.UBAfiuba
FACULTAD DE INGENIERÍA

Interval Partitioning Problem

Teoría de Algoritmos I (75.29 / 95.06)
Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Problema

Sea

E un conjunto de n eventos $\{e_1, e_2, \ldots, e_n\}$

S un conjunto de salas

Cada evento e tiene (un intervalo)

una fecha f_s de inicio

Una fecha f_e de finalización

Queremos

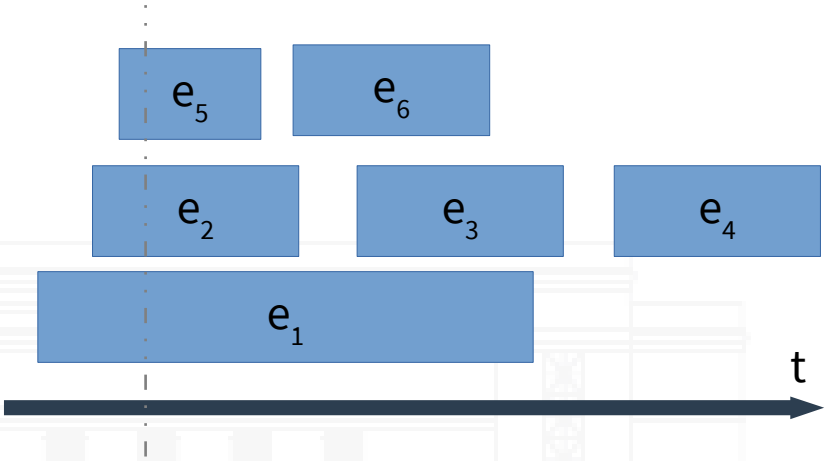
Asignar cada evento a una sala minimizando la cantidad de salas utilizadas.

(dos eventos no pueden compartir tiempo simultaneo en una misma sala)

Análisis del problema

Debemos determinar

Como asignar los eventos a las salas.



Al momento de asignar una evento

Un subconjunto de eventos podrán ser incompatibles para esa misma sala

Por lo tanto al menos se requerirá una sala extra

El número máximo de eventos incompatibles entre si

Determina el número máximo de salas necesarias

Llamaremos profundidad a este parámetro

Una primera solución (problemática)...

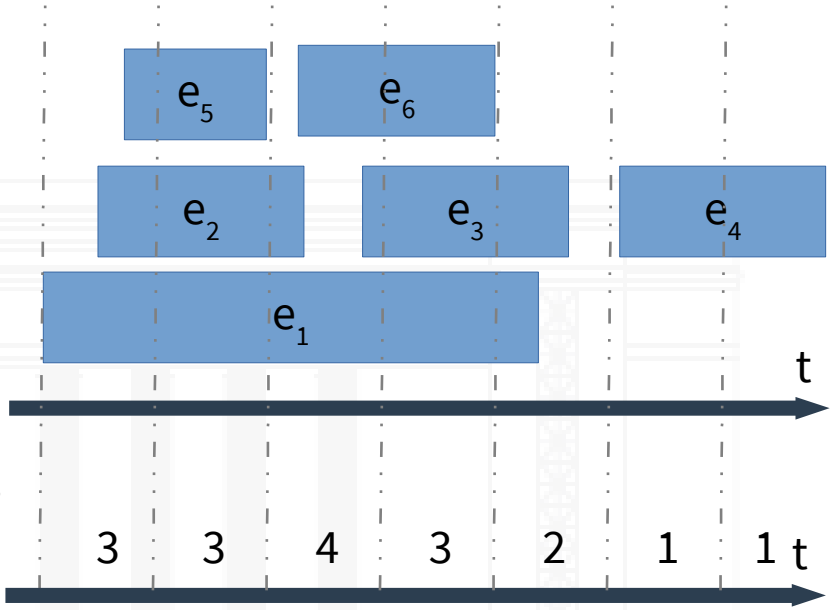
Discretizar en intervalos el uso de las salas

Por cada intervalo llevar un contador de eventos

Luego verificar que intervalo tiene mas eventos y ese será el número de salas

No cualquier separación de intervalos sirve

Una elección incorrecto llevará a resultados erroneos



El número de intervalos i requeridos puede ser muy grande

La complejidad temporal y espacial del algoritmo queda en su función → O(i*e)

No resuelve el problema de asignación

Solo de determinación de la cantidad de salas

Elaborando una solución eficiente...

El número máximo de salas posible

Corresponde al número total de eventos

(en el caso extremo todos los eventos coinciden en algún momento)

Queremos una solución

Que requiera como recursos una cantidad similar a la profundidad del los eventos

Que requiera recorrer la menor cantidad posible a los eventos

Que sea de tipo greedy

Diseño de una solución óptima

Sea d la profundidad del conjunto de eventos

Utilizaremos las etiquetas {1,2,..,d} para asignar eventos a una sala

Ordenaremos los eventos

por su fecha de inició → O(nlogn)

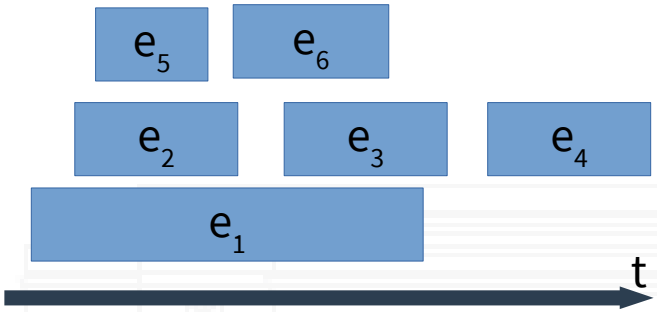
Iteraremos por los eventos ordenados

Etiquetando uno a uno con una etiqueta que no haya sido asignado previamente a un evento que se superpone

Pseudocódigo

```
Sea E set de eventos
Ordenar E los eventos por fecha de inicio

Por cada evento Ej
     Por cada evento Ei que precede a Ej y se superpone con Ej
          Excluir la etiqueta de Ei para Ej

     Si existe una etiqueta {1,2,…,d} que no fue excluida
          asignar la menor a Ej
     Sino
          dejar Ej sin etiquetar

Retornar los eventos etiquetados
```

Ejemplo



Eventos ordenados: e1, e2, e5, e6, e3, e4

Evento	Superponen	excluidas	etiqueta
e1	-	-	1
e2	e1	1	2
e5	e1,e2	1,2	3
e6	e1,e2	1,2	3
e3	e1,e6	1,3	2
e4	-	-	1

```
Sea E set de eventos
Ordenar E los eventos por fecha de inicio

Por cada evento Ej
     Por cada evento Ei que precede a Ej y se
                                   superpone con Ej

          Excluir la etiqueta de Ei para Ej

     Si existe una etiqueta {1,2,…,d} que no
                                   fue excluida

          asignar la menor a Ej

     Sino

          dejar Ej sin etiquetar

Retornar los eventos etiquetados
```

Complejidad

Ordenar

O(nlogn)

Iterar por cada evento

O(n)

Verificar por cada evento anterior

O(n)

Complejidad total

$O(n^2)$

```
Sea E set de eventos
Ordenar E los eventos por fecha de inicio

Por cada evento Ej
     Por cada evento Ei que precede a Ej y se
                                  superpone con Ej

          Excluir la etiqueta de Ei para Ej

     Si existe una etiqueta {1,2,…,d} que no
                                     fue excluida

          asignar la menor a Ej

     Sino

          dejar Ej sin etiquetar

Retornar los eventos etiquetados
```

Optimalidad

¿Puede quedar un evento sin etiquetar?

Sabemos que tenemos una profundidad de d (con d≤n)

Al analizar el evento j. Como mucho d-1 eventos anteriores se superponen

Entonces podemos etiquetar en el peor de los casos el evento con “d”

¿Pueden recibir 2 eventos superpuestos la misma etiqueta?

No, por diseño del algoritmo.

¿Puedo asignar una etiqueta no necesaria?

No, siempre se reutilizan las etiquetas a medida que se van liberando

¿Podemos mejorar la eficiencia?

Utilizando estructura de datos diferentes

Podemos evitar tener la segunda iteración de eventos de exclusión.

Nos interesa conocer la primera etiqueta disponible

Proponemos almacenar por etiqueta utilizada la fecha de liberación

(la fecha de finalización del evento última que tiene asignada)

Al evaluar un nuevo evento (con su respectiva fecha de inicio)

Queremos saber cual es la primera etiqueta (sala) que se libera.

Si la fecha de inicio del evento es anterior a la fecha de liberación de esta sala necesitaremos una sala nueva.

Si no, la asignamos en esa sala y actualizamos la fecha de liberación de esta sala

Utilizaremos

una cola de prioridades (ej: heap) de mínimos.

Pseudocódigo (método mejorado)

```
Ordenar E los eventos por fecha de inicio = {e1, e2, … en }
Q cola de prioridad
u=1 // ultima sala utilizada sala

ev = E[1] //tomo el primer evento
ev.etiqueta = u //etiqueto el evento en la primera sala

Q.push({ev.finalizacion, u})  // sala con fecha de liberacion y etiqueta

Desde j=2 a n
     ev = E[j]
     s = Q.min()

     si ev.inicio >= s.liberacion   // puedo asignar el evento en una sala existente
          s = Q.pop()
          ev.etiqueta = s.etiqueta
          Q.push({ev.finalizacion}, s.etiqueta)
     sino
          u=u+1
          ev.etiqueta = u
          Q.push({ev.finalizacion}, u)

Retornar los eventos etiquetados
```

Complejidad

Ordenar

O(nlogn)

Utilizando un heap

Push,pop → O(log(n))

Min → O(1)

Iteramos 1 vez por evento → O(n)

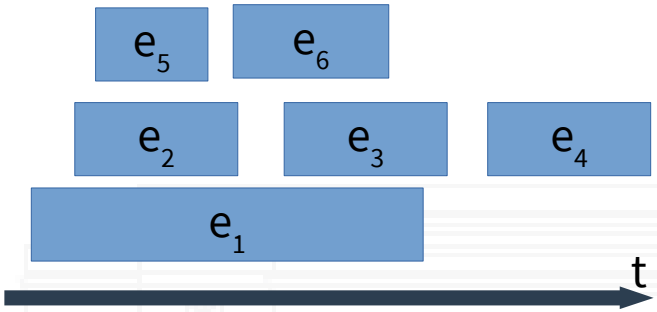
Por cada evento realizamos operaciones en el heap

Complejidad total

O(nlogn)

```
Ordenar E los eventos por fecha de inicio
Q cola de prioridad
u=1 // ultima sala utilizada sala

ev = E[1]
ev.etiqueta = u

Q.push({ev.finalizacion, u})

Desde j=2 a n
     ev = E[j]
     s = Q.min()

     si ev.inicio >= s.liberacion
          s = Q.pop()
          ev.etiqueta = s.etiqueta
          Q.push({ev.finalizacion},
s.etiqueta)
     sino
          u=u+1
          ev.etiqueta = u
          Q.push({ev.finalizacion}, u)

Retornar los eventos etiquetados
```

Ejemplo



Eventos ordenados: e1, e2, e5, e6, e3, e4

Evento	Heap	etiqueta
e1	{e1,1}	1
e2	{e2,2}{e1,1}	2
e5	{e5,3}{e2,2}{e1,1}	3
e6	{e2,2}{e1,1}{e6,3}	3
e3	{e1,1}{e3,2}{e6,3}	2
e4	{e3,2}{e6,3}{e4,1}	1

```
Ordenar E los eventos por fecha de inicio
Q cola de prioridad
u=1 // ultima sala utilizada sala

ev = E[1]
ev.etiqueta = u

Q.push({ev.finalizacion, u})

Desde j=2 a n
     ev = E[j]
     s = Q.min()

     si ev.inicio >= s.liberacion
          s = Q.pop()
          ev.etiqueta = s.etiqueta
          Q.push({ev.finalizacion},
s.etiqueta)
     sino
          u=u+1
          ev.etiqueta = u
          Q.push({ev.finalizacion}, u)

Retornar los eventos etiquetados
```

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Abril de 2021