

Tarea 3

Análisis y Diseño de Algoritmos

Estudiante:

Juan Diego Valencia Alomia

Código:

8977467

Docentes:

Camilo Rocha & Carlos Ramírez

Punto 1)

Schedule

- (c) Choose the course x that starts last, discard all classes that conflict with x , and recurse.

Esta estrategia es correcta, por eso se procede con la especificación del problema

y su respectiva demostración de correctitud mediante inducción matemática:

Especificación:

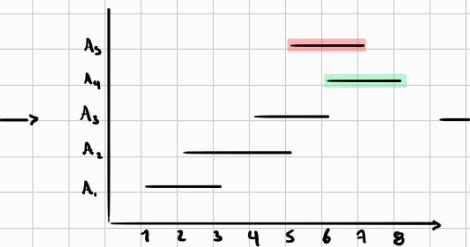
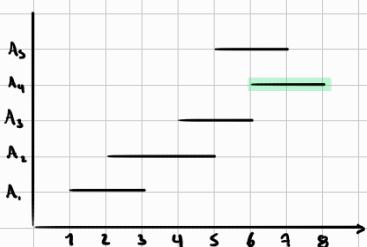
- **Entrada:** un arreglo $A[0 \dots N], N \geq 0$, de tuplas de enteros positivos, tal que $A_i = (t_i, t_f)$, donde t_i y t_f demarcan el tiempo de inicio y finalización de la actividad, $t_i < t_f$
- **Salida:** Cantidad máxima de actividades sin conflictos entre ellas
- **Estrategia Voraz:** ordenar los intervalos por el tiempo de inicio de forma descendiente, para así seleccionar la actividad que comienza de última ignorar las actividades que entran en conflicto y repetir el proceso con el siguiente intervalo hasta cubrir el tamaño de A

Ejemplo

$$\text{Sea } A = [(1,3), (2,5), (4,6), (6,8), (5,7)]$$

ordenamos A por t_i de mayor a menor:

$$A = [(6,8), (5,7), (4,6), (2,5), (1,3)]$$



Cantidad máxima de actividades sin conflicto = 3

Correctitud

Teorema de optimización local

Sea A un conjunto de actividades y a_i la actividad con mayor tiempo de inicio en A , entonces a_i hace parte de un conjunto maximal de actividades sin conflicto

Demonstración:

Sea $B \subseteq A$ un conjunto maximal de actividades sin conflicto de A y

b_i la actividad con mayor tiempo de inicio en B , se procede por casos

- $a_i = b_i$

En este caso $a_i \in B$ y B es un conjunto maximal sin conflictos en A

- $a_i \neq b_i$

En este caso hay que demostrar que a_i hace parte de un conjunto maximal.

Observe que si tenemos un conjunto C , tal que $C = B - \{b_i\} \cup \{a_i\}$.

C no tendría conflictos ya que a_i es máximo en A y por consecuencia

en B también, además $|C| = |B|$ por lo que no cambiaría el número

máximo, dado que ahora $a_i \in C$ entonces hace parte de un conjunto maximal sin conflicto de actividades en A

Por lo tanto en cualquier caso a_i hace parte de un conjunto maximal sin conflicto entre ellos

Función Objetivo

$$0 \leq i \leq N, 0 \leq t \leq \infty$$

$\phi(i, t)$ máxima cantidad de actividades en $A[i..N]$ que

antes del tiempo t pueden agendarse sin que entran en conflicto

Reformulación salida

Salida: $\phi(0, \infty)$

Planteamiento recursivo

$$\phi(i, t) = \begin{cases} 0 & \text{if } i = N \\ 1 + \phi(i+1, A[i][0]) & \text{if } i < N \wedge A[i][1] \leq t \\ \phi(i+1, t) & \text{if } i < N \wedge A[i][1] > t \end{cases}$$

Teorema de optimización global

Sea $0 \leq i \leq N$ el llamado $\phi(i, t)$ que produce la máxima cantidad de actividades si conflictos en $A[i..N]$ que terminan antes del tiempo t y pueden ser seleccionadas sin conflicto

Demonstración

Se procede por inducción sobre el tamaño de $A[i..N]$, $0 \leq i \leq N$

Caso Base

$i = N$

En este caso ya no quedan más actividades por procesar por lo que la solución óptima es 0

Caso inductivo:

$i < N$

H.I: Se asume que $\phi(i+1, t')$ calcula la cantidad máxima de actividades que terminan en un tiempo menor o igual a t' sin conflictos para cualquier $t' \leq t$. Se parte por casos

$A[i][1] \leq t$:

Como las actividades están ordenadas por mayor tiempo de inicio entonces

$A[i]$ corresponde a una actividad con mayor tiempo de inicio y por teorema

de optimización local se sabe que $A[i]$ hace parte de un conjunto maximal

Además como $t' = A[i][0]$ y por H.I entonces $\phi(i+1, t')$ produce un resultado

óptimo y como consecuencia $1 + \phi(i+1, A[i][0])$ también lo es para $A[i..N]$

$A[i][1] > t$:

En este caso la actividad $A[i]$ entra en conflicto, por lo que no puede ser escogida.

De este modo si $t^* = +$ entonces por H.I. $\phi(n, t)$ tambien da una respuesta óptima, ya que $A[i]$ no es agregada.

En conclusión, (al ser el caso $\phi(i, t)$ produce la respuesta óptima para $A[i..N]$)

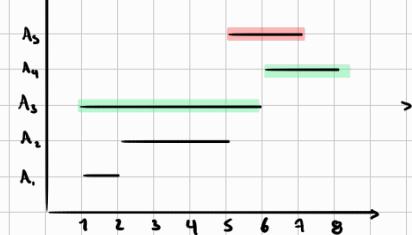
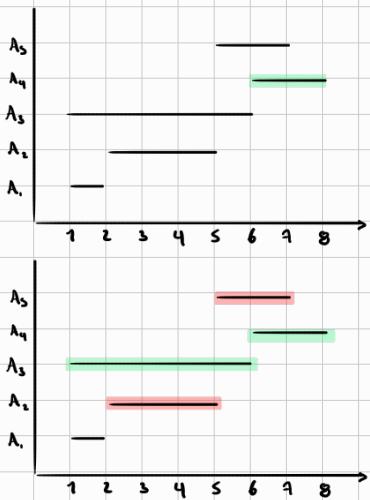
Corolario: $\phi(0, \infty)$

Complejidad $O(N \log N)$ ya que se debe ordenar las actividades

- (a) Choose the course x that ends last, discard classes that conflict with x , and recurse.

La estrategia es incorrecta, se procede por contra ejemplo:

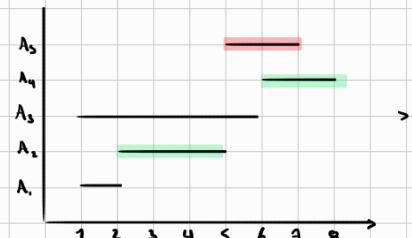
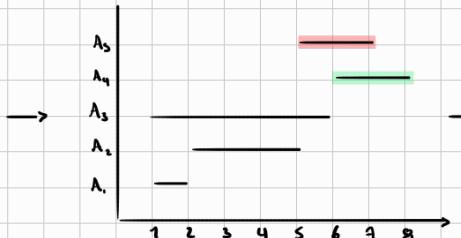
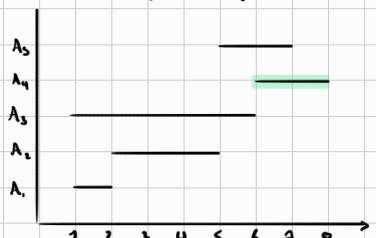
Sea $A = [(1, 2), (2, 4), (1, 6), (5, 7), (6, 8)]$



resultado 2

lo cual es incorrecto

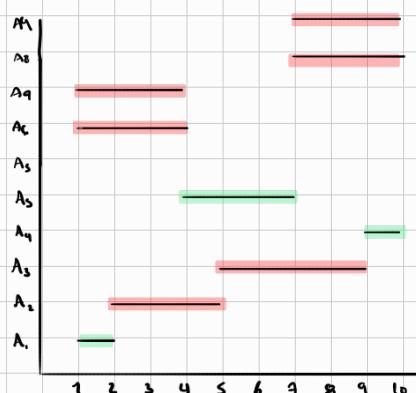
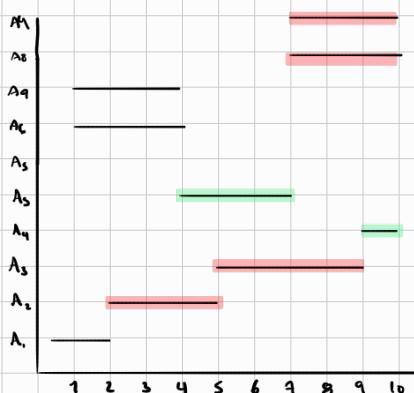
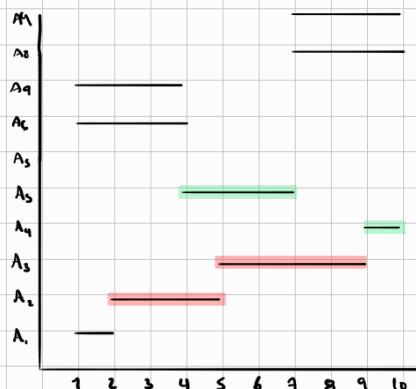
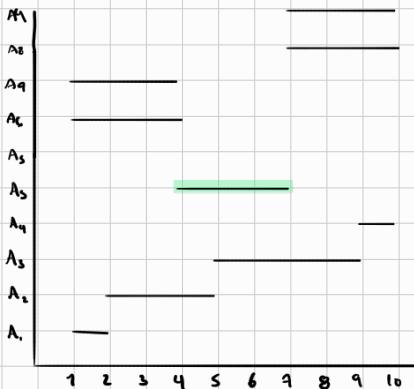
Proceso óptimo:



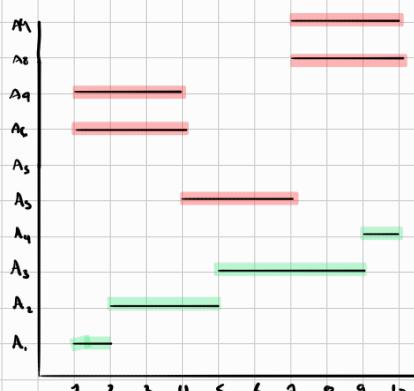
resultado 3 y óptimo

- (e) Choose a course x that conflicts with the fewest other courses, discard all classes that conflict with x , and recurse.

Es incorrecto, se procede por contraejemplo



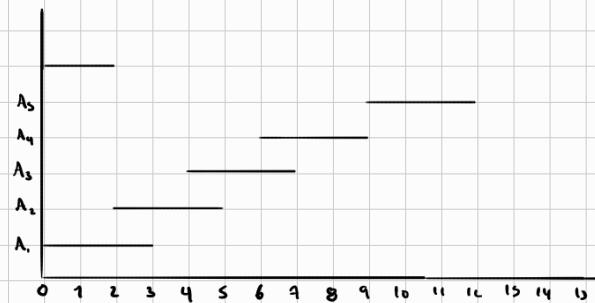
Retorna 3, Pero es incorrecto



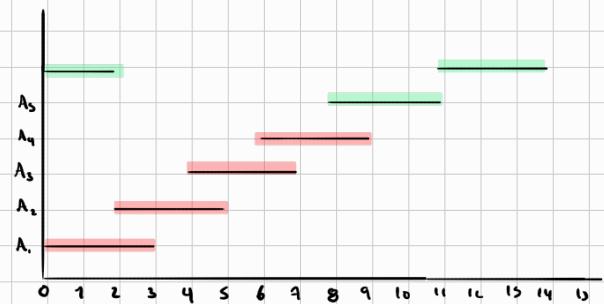
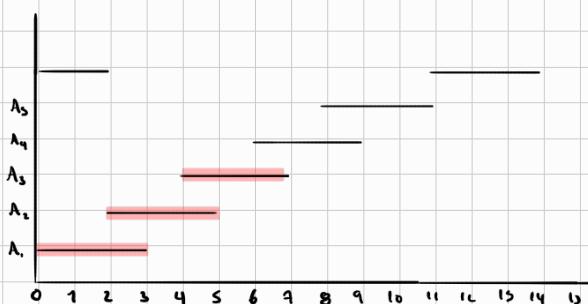
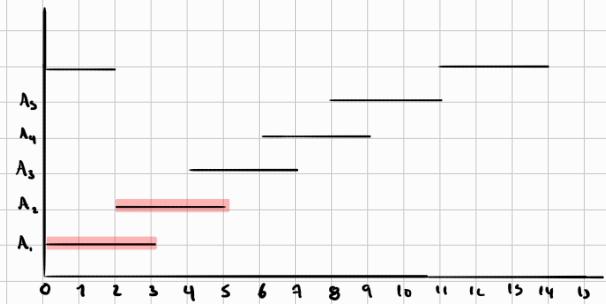
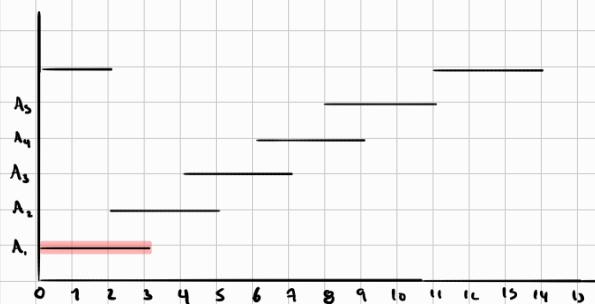
Resultado óptimo = 4

- (g) If no classes conflict, choose them all. Otherwise, discard a course that conflicts with the most other courses and recurse.

Esta estrategia es incorrecta, se procede por contra ejemplo



Como las Actividades A₁, A₂, A₃, A₄ tienen el mismo numero de conflictos entonces selecciona al azar



Resultado = 3, pero es incorrecto

Solucion optima = 4



Punto 2)

Explicación del problema

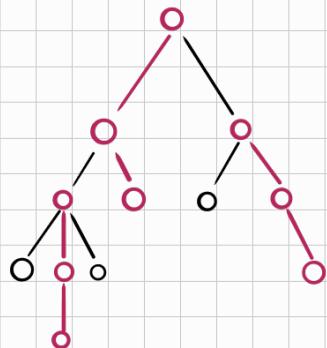
Se tiene un Árbol T y se quiere maximizar el número de caminos de longitud k , $k \geq 0$.

Siendo C un camino que comienza desde un nodo hoja y que C no entre en conflicto con otro camino C'

Estrategia voraz:

ordenar los nodos hoja por su profundidad de manera descendiente y seleccionar los nodos siguiendo ese orden hasta que no sea posible

Ejemplo: Sea T el siguiente árbol y $k=3$



La cardinalidad del conjunto maximal de caminos con $k=3$ es 3

Teorema de optimización local

Sea A un conjunto de nodos hojas y a_i el nodo con mayor profundidad en A , entonces a_i hace parte de un camino que pertenece al conjunto maximal de caminos con longitud k

Demarcación

Sea B un conjunto maximal de caminos de longitud k sin conflicto, P un camino de longitud k , tal que $P \in B$. Definimos P_i como el nodo con mayor profundidad en P y a_i el nodo hijo de P_i .

Como $p_i \neq a_i$ entonces hay que mostrar que a_i pertenece a un camino del conjunto maximal sin conflictos. Considera p_x como el nodo con menor profundidad en P , entonces sea $P' = P - \{p_x\} \cup \{a_i\}$. Observe que P' no tiene conflictos porque la profundidad de a_i es mayor. Además $|P'| = |P| = k$ por lo que seguiría cumpliendo la condición de tamaño k . Dado que $a_i \in P'$ entonces a_i hace parte de un camino que pertenece a un conjunto maximal de caminos sin conflicto.

Entrada

Sea $A[0..N]$ una lista de nodos hoja ordenados descendientemente de acuerdo con su profundidad, $P[0..M]$ la lista de los padres de los nodos y k la longitud requerida de los caminos

Salida

Cantidad máxima de caminos de tamaño k en el árbol

Función objetivo

Sea $0 \leq i \leq N$, $0 \leq n \leq M$, $0 \leq k \leq K$, y V un conjunto vacío

$\phi(i, n, k, V)$ Máxima cantidad de caminos de longitud k sin conflicto en el árbol

Reformulación especificación

Salida: $\phi(1, A[0], 0, \emptyset)$ $\# \emptyset$ es conjunto vacío

Plantamiento Recursivo

$$\phi(i, n, k, V) = \begin{cases} 0 & \text{if } i = N \\ 1 + \phi(i+1, A[i], 0, V \cup \{n\}) & \text{if } i < N \wedge k = K \\ \phi(i, P[n], k+1, V \cup \{n\}) & \text{if } i < N \wedge k < K \wedge n \notin V \\ \phi(i+1, A[i], 0, V \cup \{n\}) & \text{if } i < N \wedge k < K \wedge n \in V \end{cases}$$

Teorema de optimización global

Sea $0 \leq i \leq N$ el llamado $\phi(i, n, k, Vis)$ produce el máximo número de caminos de longitud k empezando por los nodos hoja de $A[0..N]$ de mayor profundidad y sin conflictos

Demonstración

Se procede por inducción sobre el tamaño de $A[i..N]$, $0 \leq i \leq N$

Caso Base:

$i = N$: En este caso ya no quedan más nodos hoja por procesar por lo cual

se pueden producir 0 caminos como máximo lo cual es óptimo

Caso inductivo

$i < N$:

Hipótesis inductiva: Se asume que $\phi(i+1, n', k', Vis')$ produce la cantidad máxima de caminos de longitud K y que se pueden escoger sin conflictos para cualquier

$k' \leq k$

Ahora bien se procede por casos

$k = K$:

Dado que se encontró 1 camino de longitud K , además como los nodos

hoja de $A[0..N]$ están ordenados decendentemente por su profundidad se

tiene que $A[i]$ es el nodo en $A[i..N]$ con mayor profundidad. Luego,

por el teorema de optimización local se sabe que $A[i]$ hace parte

de un posible camino y por eso se puede escoger. Entonces para

este caso $n' = A[i]$, $Vis' = Vis \cup \{n\}$ y $k' = 0$ y por hipótesis

inductiva $\phi(i+1, n', k', Vis')$ produce una solución óptima y en consecuencia

$1 + \phi(i+1, A[i], 0, Vis \cup \{n\})$ es óptimo para $A[i..N]$

$k < k \wedge n \in \text{vis}:$

En este caso el nodo n hace parte de otro camino ya antes explorado, y por ende no es posible continuar con ese camino. Al hacer $n' = A[i]$, $k' = 0$ y $\text{vis}' = \text{vis} \cup \{n\}$ entonces por H.I. $\phi(i+1, n', k', \text{vis}')$ se sabe que produce una solución óptima y por consecuencia $\phi(i+1, A[i], 0, \text{vis} \cup \{n\})$ es óptima para $A[i..N]$

$k < k \wedge n \notin \text{vis}$

En este caso n no hace parte de ningún camino antes explorado por lo que puede formar parte de un camino. Además, al hacer $n' = P[n]$, $k' = n+1$ y $\text{vis}' = \text{vis} \cup \{n\}$ por H.I. $\phi(i+1, n', k', \text{vis}')$ produce una solución óptima y dicha solución también lo es para $\phi(i, n', k', \text{vis}')$ puesto que el nodo n no tiene conflictos y puede formar parte de uno de los caminos. Por lo tanto $\phi(i, n', k', \text{vis}')$ produce la solución óptima

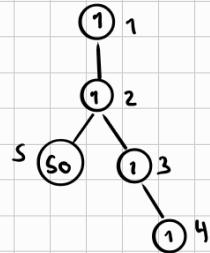
para $A[i..N]$

Colaborario: el llamado $\phi(1, A[0..0], 0, \emptyset)$ produce la cantidad máxima de caminos de longitud k sin conflictos en el árbol

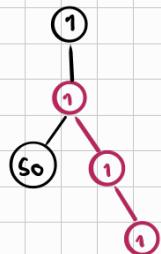
Complejidad: la función $\phi(i, n, k, \text{vis})$ en cuestión es lineal sobre el número de nodos sobre el árbol, no obstante para tener $A[0..N]$ organizado se lo tendría que aplicar un ordenamiento. Por que $T(n) = n \log n + m$, siendo m el número total de nodos y N el número total de nodos hoja. Por lo que si se tiene un árbol con muchos nodos y pocas hojas la complejidad sería $O(m)$ y para un árbol con $m \approx N$ entonces la complejidad sería $O(n \log n)$

2.6)

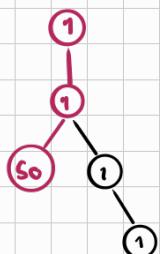
Contraejemplo

Sea T un árbol estructurado así y $k=3$ Entonces $A = [4, 50]$

Por lo que Según el algoritmo propuesto en el anterior punto tomaría la siguiente ruta



Dando una recompensa total de 3, no obstante el resultado óptimo sería

(uya recompensa máxima y óptima sería S_2)

/ /

