ESTRUCTURA DE DATOS 2
Código ST0247

Taller en Sala 11
Programación Dinámica para Exploración del Universo

Objetivos: **1.** Usar programación dinámica para resolver un problema apropiado. **2.** Traducir problemas del mundo real a soluciones algorítmicas.

Consideraciones: Lean y verifiquen las consideraciones de entrega,

Trabajo en Parejas

Mañana, plazo de entrega

Docente entrega plantilla de código en GitHub

Sí .cpp, .py o .java

No .zip, .txt, html o .doc

Alumnos entregan código sin comprimir GitHub

En la carpeta Github del curso, hay un código iniciado y un código de pruebas (tests) que pueden explorar para solucionar los ejercicios

Estructura del documento: **a)** Datos de *vida real*, **b)** I*ntroducción* a un problema, **c)** Problema a resolver, **d)** Ayudas. Identifiquen esos elementos así:

a) **b)** **c)** **d)**

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

En la vida real, se necesita resolver el problema del agente viajero para la construcción de circuitos electrónicos, entrega de domicilios, ruteo de aviones y recolección de las monedas de los teléfonos públicos. Tomado de https://bit.ly/2lS176b

Ejercicios a resolver

1. Desde el comienzo de los tiempos, la humanidad ha deseado explorar el universo. Según la NASA, una de las tecnologías clave para lograrlo es la interferometría.

A diferencia de los telescopios tradicionales, que tienen un solo espejo, la interferometría óptica utiliza varios muestreos de ondas, en varios lugares al mismo tiempo, para encontrar un patrón de interferencia.

Para poder lograr eso, es necesario ubicar varias naves espaciales, en puntos diferentes, para realizar una observación de un objeto estelar. Naturalmente, surge la pregunta de encontrar las rutas que debe tomar cada nave espacial para minimizar el consumo de combustible. Este problema es equivalente al problema del agente viajero.

Dado un grafo completo, hallen el costo mínimo del recorrido que pasa por todos los vértices exactamente una vez y vuelve al nodo inicial utilizando *programación dinámica (el algoritmo de Held-Karp*).

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Ayudas para resolver los Ejercicios

Ejercicio 1

Pista 1: Consideren el siguiente código:

```
public static int heldKarp(Digraph g) {
// complete...
}
```

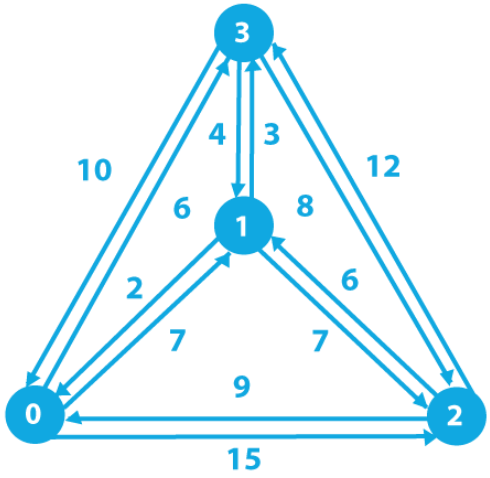
Pista 2: Es el mismo problema que hemos trabajado antes (tanto con B*actracking* como con la técnica *Greedy* del vecino más cercano), solo que esta vez lo resolveremos de manera exacta y mediante la forma más eficiente conocida.

Pista 3: Para que la implementación sea lo más eficiente posible deben representar los conjuntos utilizando máscaras de bits.

En caso de que tengan problemas con las máscaras de bits puede utilizar la clase *BitmaskSet* que hemos creado para ustedes, la cual pretende abstraer las máscaras de bits en forma de *Set* de Java (aunque con muchas limitaciones, pero ideal para la implementación de este algoritmo). Si deciden usarla, recuerden leer la documentación.

Ejemplo 1: En el siguiente grafo el costo total del recorrido de costo mínimo que pasa por los cuatro vértices (iniciando y volviendo a 0) de 22:

:



PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

Iniciando en 0, esta es la tabla de ejecución del algoritmo:

	Costo	Padre
[1, {}]	7	0
[2, {}]	15	0
[3, {}]	6	0
[2, {1}]	14	1
[3, {1}]	10	1
[1, {2}]	21	2
[3, {2}]	27	2
[1, {3}]	10	3
[2, {3}]	14	3
[3, {1, 2}]	Min (3 + [1, {2}], 12 + [2, {1}]) = Min (24, 26) = 24	1
[1, {2, 3}]	Min (6 + [2, {3}], 4 + [3, {2}]) = Min (20, 31) = 20	2
[2, {1, 3}]	Min (7 + [1, {3}], 8 + [3, {1}]) = Min (17, 18) = 17	1
[0, {1, 2, 3}]	Min (2 + [1, {2, 3}], 9 + [2, {1, 3}], 10 + [3, {1, 2}]) = 22	1

Reconstruyendo tenemos 0 → 3 → 2 → 1 → 0.

Pista 4: Si tienen problemas para la implementación del algoritmo, les recomendamos el siguiente video https://bit.ly/2kqN9dz . Recuerden activar los subtítulos (en inglés) en caso de que los necesiten.

PhD. Mauricio Toro Bermúdez
Docente | Escuela de Ingeniería | Informática y Sistemas
Correo: mtorobe@eafit.edu.co | Oficina: Bloque 19 – 627
Tel: (+57) (4) 261 95 00 Ext. 9473

¿Alguna inquietud?

CONTACTO

Docente Mauricio Toro Bermúdez
Teléfono: (+57) (4) 261 95 00 Ext. 9473
Correo: mtorobe@eafit.edu.co
Oficina: 19- 627

Agenden una cita dando clic en la pestaña -*Semana*- de http://bit.ly/2gzVg10