Instituto Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Centro Académico de Alajuela
Investigación de Operaciones
II Semestre, 2020
Profesor: Carlos Gamboa Venegas

Proyecto 2: Desempeño de la Programación Dinámica

Administrativos

Proyecto exclusivamente Individual. Entrega el domingo 6 de diciembre del 2020. Crear un repositorio en github y subir el código en zip al tec-digital la fecha indicada antes de las 11:00 pm.

Especificación

El proyecto consta de dos partes: la primera es el programa computacional que resuelve los problemas indicados más adelante, la segunda parte es un reporte escrito con los resultados de los experimentos ejecutados. Ambas partes se detallan a continuación:

Parte 1: Programa escrito en Python3, donde se implemente la versión en fuerza bruta y programación dinámica para resolver cada problema. El programa se debe ejecutar desde la linea de comandos del sistemas operativo, de la siguiente forma:

```
python3 solver.py [-h] PROBLEMA ALGORITMO ARCHIVO
```

- Opcional `-h`, si se indica, los demás argumentos no deben incluirse, `-h` debe mostrar una guiá de como usar el programa, de los diferentes problemas que resuelve y la forma de los parámetros de los archivos.
- *`PROBLEMA`* valor de 1 o 2, indicando el problema a resolver, 1 contenedor, 2 alineamiento.
- *`ALGORITMO`* valor de 1 o 2, indicando el algoritmo a usar, 1 fuerza bruta, 2 programación dinámica.
- `ARCHIVO` indica el archivo de entrada donde el programa toma los parámetros del problema y procede a resolverlo con el algoritmo especificado.

Este programa también incluye un modulo generador de experimentos, cada experimento se define como un archivo de entrada para el programa, donde se indica el problema a correr con sus respectivos parámetros como se indica más adelante en la especificación de cada uno. Se debe ejecutar de la forma:

```
python3 generator.py PROBLEMA ARCHIVO PARÁMETROS
```

Todos los argumentos son requeridos.

- *`PROBLEMA`* valor de 1 o 2 indica el problema de cual generar datos, 1 contenedor, 2 alineamiento.

- *ARCHIVO* es el nombre del archivo de salida (formato se especifica más adelante).
- *PARÁMETROS* se detallan en cada problema más adelante. Estos varían con respecto a cada problema.

Parte 2: Esta parte consiste de un reporte escrito en Latex, formato IEEEtransactions. debe incluir la especificación de cada problema, el diseño de los experimentos, detallando los parámetros y cualquier dato relevante. Debe también resultados de la ejecución de las dos versiones de los algoritmos, así como tablas o gráficos que muestren la comparación de los tiempos de corrida. Por último debe contener análisis y conclusiones. El reporte debe incluir también una explicación de la complejidad temporal de cada algoritmo (fuerza bruta y PD) para cada uno de los problemas, realizando un análisis de acuerdo a los resultados obtenidos.

Problema 1: Problema del contenedor (mochila)

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, y una cantidad dada por c_i. El problema consiste en agregar elementos al contenedor de forma que se maximice el beneficio de los elementos que contiene sin superar el peso máximo que soporta. La solución debe ser dada con la lista de los elementos que se ingresaron y el valor del beneficio máximo obtenido.

Ejemplo: Con un contenedor de peso W = 50. 3 artículos con pesos: 5, 15, 10, beneficios: 20, 50, 60, y unidades 4, 3 y 3 respectivamente. Tiene como solución un beneficio máximo de 260 agregando los artículos 3 (3 unidades) y 1 (4 unidades).

Archivo de entrada: *mochila1.txt*
linea 1: Peso máximo soportado por el contenedor
linea 2: elemento i (peso, beneficio, cantidad)
linea n+1: elemento n (peso, beneficio, cantidad)

```
Input:
50
5,20,4
15,50,3
10,60,3
```

```
Output:
260
1,4 #articulo 1 4 unidades
3,3 #articulo 3 3 unidades
```

Generador

```
python3 generator.py W N minPeso maxPeso minBeneficio maxBeneficio
minCantidad maxCantidad
```

Donde:

`W` es el peso soportado por el contenedor.
`N` la cantidad de elementos.
`minPeso,maxPeso` indica el valor mínimo y máximo para asignar el peso aleatorio a un elemento.

`minBeneficio,maxBeneficio` indica el valor mínimo y máximo para asignar el beneficio aleatorio a un elemento.
`minCantidad,maxCantidad` indica el valor mínimo y máximo para asignar la cantidad disponible de un elemento.

Problema 2: Problema del alineamiento de secuencias.

Se debe implementar el problema de alineamiento global de secuencias visto en clases. (Saul Needleman y Christian Wunsch). Donde se reciba un método para el scoring y las dos hileras a alinear. Siempre se usará el método de scoring +1, -1, -2.

El archivo de entrada es de la siguiente forma:
linea 1: Hilera 1
linea 2: Hilera 2

```
Input:
1,-1,-2
ATTGTGATCC
TTGCATCGGC
```

```
Output:
Tabla resultados [ ]

Score final: -2
Hilera1: ATTGTGATC__C
Hilera2: _TTG_CATCGGC
```

Generador: solo utilizar las letras A T C G

```
python3 generator.py largoH1 largoH2
```

Donde:
`largoH1` es el largo de la hilera 1
`largoH2` es el largo de la hilera 2

Evaluación

Los resultados principales del proyecto se detallaran en el reporte, para el código se evaluará la completitud de la solución, la correcta ejecución de las especificaciones, la escritura del código propio y la documentación de código interna.

Parte 1	70 puntos
Parte 2	30 puntos
Total	100 puntos