Programación Dinámica y Greedy

Algoritmos y Estructuras de Datos Avanzadas

Informe Teórico

Nielsen Maximiliano - Uliassi Manuel

Docente: Juan Manuel Rabasedas

6to Informática

Instituto Politecnico Superior Gral. San Martin

Informe Teórico

Objetivo

El objetivo de este Informe Teorico es explicar y complementar con teoria la resolución de los siguientes problemas de Programación Dinámica y Algoritmos Greedy asignados.

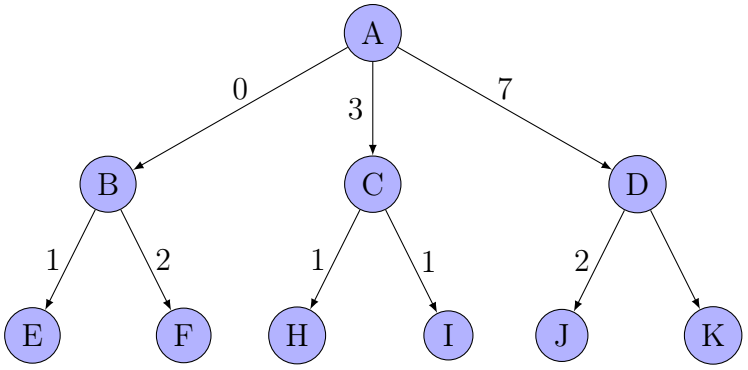
Problema de Programación Dinámica

Dado un arreglo $A[a_1, ..., a_n]$ de enteros no negativos, encontrar una subsecuencia $A[a_1, ..., a_j]$ de tamaño máximo tal que: i = j ó i = j + 1 ó para todo $r \geq 0$, valen las desigualdades $a_{i+r} \geq \sum_{k=i+r+1}^{j-r-1} a_k$ y $a_{j-r} \geq \sum_{k=i+r+1}^{j-r-1} a_k$. Por ejemplo, en el arreglo [1, 8, 2, 1, 3, 9, 10] la subsecuencia [8, 2, 1, 3, 9] cumple la propiedad, ya que $8 \geq 2+1+3$ y $9 \geq 2+1+3(r=0)$ y además $2 \geq 1$ y $3 \geq 1$ (r=1). Escriba un algoritmo de programación dinámica bottom-up para resolver el problema. La salida del algoritmo son dos índices $i \leq j$ tales que $A[a_i, ..., a_j]$ es una subsecuencia de tamaño máximo.

Problema de Algoritmos Greedy

Dado un árbol con pesos no negativos en los lados, se quiere resolver el problema de encontrar el mínimo camino desde la raíz hasta una hoja. Considere el algoritmo greedy que, partiendo desde la raíz como nodo actual, elige siempre el lado l con menor peso. El nuevo nodo actual es el apuntado por l. El algoritmo termina cuando el nodo actual es una hoja.

- Encuentre un ejemplo en que el camino encontrado por el algoritmo no es óptimo.
- Encuentre una condición C sobre árboles de manera tal que, si un árbol cumple con C, entonces el camino encontrado por el algoritmo en ese árbol es optimo. El árbol que se muestra en la figura debe cumplir la condición. Sugerencia: piense en árboles con pesos muy grandes cerca de la raíz y pesos muy chico cerca de las hojas.

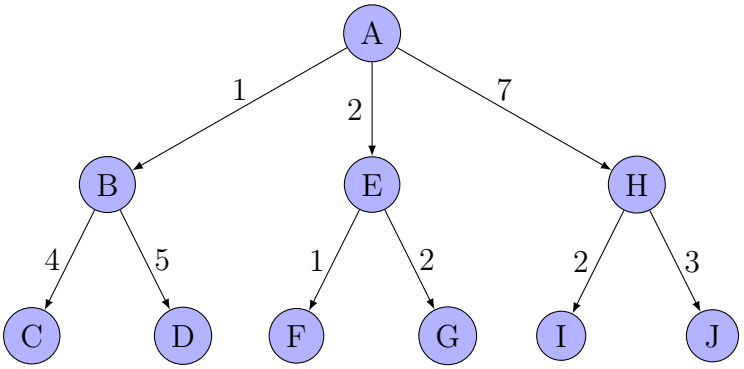


- Pruebe que la condición encontrada en el punto anterior asegura que el algoritmo encuentra el óptimo.

Problema Greedy

No óptimo

El algoritmo planteado no resulta óptimo para cualquier tipo de árbol. Esto se puede ver en el siguiente ejemplo:



El algoritmo para este árbol devuelve el camino *{A,B,C}* que tiene un peso de 5 mientras que la resolución del problema da como resultado el camino *{A,E,F}*, que tiene peso de 3.
El problema más frecuente de este algoritmo es que al no tener en cuenta el árbol en su totalidad opta por una arista liviana con hijas pesadas cuando la opción más óptima es optar por una arista pesada con hijas livianas.

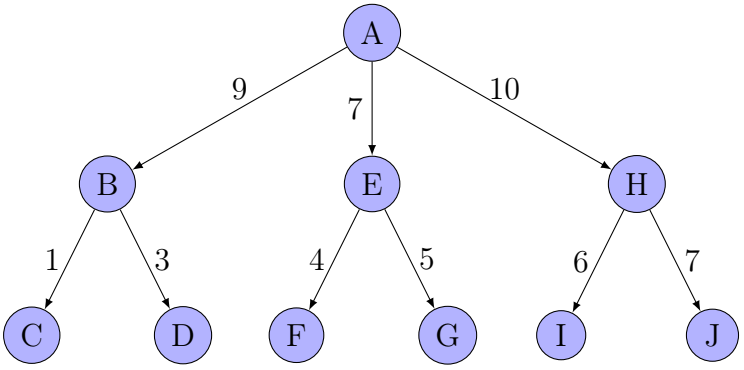
Condiciones específicas

El algoritmo no funciona para cualquier tipo de árbol. Establecimos dos condiciones específicas que debe cumplir un árbol para que el algoritmo de como resultado el mínimo camino de este.
Una rama de un árbol es el camino desde la raíz del mismo hacia cualquiera de sus hojas, el peso de una rama se calcula sumando el valor de cada arista por la que pasa la rama para poder llegar a la hoja correspondiente.

Condición 1 (C1)
La primera condición que debe complir el arbol es que las aristas hermanas deben estar ordenadas de forma creciente.

Condición 2 (C2)
La segunda condición que debe cumplir el árbol es que las ramas de este tienen que estar ordenadas de manera creciente según su peso.

La condición ***C2*** implica que el camino de la raíz a una hoja más liviano siempre será el que se encuentre más a la izquierda. Sin embargo esta condición por si sola no garantiza que tomando siempre la arista de menor peso encontraremos la rama de menor peso.
Veamos el siguiente ejemplo:



Si bien el árbol se encuentra ordenado por ramas, tomar las aristas de menor peso no nos lleva a la rama de menor peso. Tomar las aristas de menor peso resultaría en el camino ***{A,E,F}*** con un costo igual a 11. Mientras que el camino más liviano de la raíz a una hoja se encuentra a la izquierda, ***{A,B,C}*** con un costo de 10.
Esto muestra que además se debe cumplir la condición ***C1***. Esta condición establece que las aristas hermanas deben estar ordenadas por peso de forma creciente. Esto implica que la arista más a la izquierda siempre será la de menor peso. A su vez, gracias a la condición ***C2***, la rama mÃ¡s izquierda será la de menor peso.

De esta forma se prueba que siempre la rama más izquierda contendrá las aristas de menor peso con respecto a sus hermanas y además llevará al camino más liviano hacia una hoja.
Probando así que la elección de la arista mÃ¡s liviana siempre llevará a la rama de menor peso, es decir al camino de menor peso de la raíz a una hoja, siempre que el árbol cumpla con las condiciones ***C1*** y ***C2***.

Implementación del algoritmo Greedy

```
-- El Int es para la arista que da origen al nodo, en el caso de la raiz es 0
data Tree a = N Int a [Tree a] deriving (Show)

-- Funcion que toma una listas de arboles y devuelve el arbol cuya arista que le da origen es la de men
tmin :: [Tree a] -> Tree a
tmin [x] = x
tmin (x:xs) = let
minim (x@(N xe xa xs)) (y@(N ye ya ys)) | xe > ye  = y
| xe <= ye = x
in
minim x (tmin xs)

-- Funcion que ejecuta el algoritmo Greedy
greedy :: Tree a -> [a]
greedy (N e a []) = [a]
greedy (N e a xs) = a : greedy (tmin xs)
```