Universidad Rey Juan Carlos

Práctica 4: Algoritmos Voraces

Diseño y Análisis de Algoritmos

Grado en Ingeniería Informática Online

- Valor: 10 % de la nota final (1 punto sobre 10)
- Los códigos tendrán que probarse con **Mooshak**
  - `http://gibson.escet.urjc.es/~mooshak`
  - Registrarse en Mooshak:
    - El nombre debe tener el formato "NombreApellido1Apellido2", por ejemplo: `ManuelMunozSanchez` (todo junto, con iniciales en mayúsculas, sin tildes ni eñes)
    - El grupo es el asociado a la titulación y número de expediente del alumno
- Grupos: individual
- No se entregará una memoria
- Fecha límite: 23 de abril de 2013 a las 23:00

Índice

1. Selección de actividades

1.1. Introducción

El objetivo de la práctica es implementar una estrategia óptima para resolver el problema de la selección de actividades mediante un algoritmo voraz. El problema es tratado en profundidad en el apartado 16.1 del libro "Introduction to Algorithms".

1.2. Enunciado del problema

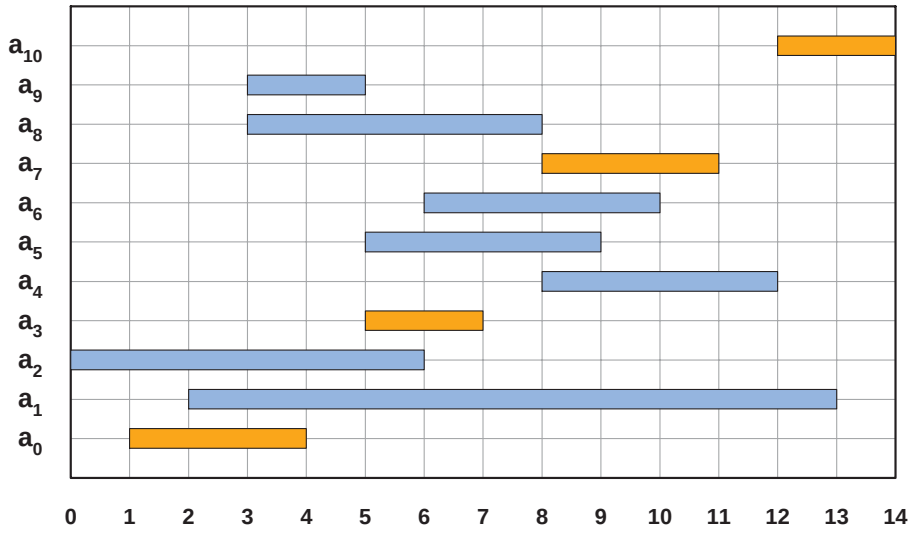
Sea un conjunto A de n actividades $\{a_0, a_1, \ldots, a_{n-1}\}$ que necesitan utilizar un recurso común, por ejemplo, una sala de reuniones. El recurso solo puede ser usado por una actividad en cada momento. Cada actividad tiene un instante de comienzo c_i y un instante de finalización f_i, donde $0 \leqslant c_i < f_i < \infty$.

Si se selecciona la actividad a_i, se desarrolla en el intervalo semiabierto de tiempo $[c_i, f_i)$. Las actividades a_i y a_j son compatibles si sus intervalos $[c_i, f_i)$ y $[c_j, f_j)$ no se solapan, es decir, si $c_i \geqslant f_j$ o $c_j \geqslant f_i$.

El *problema de selección de actividades* consiste en determinar un subconjunto de actividades compatibles cuya cardinalidad sea máxima.

Por ejemplo, sea el siguiente conjunto de actividades:

i	0	1	2	3	4	5	6	7	8	9	10
c_i	1	2	0	5	8	5	6	8	3	3	12
f_i	4	13	6	7	12	9	10	11	8	5	14



Un subconjunto S de actividades compatibles es $\{a_2, a_4, a_{10}\}$. Sin embargo, no es un subconjunto de cardinalidad máxima, como lo son $\{a_0, a_3, a_7, a_{10}\}$ (en naranja en la figura) y $\{a_9, a_3, a_4, a_{10}\}$ (ya que tienen 4 elementos).

1.2.1. Descripción de la entrada

En su primera línea la entrada contiene el número de tareas, que será un número entero positivo $n \in [1, 500]$. En la línea siguiente se especificarán los tiempos de comienzo de las tareas c_i, para $i = 1, \ldots, n$, separados por espacios en blanco. En la tercera línea se indicarán los tiempos de finalización de las tareas f_i, para $i = 1, \ldots, n$, separados por espacios en blanco. Se asume que las entradas cumplen las precondiciones del problema: $0 \leqslant c_i < f_i < \infty$.

1.2.2. Descripción de la salida

La salida contendrá el número máximo de tareas que pueden llevarse a cabo sin solaparse en el tiempo (no se pide indicar qué tareas se realizan), seguido de un salto de línea.

1.2.3. Ejemplo de entrada

```
11↵
1 2 0 5 8 5 6 8 3 3 12↵
4 13 6 7 12 9 10 11 8 5 14↵
```

1.2.4. Salida para el ejemplo de entrada

```
4
```

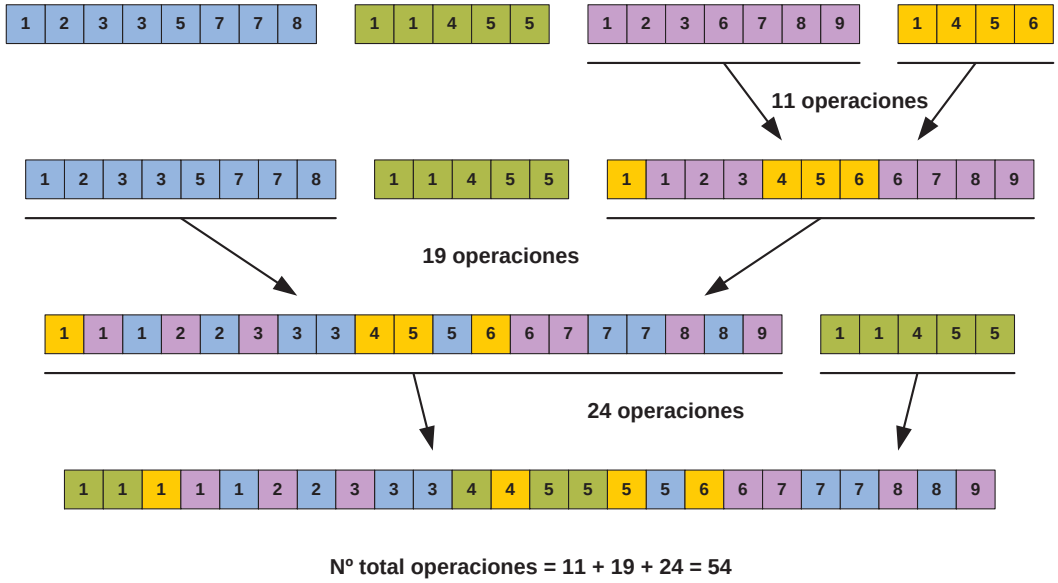
1.3. Tarea a desarrollar

- Implementar la estrategia voraz óptima para resolver el problema.
- El código debe pasar las pruebas de Mooshak

2. Mezcla de listas

2.1. Introducción

Este problema está relacionado con la eficiencia de una generalización del algoritmo de ordenación *mergesort*, en el que la lista original a ordenar se descompone en varias sublistas de tamaños diferentes. El algoritmo emplea un método para combinar las sublistas mezclando parejas de éstas progresivamente hasta ordenar la lista original.



El problema de la mezcla de listas consiste en hallar el número mínimo de operaciones que necesitaría una estrategia (voraz) óptima para ordenar la lista original, suponiendo que realiza una operación por cada elemento a ordenar de una pareja de sublistas. En el ejemplo de la figura partimos de cuatro sublistas de tamaños 8, 5, 7 y 4. Al mezclar por parejas de la forma ilustrada el número total de operaciones realizadas es 54. Sin embargo, esa estrategia no es óptima. Hay una forma más eficiente de mezclar que lleva a cabo solo 48 operaciones.

2.2. Enunciado del problema

Sea una lista de n elementos que se ha dividido en $m \geqslant 1$ sublistas de tamaños $1 \leqslant n_i \leqslant n$, para $i = 1, \ldots, m$, donde $n = \sum_{i=1}^{m} n_i$. Dos sublistas j y k se pueden mezclar para formar una sola lista de tamaño $n_j + n_k$, lo cual requiere realizar justamente $n_j + n_k$ operaciones. Aplicando este procedimiento $m - 1$ veces se puede llegar a obtener una nueva lista de n elementos. Si en el paso l-ésimo de este proceso

se realizan t_l operaciones (para $l = 1, \ldots, m-1$), el número total de operaciones es:

$$T = \sum_{l=1}^{m-1} t_l$$

Se pide calcular el valor mínimo de T posible al emplear una estrategia voraz óptima a la hora de elegir las parejas de sublistas a mezclar en cada uno de los $m-1$ pasos. Si $m = 1$ se considera que el algoritmo no realizará ninguna operación.

2.2.1. Descripción de la entrada

En su primera línea la entrada contiene el número de sublistas m, que será un número entero positivo $m \in [1, 100]$. En la línea siguiente se especificarán los tamaños (enteros) de las sublistas $n_i \in [1, 1000]$, para $i = 1, \ldots, m$, separados por espacios en blanco.

2.2.2. Descripción de la salida

La salida contendrá T, seguido de un salto de línea.

2.2.3. Ejemplo de entrada

```
4↵
8 5 7 4↵
```

2.2.4. Salida para el ejemplo de entrada

```
48
```

2.3. Tarea a desarrollar

- Implementar la estrategia voraz óptima para resolver el problema.
- El código debe pasar las pruebas de Mooshak