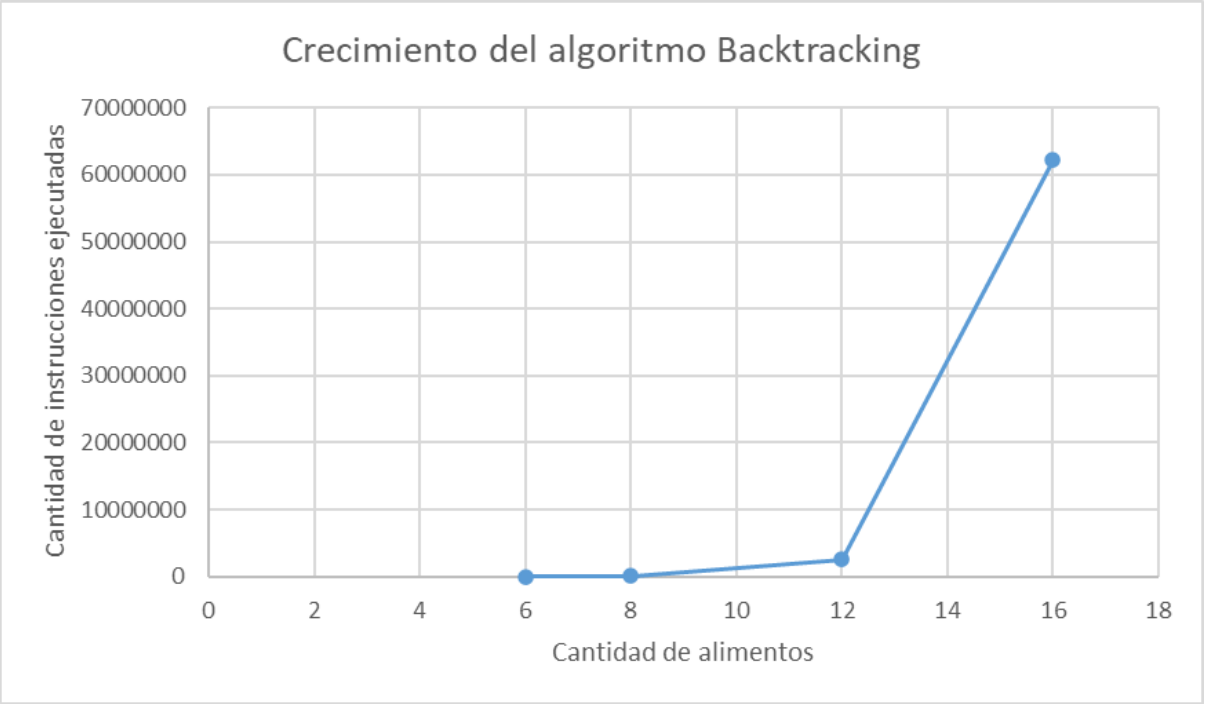
Medición gráfica

Las siguientes mediciones gráficas se realizaron tomando el total de líneas ejecutadas las cuales equivalen a la suma entre las asignaciones y comparaciones.

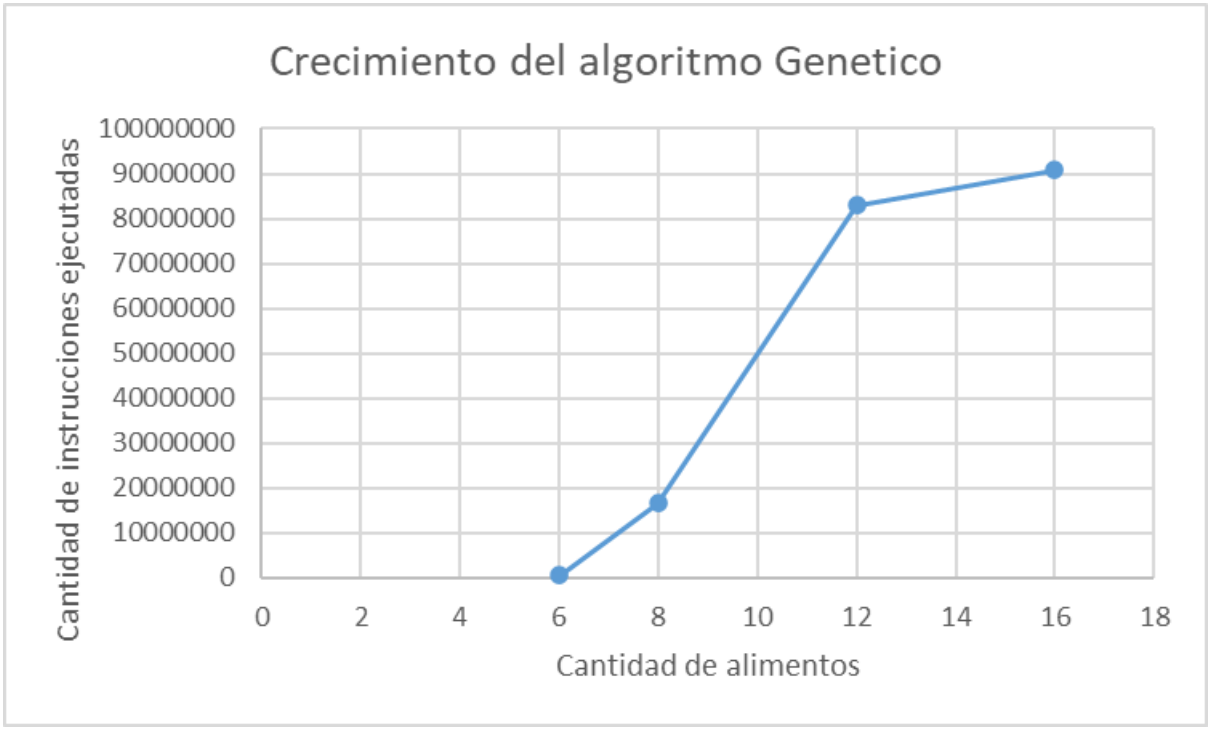
Nombre del algoritmo #1: <u>Algoritmo Voraz</u>



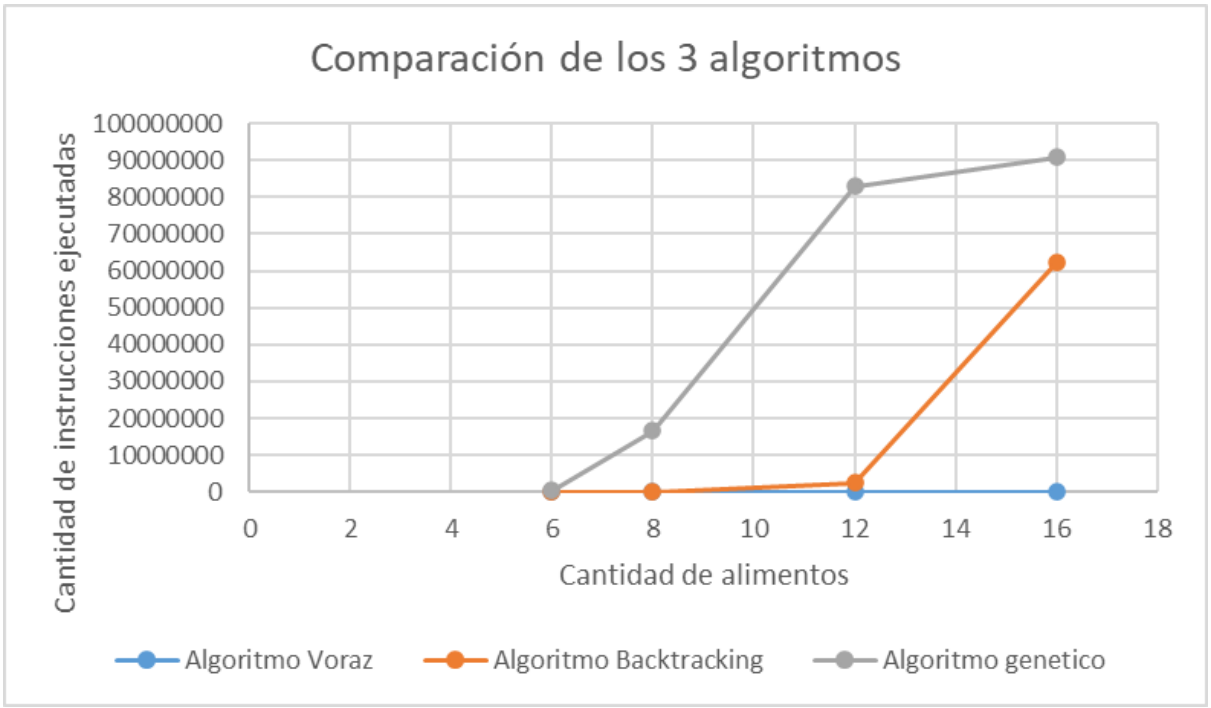
Nombre del algoritmo #2: <u>Algoritmo backtracking</u>



Nombre del algoritmo #3: <u>Algoritmo Genético (Cruce PMX)</u>



Comparación entre los 3 algoritmos



Conclusiones

La medición realizada deja un claro ganador por encima de los otros dos algoritmos utilizados, y es el algoritmo voraz, ya que, como se venía adelantando, los algoritmos voraces trabajan de una forma en la que no se busca la perfección de la respuesta, así que mejor se enfocan en dar una respuesta muy buena mientras la realizan de una manera que resulte a los recursos de la computadora en la que se está ejecutando, mucho más saludable. El algoritmo genético, a pesar de que en un principio parece dar muy malos resultados, alcanza un punto en el que su crecimiento se ve enormemente reducido, mientras que el backtracking presenta un crecimiento mucho más descontrolado.

Sin lugar a dudas la mutación que funciona de mejor manera para salvar a los individuos inválidos, es la mutación con criterio aplicada. Esto debido a que el peso es, con diferencia, el factor que más afecta al hecho de descartar individuos. Cuando un individuo excede de peso indicado, el hecho de quitarle el alimento que más peso cuenta por lo general siempre salva al individuo, puesto que su peso no suele exceder por demasiado el máximo indicado, así que el criterio aplicado resulta ser realmente eficiente y mantiene con vida a muchos individuos que pueden ayudar a generar la mejor solución a futuro.

En este aspecto, el algoritmo voraz es imbatible por su comportamiento logarítmico. Sin embargo, cuando se habla del factor de talla se ve rápidamente como al comparar el algoritmo de backtracking y el genético, el genético se va volviendo mucho más eficiente mientras más elevada sea la cantidad de valores, se puede concluir a simple vista que el backtracking es un algoritmo cuya curva de crecimiento no presenta mucha mayor disminución mientras mayor sea su valor de entrada, mientras que el algoritmo genético no presenta un crecimiento tan exponencial, y respeta unos límites los cuales lo hacen mucho más estable.

Recomendaciones

Se recomienda que, al momento de trabajar con problemas como el planteado en este proyecto, se realice una investigación previa sobre el funcionamiento de los diferentes algoritmos que se van a realizar como propuesta de solución, ya que teniendo unas buenas bases sobre cómo funciona un algoritmo, el desarrollo del mismo será un proceso mucho más sencillo, sin importar si el contexto en el cual se aplicará el algoritmo cambia, ya que la idea central se mantiene y con eso realizar las modificaciones pertinentes no es mucho mayor problema.

En segunda instancia, cuando se está trabajando en proyectos de este estilo, se recomienda que en la medida de los posible, se tomen muestras de datos con la mayor cantidad de datos de entrada posibles, esto debido a que, usualmente los algoritmos pueden llegar a engañar en cuanto a lo que su comportamiento respecta, y un algoritmo que podría parecer realmente ineficiente, puede llegar a ser mucho mejor que sus contrincantes llegados hasta cierto tamaño de los datos de entrada, como se pudo ver en el caso del algoritmo genético y de backtracking de este proyecto; por lo que aumentando los datos de entrada lo más que se pueda ayudaría a disminuir problemas como lo pueden ser las equivocaciones, y el margen de error causado por las equivocaciones humanas sería rápidamente desmentido, por los datos objetivos que muestra el programa, una vez es ejecutado.

Por último, se recomienda aplicar una solución al problema como la planteada en proyecto, bajo nuevos contextos en los que se puede incluir la naturaleza de este tipo de algoritmos. En este proyecto se realizó un análisis de las mejores combinaciones de alimentos, sin embargo se puede aplicar para nuevas vertientes como, las mejores posibilidades para ganar en un juego de cartas, los mejores movimientos en momento de una partida de ajedrez determinada, y en general en una gran cantidad variantes de las cuales los

resultados podrían cambiar, y que por lo tanto, merecen ser analizadas así como apropiadamente documentadas mejorar el conocimiento en este ámbito.

Referencias

Galán, S. S. (2013, junio 3). Tabla de calorías de todos los alimentos. Recetas de cocina fáciles paso a paso | Recetas del Señor Señor | Recetas de cocina fácil del Señor Señor. Encuentra recetas en este blog de cocina. Recetas caseras con explicaciones fáciles y con fotografías paso a paso; Recetas de cocina fáciles paso a paso | Recetas del Señor. https://recetasdecocina.elmundo.es/tabla-calorias

Gul, S. (2022, mayo 31). Genera todas las combinaciones posibles en Java. Delft Stack. https://www.delftstack.com/es/howto/java/java-combinations/

iProfesional, P. (s/f). Cuántas calorías debo consumir para adelgazar. iProfesional. Recuperado el 23 de noviembre de 2022, de https://www.iprofesional.com/health-tech/310123-cuantas-calorias-debo-consumir-para-adelgazar

Shelley, B. (2019). *PmxCrossover.java*. Recuperado de: https://github.com/Brett-Shelley/EC-Assignment-1/blob/master/PmxCrossover.java