Segundo Proyecto – Algoritmos y estructuras de Datos 2

Realizadores del proyecto:

* Diego Alfaro
* Alonso Vega

# Descripción del problema

Para el proyecto presente se pidió que se cumplieran varios objetivos y que se pudieran realizar diversas operaciones y consultas. Básicamente, el proyecto consiste en una aplicación de escritorio que se encarga de manejar sistemas de grafos, utilizando estructuras conocidas como vértices y arcos, además de objetos, para la solución del problema del viajero y de la mochila; lectura de archivos de texto plano, algoritmos de varias categorías y mediciones en ellos. El código fuente de la aplicación debe estar escrito en el lenguaje de programación orientado a objetos conocido como C++, y se debe aprovechar esta característica mencionada.

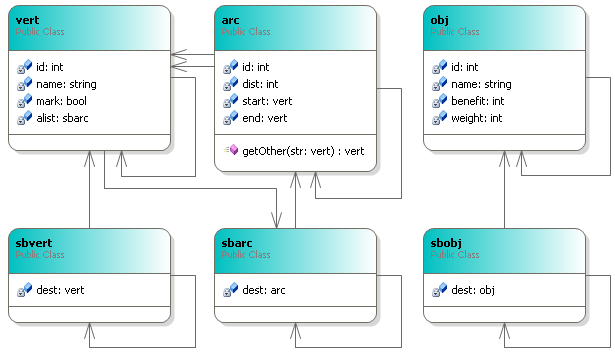
La idea general es que la aplicación pueda abrir archivos de texto plano con un formato definido, del cual se leerán varias listas: vértices, arcos y objetos. Usando esta información, se construirá un grafo y una lista de objetos. Una vez que están listos, el usuario puede seleccionar una serie de algoritmos para buscar la solución de cada problema.

En el caso del viajero, se debe recorrer el grafo dado un punto inicial y se deben recorrer todos los vértices, llegando al origen nuevamente. Lo que se busca es encontrar la ruta con la que se obtiene la menor distancia. El problema de la mochila consiste en seleccionar de una lista de objetos con una cantidad definida de beneficio y de peso, los que puedan brindar la mayor cantidad de beneficio sin pasarse de un máximo establecido para la mochila.

# Solución del problema

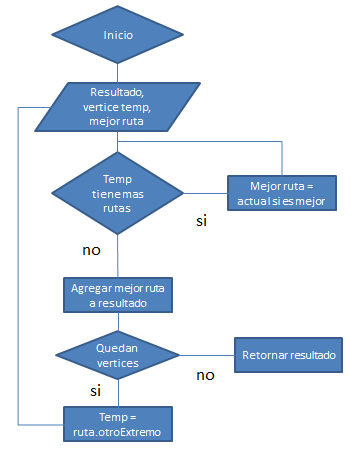
## Diagrama de clases

Estas son las clases básicas utilizadas. Consisten en 3 clases principales: vértice, arco y objeto; y otras 3 clases asociadas como sublistas, correspondientes a las principales.

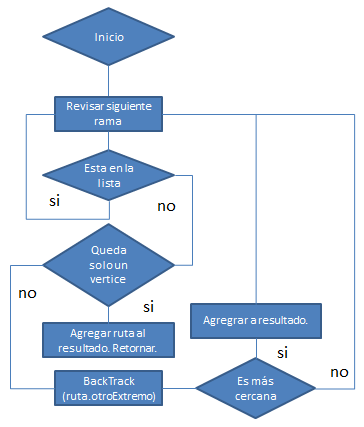


## Diagramas de flujo de algoritmos

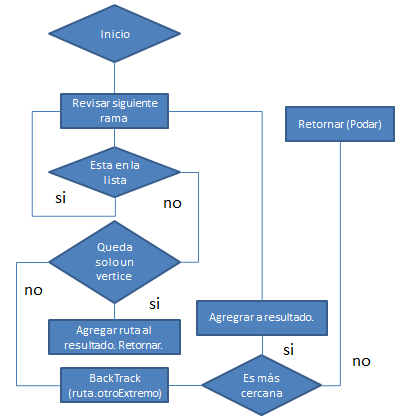
**Problema del viajero: Método voraz**



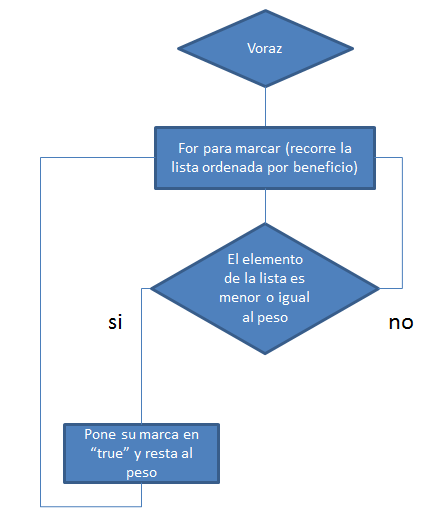
**Problema del viajero: Método de Back Tracking**



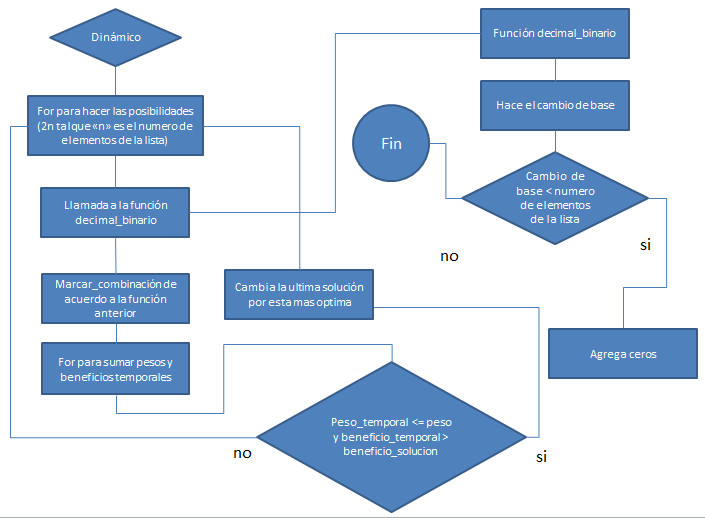
**Problema del viajero: Método de Ramificación y poda**

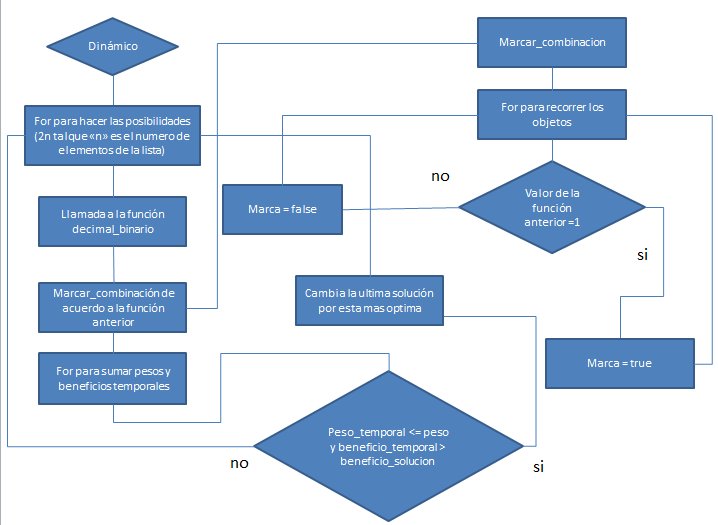


**Problema de la mochila: Voraz**



**Problema de la mochila: Programación dinámica**





## Análisis empírico de costes del problema del viajero en un grafo de transporte conexo implementado con varias estrategias de diseño de algoritmos:

## Voraz, Back tracking, Ramificación y Poda

*Tabla 1.1: Método de Viajero:* ***Estrategia voraz***

| Operaciones |
| --- |
| Cantidad de arcos | | |
| Primer tamaño | Segundo tamaño | Tercer tamaño |
| Asignaciones | 51 | 60 | 137 |
| Comparaciones | 45 | 65 | 135 |
| Cantidad de líneas de código ejecutadas | 96 | 125 | 272 |
| Cantidad de líneas de código. | 29 | | |
| Cantidad de memoria consumida | 64 bytes | 64 bytes | 124 bytes |

¿Según los resultados que has obtenido, cómo se comporta el algoritmo del viajero en diseño voraz en función de de los tamaños del grafo? Marcar con X la respuesta que coincida con tu caso.

| (a) | El tamaño no importa a nivel de costes |  |
| --- | --- | --- |
| (b) | El tamaño afecta sólo al número de comparaciones |  |
| (c) | El tamaño afecta sólo al número de asignaciones |  |
| (d) | El tamaño afecta a los dos tipos de operaciones | **X** |
| (e) | El tamaño afecta la cantidad de memoria consumida | **X** |
|  |

*Tabla 1.2: Factor de incremento del número de operaciones al crecer el tamaño (en cantidad de arcos)*

| Talla | Factor talla | Factor Asig | Factor Comp | Cantidad de memoria consumida | Cantidad de líneas ejecutadas |
| --- | --- | --- | --- | --- | --- |
| Del primer al segundo tamaño | 2 | 1 | 2 | 1 | 1 |
| Del segundo al tercer tamaño | 1.5 | 2 | 2 | 2 | 2 |

El comportamiento esperado del algoritmo será que, en general:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | O(1) | O(n) | Otro\_\_\_\_\_\_\_\_ |  |
| Las asignaciones crecen |  | **X** |  | Con respecto a la talla |
| Las comparaciones crecen |  | **X** |  | con respecto a la talla |
| Cantidad de memoria consumida |  | **X** |  | con respecto a la talla |
| Cantidad de líneas ejecutadas crece |  | **X** |  | con respecto a la talla |

*Tabla 2.1: Método del viajero:* ***Back tracking***

| Operaciones |
| --- |
| Cantidad de arcos | | |
| Primer tamaño | Segundo tamaño | Tercer tamaño |
| Asignaciones | 112 | 868 | 3470 |
| Comparaciones | 139 | 1255 | 6641 |
| Cantidad de líneas de código ejecutadas | 251 | 2123 | 10111 |
| Cantidad de líneas de código. | 34 | | |
| Cantidad de memoria consumida | 228 bytes | 1880 bytes | 5068 bytes |
| Cantidad de combinaciones posibles. | 2 | 24 | 12 |

¿Según los resultados que has obtenido, cómo se comporta el algoritmo del viajero con el diseño back tracking en función de de los tamaños del grafo? Marcar con X la respuesta que coincida con tu caso.

| (a) | El tamaño no importa a nivel de costes |  |
| --- | --- | --- |
| (b) | El tamaño afecta sólo al número de comparaciones |  |
| (c) | El tamaño afecta sólo al número de asignaciones |  |
| (d) | El tamaño afecta a los dos tipos de operaciones | **X** |
| (e) | El tamaño afecta la cantidad de memoria consumida | **X** |
|  |

.

*Tabla 2.2: Factor de incremento del número de operaciones al crecer el tamaño de los arcos.*

| Talla | Factor talla | Factor Asig | Factor Comp | Cantidad de memoria consumida | Cantidad de líneas ejecutadas |
| --- | --- | --- | --- | --- | --- |
| Del primer al segundo tamaño | 2 | 8 | 9 | 8 | 8 |
| Del segundo al tercer tamaño | 1.5 | 4 | 5 | 3 | 5 |

El comportamiento esperado del algoritmo será que, en general:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | O(1) | O(n) | Otro 5n |  |
| Las asignaciones crecen |  |  | **X** | Con respecto a la talla |
| Las comparaciones crecen |  |  | **X** | con respecto a la talla |
| Cantidad de memoria consumida |  |  | **X** | con respecto a la talla |
| Cantidad de líneas ejecutadas crece |  |  | **X** | con respecto a la talla |

*Tabla 3.1: Método del viajero:* ***Ramificación y Poda.***

| Operaciones |
| --- |
| Cantidad de arcos | | |
| Primer tamaño | Segundo tamaño | Tercer tamaño |
| Asignaciones | 87 | 654 | 2715 |
| Comparaciones | 116 | 981 | 4718 |
| Cantidad de líneas de código ejecutadas | 203 | 1635 | 7433 |
| Cantidad de líneas de código. | 41 | | |
| Cantidad de memoria consumida | 188 | 1528 | 4340 |
| Cantidad de ramas (combinaciones) | 13 | 134 | 439 |
| Cantidad de podas | 5 | 70 | 269 |

¿Según los resultados que has obtenido, cómo se comporta el algoritmo del viajero con ramificación y poda en función de de los tamaños del grafo? Marcar con X la respuesta que coincida con tu caso.

| (a) | El tamaño no importa a nivel de costes |  |
| --- | --- | --- |
| (b) | El tamaño afecta sólo al número de comparaciones |  |
| (c) | El tamaño afecta sólo al número de asignaciones |  |
| (d) | El tamaño afecta a los dos tipos de operaciones | **X** |
| (e) | El tamaño afecta la cantidad de memoria consumida | **X** |
|  |

*Tabla 3.2: Factor de incremento del número de operaciones al crecer el tamaño de la cantidad de arcos.*

| Talla | Factor talla | Factor Asig | Factor Comp | Cantidad de memoria consumida | Cantidad de líneas ejecutadas |
| --- | --- | --- | --- | --- | --- |
| Del primer al segundo tamaño | 2 | 8 | 8 | 8 | 8 |
| Del segundo al tercer tamaño | 1.5 | 4 | 5 | 3 | 5 |

El comportamiento esperado del algoritmo será que, en general:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | O(1) | O(n) | Otro 5n |  |
| Las asignaciones crecen |  |  | **X** | Con respecto a la talla |
| Las comparaciones crecen |  |  | **X** | con respecto a la talla |
| Cantidad de memoria consumida |  |  | **X** | con respecto a la talla |
| Cantidad de líneas ejecutadas crece |  |  | **X** | con respecto a la talla |

*Tabla 4.1: Método de Mochila:* ***Voraz***

| Operaciones |
| --- |
| Cantidad de objetos (vértices) | | |
| Primer tamaño | Segundo tamaño | Tercer tamaño |
| Asignaciones | 8 | 8 | 10 |
| Comparaciones | 11 | 11 | 21 |
| Cantidad de líneas de código ejecutadas | 19 | 19 | 31 |
| Cantidad de líneas de código. | 5 | | |
| Capacidad mochila | 5 | 20 | 20 |
| Cantidad de objetos dentro de la mochila | 3 | 3 | 4 |
| Máximo beneficio dentro de la mochila | 185 | 255 | 139 |
| Suma de los pesos dentro de la mochila. | 5 | 16 | 20 |
| Suma de todos los beneficios. | 260 | 355 | 612 |
| Cantidad de memoria consumida | 4 bytes | 4 bytes | 4 bytes |

¿Según los resultados que has obtenido, cómo se comporta el algoritmo de la mochila con el diseño voraz en función de la cantidad objetos? Marcar con X la respuesta que coincida con tu caso.

| (a) | El tamaño no importa a nivel de costes |  |
| --- | --- | --- |
| (b) | El tamaño afecta sólo al número de comparaciones |  |
| (c) | El tamaño afecta sólo al número de asignaciones |  |
| (d) | El tamaño afecta a los dos tipos de operaciones | **X** |
| (e) | El tamaño afecta la cantidad de memoria consumida |  |
|  |

*Tabla 4.2: Factor de incremento del número de operaciones al crecer el tamaño de la cantidad de objetos.*

| Talla | Factor talla | Factor Asig | Factor Comp | Cantidad de memoria consumida | Cantidad de líneas ejecutadas |
| --- | --- | --- | --- | --- | --- |
| Del primer al segundo tamaño | 1 | 1 | 1 | 1 | 1 |
| Del segundo al tercer tamaño | 2 | 1 | 2 | 2 | 1 |

El comportamiento esperado del algoritmo será que, en general:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | O(1) | O(n) | Otro\_\_\_\_\_\_\_\_ |  |
| Las asignaciones crecen |  | **X** |  | Con respecto a la talla |
| Las comparaciones crecen |  | **X** |  | con respecto a la talla |
| Cantidad de memoria consumida |  | **X** |  | con respecto a la talla |
| Cantidad de líneas ejecutadas crece |  | **X** |  | con respecto a la talla |

*Tabla 5.1: Método de Mochila:* ***programación dinámica.***

| Operaciones |
| --- |
| Cantidad de objetos (vértices) | | |
| Primer tamaño | Segundo tamaño | Tercer tamaño |
| Asignaciones | 1159 | 1159 | 62584 |
| Comparaciones | 753 | 753 | 39911 |
| Cantidad de líneas de código ejecutadas | 1912 | 1912 | 102459 |
| Cantidad de líneas de código. | 20 | | |
| Capacidad mochila | 5 | 25 | 50 |
| Cantidad de objetos dentro de la mochila | 3 | 4 | 8 |
| Máximo beneficio dentro de la mochila | 185 | 275 | 568 |
| Suma de los pesos dentro de la mochila. | 5 | 22 | 46 |
| Suma de todos los beneficios. | 260 | 355 | 612 |
| Cantidad de posibilidades analizadas | 32 | 32 | 1024 |
| Cantidad de memoria consumida | 1392 | 1392 | 65520 |

¿Según los resultados que has obtenido, cómo se comporta el algoritmo de la mochila con el diseño de programación dinámica en función de la cantidad objetos? Marcar con X la respuesta que coincida con tu caso.

| (a) | El tamaño no importa a nivel de costes |  |
| --- | --- | --- |
| (b) | El tamaño afecta sólo al número de comparaciones |  |
| (c) | El tamaño afecta sólo al número de asignaciones |  |
| (d) | El tamaño afecta a los dos tipos de operaciones | **X** |
| (e) | El tamaño afecta la cantidad de memoria consumida | **X** |
|  |

*Tabla 4.2: Factor de incremento del número de operaciones al crecer el tamaño de la cantidad de objetos.*

| Talla | Factor talla | Factor Asig | Factor Comp | Cantidad de memoria consumida | Cantidad de líneas ejecutadas |
| --- | --- | --- | --- | --- | --- |
| Del primer al segundo tamaño | 1 | 1 | 1 | 1 | 1 |
| Del segundo al tercer tamaño | 2 | 54 | 53 | 47 | 54 |

El comportamiento esperado del algoritmo será que, en general:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | O(1) | O(n) | Otro n6 |  |
| Las asignaciones crecen |  |  | **X** | Con respecto a la talla |
| Las comparaciones crecen |  |  | **X** | con respecto a la talla |
| Cantidad de memoria consumida |  |  | **X** | con respecto a la talla |
| Cantidad de líneas ejecutadas crece |  |  | **X** | con respecto a la talla |

¿Los datos de la tabla 1.1, 2.1, 3.1, coinciden en? Marcar con X la respuesta que coincida con tu caso.

| (a) | Coinciden todos los resultados |  |
| --- | --- | --- |
| (b) | Sólo coinciden las asignaciones |  |
| (c) | Sólo coinciden las comparaciones |  |
| (e) | Sólo coincide en la cantidad de memoria consumida. |  |
| (f) | No coinciden ninguno de los resultados | **X** |

Justifica el resultado obtenido en la cuestión anterior

| Al analizar las tablas nos damos cuenta de que los valores no coinciden puesto que cada uno de ellos se ejecuta de diferente manera, haciendo así el mejor resultado según lo planteado en el problema, aunque en el método voraz no siempre va tener una respuesta requerida, pero los otras dos algoritmos utilizados deberían ser igual en cuanto a resultado; pero en asignaciones, comparaciones y cantidad de memoria no lo son puesto que el proceso para llegar a la respuesta que requiere el problema es diferente, es decir, mientras se recorre el grafo en el voraz de una forma en el back tracking lo hace de otras y estas de otra diferentes al ramificación y poda, siendo así las asignaciones, comparaciones y cantidad de memoria, en el ramificación y poda deben de disminuir de una forma gradual conforme aumenta la cantidad de arcos y esto también ocurre en voraz. |
| --- |

¿Los datos de la tabla 4.1, 5.1 coinciden en? Marcar con X la respuesta que coincida con tu caso.

| (a) | Coinciden todos los resultados |  |
| --- | --- | --- |
| (b) | Sólo coinciden las asignaciones |  |
| (c) | Sólo coinciden las comparaciones |  |
| (e) | Sólo coincide en la cantidad de memoria consumida. |  |
| (f) | No coinciden ninguno de los resultados | **X** |

Justifica el resultado obtenido en la cuestión anterior

| Según lo analizado podemos ver que el voraz tiene menos asignaciones, comparaciones y cantidad de memoria que el programación dinámica, ya que el dinámico a pesar de que es el que da la respuesta más óptima sacrifica el uso de almacenamiento, entonces por ende, muchas veces el voraz no va dar la respuesta correcta, contrario al dinámico que siempre lo hace. Y no coinciden puesto que el manejo de las asignaciones, comparaciones y cantidad de memoria es de forma diferente. |
| --- |