12.8. La filmoteca ha organizado un maratón de cine de terror. Durante 24 horas se proyectarán películas (todas diferentes) en las n salas disponibles. Deborah Cinema, gran aficionada a este género de películas, ha conseguido la programación completa donde aparecen todas las películas que se van a proyectar durante el maratón; junto con el título, nombre del director, duración de la película y otros datos de interés, se indica la sala de proyección y la hora de comienzo. Ayudar a Deborah a planificar su maratón de cine, teniendo en cuenta que su único objetivo es ver el máximo número posible de películas.

Solución

Tenemos un conjunto de n películas, cada una con un instante de comienzo c_k y una duración d_k, lo que permite calcular el correspondiente instante de finalización: $f_k = c_k + d_k$. Por tanto, las películas i y j son *compatibles* si los intervalos $[c_i, f_i)$ y $[c_j, f_j)$ no se solapan, es decir si $c_i \geq f_j$ o $c_j \geq f_i$. Tenemos que seleccionar el mayor número de películas que sean mutuamente compatibles. La estrategia voraz consiste en considerar las películas por orden creciente de instante de finalización, y en cada paso i seleccionar la película i-ésima si no solapa con ninguna de las ya elegidas, y rechazarla en caso contrario. Para no perder la relación entre los índices en los vectores de entrada C y D y los tiempos de finalización correspondientes en el vector F, este no se ordena directamente sino utilizando la función ordenar-índices del ejercicio 11.6.

```
fun películas(C[1..n] de nat, D[1..n] de nat⁺) dev sol[1..n] de bool
var F[1..n] de nat⁺     { tiempos finales }
    I[1..n] de 1..n     { para ordenar los índices }
   para i = 1 hasta n hacer
      F[i] := C[i] + D[i] ;  sol[i] := falso
   fpara
   I := ordenar-índices(F)     { F[I[1]] ≤ F[I[2]] ≤ ... ≤ F[I[n]] }
   sol[I[1]] := cierto     { la primera película se elige siempre }
   final := F[I[1]]     { final de la última película elegida }
   para i = 2 hasta n hacer
      si C[I[i]] ≥ final entonces
         sol[I[i]] := cierto ;  final := F[I[i]]
      fsi
   fpara
ffun
```

Como las películas se eligen en orden creciente de tiempo de finalización, *final* es siempre el máximo de los tiempos de finalización de las películas ya elegidas, y, por tanto, para comprobar si la

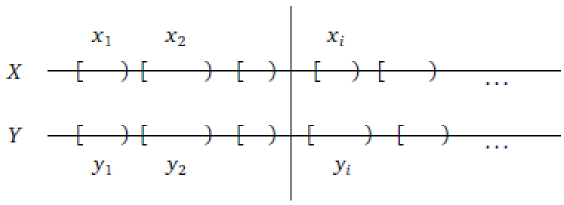

Figura 12.1: Reducción de diferencias para el problema del maratón de cine.

película i es compatible con las ya elegidas es suficiente comprobar si c_i no es menor que *final*. De esta forma, el bucle voraz es de complejidad lineal respecto al número de películas, y el orden de complejidad del algoritmo completo corresponde al de la ordenación del vector de tiempos finales.

Demostramos ahora que la estrategia es óptima, mediante el método de reducción de diferencias, comparando la solución obtenida por el algoritmo, X, con una solución óptima, Y, que existe por ser finito el número de soluciones. Suponemos que ambas están ordenadas por tiempo de finalización creciente. Sea i la primera posición donde ambas soluciones difieren, como muestra la figura 12.1. Puesto que el algoritmo voraz ha elegido x_i y no y_i, sabemos que $f_{x_i} \leq f_{y_i}$. Podemos entonces sustituir la película y_i por la película x_i en la solución Y para obtener otra solución también óptima. Es solución porque la película x_i no solapa con ninguna de las demás películas en Y: no solapa con las anteriores porque no lo hace colocada en X (y las películas anteriores a la i son iguales en ambas soluciones) y no lo hace con las siguientes porque no lo hacía y_i y x_i termina antes. La solución Y sigue siendo óptima porque el número de películas no varía al hacer el intercambio.

Siguiendo con este proceso, podemos hacer que todas las películas seleccionadas por el algoritmo voraz aparezcan en una solución óptima. Además, la solución óptima así obtenida no tiene más películas que X porque, en otro caso, al tener dichas películas un tiempo de finalización mayor, el algoritmo voraz las habría considerado y seleccionado.