Análisis y diseño de algoritmos
Tarea 4
12 de abril de 2020
Dr. Eduardo A. Rodríguez Tello

1. Introducción

El objetivo de esta tarea es diseñar, implementar y analizar dos algoritmos para resolver el problema de la mochila 0/1 (*0/1 knapsack problem*). El primero es un algoritmo de programación dinámica, y el segundo un algoritmo avaro (greedy). Con esto se busca que pongan en práctica los conocimientos adquiridos en el curso sobre estas dos técnicas para el diseño de algoritmos.

Para realizar esta tarea deberán formar equipos de 4 personas (puede haber sólo un equipo con 5 integrantes en el grupo). Cada implementación de los algoritmos solicitados deberá ser lo más eficiente posible, pues competirá contra las implementaciones de los otros equipos. Las calificaciones asignadas a esta tarea dependerán directamente del desempeño mostrado en la competencia por los algoritmos que desarrollen.

Para mayores detalles sobre los algoritmos de programación dinámica y avaros (greedy) pude consultar las siguientes fuentes:

- `https://www.youtube.com/watch?v=xCbYmUPvc2Q&t=799s`
- `https://www.youtube.com/watch?v=nLmhmB6NzcM&t=159s`
- `https://www.youtube.com/watch?v=oTTzNMHM05I`
- `https://www.youtube.com/watch?v=M79iHjAG1tg`

2. Entregables

La entrega (una sola por cada equipo) se realizará por medio de Canvas (Tarea 4). Ésta incluye sólo dos archivos: 1) El PDF del reporte, y 2) Un ZIP con los códigos fuentes y respaldos de los resultados. Los entregables se encuentran descritos a detalle a continuación:

1. Implemente en el lenguaje de programación[1] de su preferencia un algoritmo de programación dinámica y otro avaro (greedy) que permitan resolver instancias del problema de la mochila 0/1 (*0/1 knapsack problem*). Cada implementación debe cumplir con las siguientes características:

   a) Recibe las instancias del problema a partir de la entrada estándar con el siguiente formato: 1) un entero que indica el número total de objetos n de la instancia. 2) una serie de n enteros con los beneficios de cada objeto. 3) una serie de n enteros con los pesos de cada objeto. 4) un entero con la capacidad máxima W de la mochila. Por lo anterior debe ser posible ejecutarlo desde una consola y enviarle una instancia a través de redireccionamiento. Por ejemplo, *ejecutable* < *archivoInstancia.txt*.

   b) Imprime a la salida estándar **únicamente** los siguientes datos: 1) El máximo beneficio encontrado expresado como un entero. 2) La lista de objetos que fueron introducidos en la mochila (1 basados). 3) En el caso del algoritmo de programación dinámica

[1] Recuerde que el lenguaje de implementación puede influir en el tiempo de ejecución de un algoritmo.

también debe imprimir la matriz resultante empleada para almacenar los beneficios de los subproblemas, empleando una línea por cada fila de la matriz y separando cada elemento con un espacio en blanco. Debe ser posible, por lo tanto, usar redireccionamiento para recuperar en un archivo la salida de sus programas. Por ejemplo, *ejecutable > resultados.txt*.

c) En caso de que su programa reciba una entrada con formato distinto al establecido, deberá detener la ejecución y enviar a la salida estándar un mensaje de error indicando claramente el problema detectado. Sea cuidadoso con esta validación, ya que podría ser motivo de descalificación de la competencia el no cumplir con el formato de entrada especificado.

2. Diseñe 10 instancias de prueba para cada uno de sus algoritmos y guarde cada una de ellas en un archivo de texto. Al momento de diseñar sus instancias de prueba, busque generar casos que sean lo más difíciles de resolver y lleven al límite de su capacidad a sus algoritmos. Considere que estas instancias se podrán usar para probar el desempeño de otros equipos competidores. El tiempo máximo de ejecución que se permitirá durante la competencia para los algoritmos es de 3 minutos. Este tiempo será medido con el comando *timeout* de Linux.

   a) En el caso de las instancias para el algoritmo de programación dinámica los archivos deben ser nombrados como dp01.txt, dp02.txt, ..., dp10.txt.

   b) En el caso de las instancias para el algoritmo avaro (greedy) los archivos deben ser nombrados como ga01.txt, ga02.txt, ..., ga10.txt.

3. **Reporte:** el documento **debe** estar en formato PDF e incluir los siguientes puntos:

   - Portada que incluya los nombres completos y matriculas de los miembros del equipo.
   - Introducción
   - Definición formal del problema a resolver
   - Descripción detallada de cada uno de los algoritmos desarrollados, usando pseudocódigos.
   - Análisis matemático de la complejidad temporal y espacial de los algoritmos desarrollados
   - Experimentación con los algoritmos desarrollados. Los resultados de estos experimentos deben ser analizados y discutidos a detalle
     - Describa a detalle las condiciones experimentales: lenguaje de programación empleado, banderas de compilación usadas, características de la computadora donde se realizaron las pruebas, etc.
     - Ejecute una vez cada algoritmo desarrollado sobre cada una de las instancias de prueba que diseñó y registre los tiempos de cómputo.
     - Genere para cada algoritmo una tabla que incluya la siguiente información (columnas): nombre de la instancia, número de objetos n, capacidad máxima de la mochila W, beneficio total y tiempo de cómputo.
     - Documente un ejemplo práctico (encontrado con ayuda de sus programas) donde se observe que su algoritmo de programación dinámica es capaz de resolver de manera óptima una instancia, mientras el algoritmo avaro sólo brinda una solución aproximada. Para ello es probable que requiera diseñar una instancia que ambos

algoritmos puedan resolver. Su ejemplo de incluir figuras donde se observen, la instancia del problema, así como la solución brindada por cada algoritmo.

- Conclusiones globales de la tarea
- Referencias bibliográficas

4. **Código fuente documentado:** Incluye todos los códigos fuente, tanto de programas como de scripts, que hayan empleado en el desarrollo del proyecto. Estos deben estar organizados en carpetas con archivos README.TXT que faciliten la identificación de cada elemento. El archivo debe incluir toda la información necesaria para compilar y ejecutar sus algoritmos.

5. **Respaldo:** Todos los resultados producto de los experimentos realizados en este proyecto deben estar respaldados y perfectamente organizados en carpetas con archivos README.TXT que faciliten la identificación de cada elemento.