Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Centro Académico de Alajuela
Investigación de Operaciones
II Semestre, 2021
Profesor: Carlos Gamboa Venegas

Proyecto 4: Desempeño de Programación Dinámica

Administrativos

Miércoles 10 de noviembre, antes de las 11:59 subir archivo zip al tec-digital conteniendo el programa. Trabajo en grupos (mismos del proyecto 1).

Especificación

Parte 1: Implementación de algoritmos

La primera parte del proyecto la implementación de la solución de cada uno de los problemas usando Python. Para cada corrida debe medir el tiempo de ejecución. Se debe correr varias veces con los mismos datos para tener al final un promedio de todas las corridas y tener resultados más confiables.

Problema 1: Problema del contenedor

Se tienen n elementos distintos, y un contenedor que soporta una cantidad especifica de peso W. Cada elemento i tiene un peso w_i, un valor o *beneficio* asociado dado por b_i. El problema consiste en agregar elementos al contenedor de forma que se maximice el beneficio total sin superar el peso máximo que soporta el contenedor. La solución debe ser dada con la lista de los elementos que se ingresaron y el valor del beneficio máximo obtenido.

Se deben programar tres versiones del algoritmos, el primero es fuerza bruta, también llamada búsqueda exhaustiva. No confundir con backtracking. El segundo es programación dinámica bottom-up (visto en clase), el tercero es programación dinámica top-down (utilizando memoization). El archivo principal debe ser llamada `contenedor.py`

El algoritmo a correr debe ser indicado como parámetros en la linea de entrada del programa. Además, el programa deberá permitir dos maneras de ejecución:

1. recibiendo un archivo con el problema (indicado un la opción `-a`)
2. recibiendo parámetros para generar datos de un problema aleatorio (indicado por `-p`).

Ejemplo de ejecución:
`./contenedor.py algoritmo -a archivo.txt iteraciones`

- `algoritmo` requerido, valores 1=fuerza bruta, 2=pd bottom-up, 3=pd top-down
- `-a archivo.txt` archivo con los datos del problema
- `iteraciones` cantidad de corridas para medir tiempo promedio

Estructura de un archivo de entrada:

> *linea 1: Peso máximo de la mochila*
> *linea 2: elemento i (peso, beneficio) separados por comas*
> *linea n+1: elemento n (peso, beneficio)*

Ejemplo: Con un contenedor de peso W = 30. 5 artículos con pesos: 5, 15, 10, 10, 8, beneficios: 20, 50, 60, 62 y 40. Tiene como solución un beneficio máximo de 162 agregando los artículos 3, 4, y 5.

```
./contenedor.py 1 -a problema1.txt 1
Input: (problema1.txt)
30
5,20
15,50
10,60
10,62
8,40

Output:
Beneficio máximo: 162
Incluidos: 3,4,5
```

La segunda forma de correrlo es indicando parámetros (`-p`) para que se genere un problema de tamaño determinado y con valores aleatorios. Esto con el fin de que se puedan realizar pruebas de problemas con mayor cantidad de elementos, sin la necesidad de tener que crear archivos con valores especificos.

`./contenedor.py algoritmo -p W N rangoPesos rangoBeneficios iteraciones`

- `W` es el tamaño del contenedor
- `N` la cantidad de elementos
- `rangoPesos` rango de pesos a asignar
- `rangoBeneficios` el rango de valores para asignar el beneficio a cada elemento,
- `iteraciones` la cantidad de veces que se debe correr el programa para medir el tiempo. Tomar en cuenta que solo se generar los datos del problema una vez, y el problema se corre la cantidad indicada de iteraciones, para luego calcular un promedio del tiempo. Para finalmente desplegar los resultados junto con la asignación de elementos

Todos los rangos son dados de la forma `Min-Max`, ejemplo `7-25`.

Ejemplo

```
./contenedor.py 2 -p 25 8 4-8 10-14 20
```

Algoritmo 1 de PD con bottom-up, peso total del contenedor 25, 8 elementos, de pesos entre 4 y 8, beneficios entre 10 y 14. Para correrlo 20 veces con los mismos datos generados una única vez.

Parte 2. Experimentos y Resultados 40 puntos

La elaboración de experimentos es sumamente importante en el proyecto. Acá se medirá el desempeño dado como el tiempo ejecución de cada implementación.

1. Se debe usar el generador para crear al menos cuatro problemas de diferente tamaño, esto es, diferente cantidad de elementos y un tamaño de contenedor acuerdo al rango de peso indicado.
2. El programa debe registrar el tiempo de ejecución de cada corrida del algoritmo. Las iteraciones indican cuantas veces se va a correr un programa con los mismos datos, luego de haberlo corrido lo indicado, se debe hacer un cálculo del promedio de todos los valores registrados. Este cálculo debe ser registrado para cada uno de las implementaciones con el objetivo de hacer la comparación de tiempos.
3. Para la comparación se deben realizar gráficos de barras, o alguno otro que permita visualizar los datos promedios y se pueda explicar la comparación usando los medios visuales. Así los gráficos creados deben comparar los 4 problemas de diferente tamaño y el tiempo de ejecución de las implementaciones para cada problema. Puede que existan entonces cuatro gráficos para visualizar cada problema, o todos conjuntos en un solo gráfico.
4. Los gráficos deben ser implementados usando la biblioteca de Matplotlib.

Evaluación

Escritura de código propio, documentación, implementación de algoritmo, resultados correctos, experimentos y tiempo de ejecución. Ejecución de experimentos y gráficos visualmente correctos y entendibles.

Parte 1: implementación de algoritmos	60%
Parte 2: experimentos y resultados	40%