

Ejercicio 1: Prueba la correctez del algoritmo ambicioso para el problema de Selección de Actividades

1) function R-A-S (s, f, k, n)

2) m = k + 1

s = vector starting times

3) while m ≤ n and s[m] <= f[k] do

f = vector finishing times

4) m = m + 1

k = Subproblema actual

5) end while

n = número actividades

6) if m ≤ n then

7) return {a_m} ∪ R-A-S (s, f, m, n)

8) else

9) return Ø

10) end if

11) end function

Probar que el algoritmo es correcto y que el algoritmo termina

En la linea 3 del programa tenemos un while en el que una de las condiciones es que m sea menor o igual que n. m es una variable que acumula a la que vamos sumando 1 cada que se entra en el while. De esta manera, sabemos que m en algún punto llegará al valor de n y el ciclo terminará (inclusive puede salir antes por la otra condición). Por otra parte, el programa termina con un return que devuelve el vector

de actividades compatibles o el conjunto vacío en otro caso. Por lo anterior podemos concluir que el programa termina. Proponemos la siguiente invariante: En la iteración m , las actividades $[a_k, \dots, a_{m-1}]$ son incompatibles con a_k . El problema define como overlap cuando una actividad inicia antes que termine la anterior: $s[m] < f[k]$. Utilizaremos esta invariante para probar 2 cosas: 1) validez del ciclo while, 2) validez de la recursión.

Collectez del ciclo while:

Inicialización: Antes que comencemos el ciclo tenemos $k=0$, por lo tanto el valor $m=k+1=1$. Es intuitivo pensar que una actividad a_k es incompatible consigo misma, ya que se tiene un overlap. Por lo tanto la actividad a_{m-1} cumple el invariante.

Mantenimiento: Para el mantenimiento sabemos que la invariante es cierta para la iteración m y probamos que es cierta para $m+1$ como tenemos dos condiciones de salida:

en el caso que $m+1 > n$ salimos del ciclo por la linea 3 del código. Esto implica que ya se han revisado todas las actividades y que estas son incompatibles con la actividad a_k (por la condición de overlap) $s[m+1] > f[k]$ implica que encontramos una actividad compatible con a_k y por lo tanto salimos del while. Con este

condición, las actividades $\{a_k, \dots, a_{m-1}\}$ son incompatibles con a_k

Terminación

Salimos del while cuando $m > n$ o $s[m] > f[k]$. Por la linea 9 sabemos que m va incrementando y cuando llegue al valor de $n+1$ saldremos del while. Al salir del ciclo tendremos que el vector de actividades resultante de la operación $\{a_m\} \cup R.A.S(s, t, m, n)$ es el conjunto vacío. El invariante se cumple en las 3 condiciones \Rightarrow inviánte.

Correctez de la recursión (con inducción)

Para la recursión, suponemos que es correcta para m y probamos para $m+1$. Para este ejercicio usamos el lema 1 visto en clase:

lema 1: El programa $R.A.S$ es un algoritmo que resuelve el problema de encontrar un conjunto máximo de actividades mutuamente compatibles a partir de dos secuencias numéricas de n elementos s, f donde el intervalo $[s_i, f_i]$ representa el tiempo de inicio y finalización de la actividad a_i .

Por la hipótesis de inducción sabemos que cuando utilizamos $R.A.S$ en la iteración $\{m\}$ será correcta. Por lo que el conjunto $\{a_1, \dots, a_m\}$ contiene un subconjunto de actividades compatibles. Para el subconjunto $\{a_{m+1}, \dots, a_n\}$ tenemos dos posibilidades:

1) Que ninguna actividad $\{a_{m+1}, \dots, a_n\}$ sea compatible y por lo tanto, el subconjunto de compatibles a la vez con $\{a_1, \dots, a_m\}$ sea la solución para la lista de tamaño m y también para la lista de tamaño n .

2) Que haya un subconjunto de actividades en $\{a_{m+1}, \dots, a_n\}$ compatible con el subconjunto de actividades compatibles en $\{a_1, \dots, a_m\}$. Por la linea 7 sabemos que cada una de estas actividades se no añadiendo al subconjunto de actividades compatibles con $\{a_1, \dots, a_m\}$. Por la linea 3 sabemos que el ciclo terminará cuando m sea mayor a n . (ya se considera todas las actividades) o cuando se encuentra una actividad compatible (no hay overlap).

El programa terminará al ejecutar la última recursión cuando $m+1 > n$ y regresará el subconjunto de actividades compatibles del conjunto $\{a_1, \dots, a_n\}$ o el conjunto vacío en otro caso. Por las pruebas de corrección del ciclo while y para la recursión, podemos concluir que el programa es correcto.

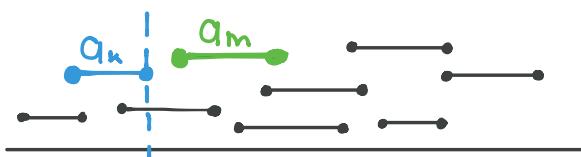
Ejercicio 2: Supongamos que en lugar de escoger la actividad que termina primero, escogemos la actividad que empieza al último que es compatibles con todas las actividades previamente seleccionadas. Prueba que esta propuesta puede ser usada para construir un algoritmo ambicioso que proporcione una solución óptima.

Selección de actividades:

Entrada: Dos secuencias numéricas de n elementos s, f

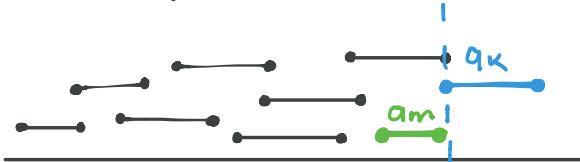
Salida: Un conjunto máximo $A \subseteq \{a_1, \dots, a_n\}$ de actividades mutuamente excluyentes

Intuitivamente, en el primer approach:



Termina primero

En el segundo approach:



empieza al último

El approach a la izquierda es el visto en clase (demostrado en el lema 1)

En la propuesta de la derecha buscamos resolver para S_k un subproblema de Selección de actividades, en este caso queremos demostrar que a_m pertenece a algún subconjunto máximo de actividades mutuamente compatibles de S_k .

- Sea A_K un subconjunto máximo de actividades mutuamente compatibles de S_K
- Sea $a_j \in A_K$ una actividad con el tiempo de inicio más alto.

De la misma manera, dividimos en dos casos:

- Si $a_m = a_j$ entonces terminamos. Por definición A_K es un subconjunto máximo de actividades mutuamente compatibles de S_K y $a_m \in S_K$
- El caso interesante es cuando $a_m \neq a_j$. entonces:

$$A_K' = A_K - \{a_j\} \cup \{a_m\}$$

Queremos demostrar (análogamente) que A_K' es subconjunto máximo de actividades mutuamente compatibles de S_K

$$A_K' = \underline{A_K - \{a_j\}} \cup \{a_m\}$$

$A_K - \{a_j\} \subseteq A_K \Rightarrow$ mutuamente compatible

Es evidente que a_j es compatible con $A_K - \{a_j\}$ por lo que al ser la actividad con tiempo de inicio más alto tenemos que:

$I_j \geq I_l$ para $a_l \in A_K - \{a_j\}$ (a_j tiene el tiempo de inicio más alto en A_K)

Pero... por definición, tenemos que a_m tiene el tiempo de inicio más alto en S_K . Entonces tenemos que $I_m \geq I_j$

Por lo tanto para cualquier actividad $a_l \in A_K - \{a_j\}$ tenemos que: $I_l \leq I_m$. Por lo tanto a_m es compatible con todas

las actividades de $A_K - \{a_{ij}\}$. Entonces A_K' es un subconjunto máximo de actividades mutuamente compatible.

Entonces esta alternativa de selección ambiciosa es segura para resolver el problema ■