PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
Departamento de Ingeniería Industrial y de Sistemas
Departamento de Ingeniería de Transporte y Logística

Curso: ICS3105/ICT3473 - Optimización Dinámica
Semestre: 02-2018
Profesor: M. Klapp

Tarea 1

Fecha de entrega: Lunes 10 de Septiembre

Por favor conteste la tarea sólo o en pareja y suba documento digital a Siding (Word o Latex) por cuestionario web. Una respuesta x grupo. Penalidad por atraso: 1 punto el primer día, 2 puntos el segundo día, más de 3 días de atraso es un 1.

Pregunta 1

Considere el Problema de Mochila (o Knapsack) que consiste en seleccionar un subconjunto $S \subset [n]$ sobre n objetos, cada uno con volumen $v_i \in \mathbb{Z}_+$ y valor $w_i \in \mathbb{Z}_+$, de manera que quepan en una mochila de volumen B y que maximicen el valor seleccionado.

- ◇ Escoja su lenguaje de programación y solver favorito y resuelva las tres instancias (Fácil, Normal y Difícil) publicadas junto a esta tarea como un problema de Optimización Entera. Recomendación: use Python + Gurobi. Describa como resolvió el problema, las características del PC que utilizó y el tiempo requerido de su algoritmo para computar el óptimo.
- ◇ Ahora resuelva las tres instancias como un problema de Programación Dinámica. Describa como resolvió el problema (pseudocódigo, estructura de datos, etc), las características del PC que utilizó y el tiempo requerido de su algoritmo para computar el óptimo. Se bonificará a algoritmos eficientes que eviten hacer cálculos irrelevantes. Puede recurrir a códigos en internet, pero debe adaptarlos y referenciar al autor original como corresponde (no vale llamar una librería externa). Si además resuelve la instancia desafío recibirá puntaje extra.
- ◇ Suponga ahora que está obligado a escoger los 2 primeros objetos de cada instancia. ¿Cómo puede aprovechar la solución anterior para encontrar el óptimo de este caso?
- ◇ Resuelva el mismo problema de mochila, pero ahora considere que puede repetir cada objeto las veces que usted desee. Modele el problema de forma teórica como un DP definiendo todos su elementos y vuelva a resolverlo computacionalmente indicando la solución para cada instancia publicada.

Pregunta 2

Considere que dispone de n cajas y que cada caja $i \in [n]$ tiene un alto h_i, un ancho a_i y una profundidad ℓ_i. Una caja i puede ser colocada encima de otra caja j si es que $a_i \leq a_j$ y $\ell_i \leq \ell_j$ ó si $a_i \leq \ell_j$ y $\ell_i \leq a_j$ (al girar una caja). Las caja no se pueden rotar (la altura de cada caja no cambia).

- ◇ Plantee el problema DP que determine la torre de cajas más alta que puede formar. Defina etapas, estado, costo inmediato, cost-to-go y decisión en cada estado.
- ◇ Suponga que ahora usted puede rotar las cajas. Es decir, una caja de alto h_i, ancho a_i y profundidad ℓ_i puede ser también una caja de alto a_i, ancho h_i y profundidad ℓ_i o una de alto ℓ_i, ancho a_i y profundidad h_i. Vuelva a plantear el problema. ¿Qué torre debiese ser más alta?

.

Pregunta 3

Considere el principio de recursión discutido en clases. Busque un ejemplo de problema de optimización en donde el principio de recursión no es aplicable. Debe ser estructuralmente diferente al planteado en clases. Muestre, como falla mediante un ejemplo de juguete que no logre la solución óptima mediante recursión.