TAREA SEMANAL 2 – 2

BACKTRACKING

Integrantes:
Andrés Monetta
Alejandro Clara
Sebastián Daloia

Contenido

3 PREGUNTAS

a. ¿Cómo implementa Backtracking la recorrida sobre el espacio de soluciones?

Los algoritmos basados en Backtracking proceden construyendo la tupla solución componente a componente y evaluando si el prefijo de tupla que se tiene en cada momento tiene posibilidades de conducir a una tupla solución, si no existe tal posibilidad se descarta el valor de dicha componente y se analiza con otro valor para la misma. Si no quedan valores para dicha componente se descarta el valor de la penúltima componente y se sigue de esta forma hasta agotar el espacio de soluciones.

b. ¿Qué diferencia a Backtracking de los algoritmos de “fuerza bruta”?

Un algoritmo de fuerza bruta hallaria todas las soluciones posibles a cirto problema, luego se fijaria cual de ellas cumple con las restrinciones dadas. Para esto el algoritmo debe chequear todos los subconjuntos generados, lo que llevaria a un tiempo de ejecución muy elevado.
Por otro lado tenemos a los algoritmos basados en Backtracking, que son mas eficientes en tiempo de ejecución. Estos algoritmos cuentan con ciertas restrinciones y condiciónes que hacen que el espacio de soluciones generado sea mas chico (comparando con todas las soluciones posibles). Como resultado, se buscara la solucion en un conjunto de soluciones mas chico.

c. ¿Qué es la función objetivo y cuándo corresponde tenerla?

La función objeto se especifica en problemas de optimización, y se refiere a optimizar alguna cantidad.
Por ejemplo si se tiene un problema de colorear un mapa con la minima cantidad de colores, es un problema de optimización. Entonces en este caso corresponde tener una función objeto. Esta seria f = min (cantColores(t)), donde cantColores(t) es la cantidad de colores deistintos utilizados en una tupla solución. Por otro lado si se tiene un problema de colocar n reinas en un tablero de ajedrez de nxn, de tal forma que dos reinas cualesquiera no puedan comerse entra si. No es un problema de optimización, por lo que no corresponde tener una función objeto.

d. Si no se controla un predicado de poda, ¿se pierden soluciones?

Dependiendo del problema a resolver, pueden existir datos obtenidos del planteo del mismo que permitan mejorar la recorrida del árbol de soluciones en cuanto a la cantidad de tuplas a generar. Estos datos se denominan predicados de poda. Se debe tener en cuenta que este no determina si una tupla sera solución, es decir el hecho de no controlar este tipo de predicados no genera “falsas”

soluciones, porque dicha tupla cuando este completamente construida no verificará alguna restricción implícita o la función objetivo.

e. ¿Cómo se determina la performance (en general) de los algoritmos de Backtracking?

En general la eficiencia de un algoritmo de Backtracking depende de:

1. Tiempo necesario para generar cada valor de una componente.
2. Cantidad de valores de cada componente que satisfacen las restricciones implícitas.
3. Tiempo necesario para evaluar las restricciones implícitas y los predicados de poda.

4 PROBLEMAS

Caso m = 1, m cantidad de salas

Formalización

Se definen
array salas
array charlas

Las estructuras para el problema son:

Charla
- inicio : unsigned int
- fin : unsigned int
- asistentes : unsigned int /*cantidad esperada de asistentes*/

Sala
- inicio : unsigned int
- fin: unsigned int

Caso m=1 siendo m la cantidad de salas disponibles

□ Forma de solución
k variable de la forma $<x_0, \dots, x_{k-1}>$ con $0 \leq k < n$ y n = |charlas|
x_i representa el índice de un elemento del conjunto charlas para i=0, … , k-1

□ Restricciones explícitas
xi $\in \{0, \dots, n-1\}$ para todo i $0 \leq i \leq k-1$

□ Restricciones implícitas
Dos charlas pueden ser asignadas a la única sala del problema si sus horarios son compatibles.

$$\forall i, j \in charlas \text{ si no } compatibles(i,j) \rightarrow \{i,j\} \notin Misma\ tupla\ solución$$

La suma del horario de las charlas no debe exceder el total de horas disponibles por la única sala.

$$\sum_{i=0}^{k-1} \{charlas[x_i].fin - charlas[x_i].inicio\}$$

$$x_i < x_{i+1} \leq salas[0].fin - salas[0].inicio \ \forall i, 0 \leq i \leq k-2$$

□ Función objetivo

$$T = \{t = < x_0, x_1, \dots, x_{k-1} > . \because t \text{ es solución}\}$$

$f=\max_{t \in T}\{cantAsistentes(t)\}$

$$cantAsistentes(t) = \sum_{i=0}^{k-1} charlas[x_i].asistentes$$

Recorrido: Casos de ejemplo

Caso en donde no todas las charlas pueden ser asignadas a la sala

Juego de datos

Sala 0

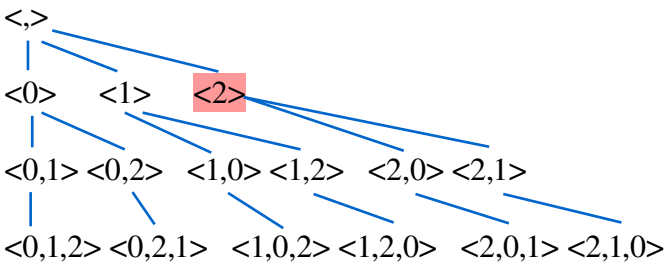
inicio: 16
fin: 19

Charla 0
asistentes: 15
inicio: 16
fin: 17

Charla 1
asistentes: 30
inicio: 17
fin: 18

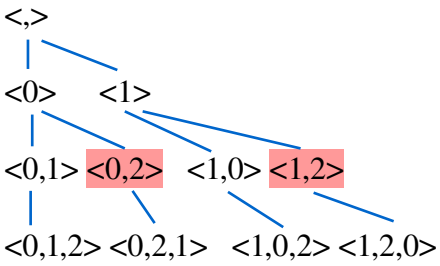
Charla 2
asistentes: 30
inicio: 18
fin: 20

Árbol de soluciones



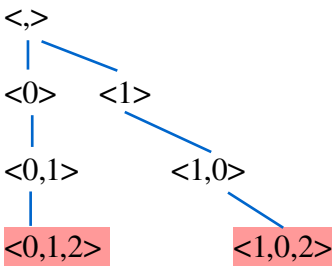
En la raíz se cumple la restricción implícita de la no coincidencia horaria charla-sala, ya que la charla 2 no está contenida por los horarios de la sala.
Luego no se sigue recorriendo el árbol de soluciones por los estados de los subárboles generados por <2>.

Se procede a observar el siguiente árbol



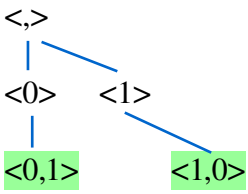
Tanto en <0> como en <1> se cumple la misma restricción implícita mencionada anteriormente. Entonces no se sigue recorriendo por
<0,2> <1,2>

Se procede a observar el siguiente árbol



Tanto en <0,1> como en <1,0> se cumple la misma restricción implícita mencionada anteriormente. Entonces se descartan
<0,1,2> <1,0,2>

Se procede a observar el siguiente árbol



en el cual todas las tuplas son soluciones posibles, y la función objetivo (dependiendo de su implementación) va a devolver <0,1> o <1,0> (son permutaciones una de la otra) que son las soluciones con mayor cantidad de asistentes.

Caso en donde todas las charlas pueden ser asignadas a la sala

Juego de datos

Sala 0

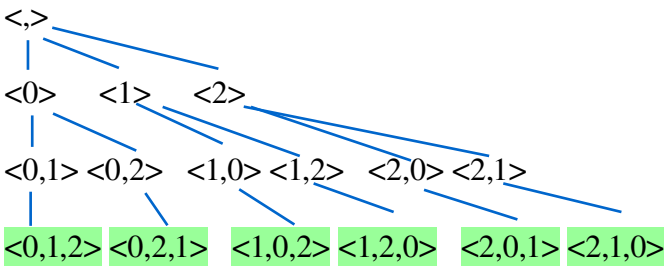
inicio: 16
fin: 19

Charla 0
asistentes: 15
inicio: 16
fin: 17

Charla 1
asistentes: 30
inicio: 17
fin: 18

Charla 2
asistentes: 15
inicio: 18
fin: 19

Árbol de soluciones



Para este caso no existen restricciones de ningún tipo, ya que todas las charlas entran en la sala y no hay coincidencias de horarios entre charlas.

Por lo tanto todas las tuplas son soluciones posibles, y la función objetivo (dependiendo de su implementación) va a devolver cualquier tupla de las que son hoja (<0,1,2>, <0,2,1>, etc), las cuales son permutaciones una de la otra.

Caso general

Formalización

Las estructuras para el problema son:

Charla
inicio : unsigned int
fin : unsigned int
asistentes : unsigned int /*cantidad esperada de asistentes*/

Sala
inicio : unsigned int
fin: unsigned int
charlas : lista de charlas asignadas

□ Forma de solución
$<x_0, \dots, x_{n-1}>$ tupla de largo n donde n es la cantidad de charlas
Cada componente xi representa la sala donde se ubicará la charla valores del 1 al m, o se indica con el valor 0 la no disponibilidad de sala la charla correspondiente al índice del valor.

□ Restricciones explícitas
$0 \leq x_i \leq m$
$0 \leq i \leq n-1$, n cantidad de charlas, m cantidad de salas.

□ Restricciones implícitas

Dos charlas pertenecen a la misma sala si son compatibles (o la negativa de esto)

A una sala se le asigna una charla si el horario de disponibilidad se ajusta a as horas de exposición de la charla.

□ Función objetivo

Devolver aquella solución que asegura la maxima asistencia.

$$T = \{t = < x_{0,} x_{1,} \dots, x_{n-1} \} . \because t \ es \ solución \}$$

$f=\max_{t \in T}\{cantAsistentes(t)\}$

$$cantAsistentes(t)= \sum_{i=0}^{n-1} deltaSala(x_i)\ charlas[i].asistentes$$

$$deltaSala(j) = \begin{matrix} 0 \ si \ j=0, \\ 1 \ si \ j \neq 0, \end{matrix}$$

□ Predicados de poda
Si una charla es compatible con la disponibilidad de al menos una sala, si no es compatible con las

charlas ya asignadas en la lista de charlas de la sala y su cantidad de asistentes es menor a la de las charlas ya asignadas a esas salas. (entonces poda pues queremos asegurar la máxima cantidad de asistentes y esta tiene pocos).

Recorrido: Casos de ejemplo

Casos en donde todas las charlas pueden ser asignadas a alguna sala

Observación: Cabe destacar que en estos casos, la función objetivo va a tener cierto grado ambigüedad (cuando tenga más de una solución posible) debido a que, al poder ser asignadas todas las charlas a alguna sala el máximo número de asistentes siempre va a estar garantizado.

Caso en donde todas las charlas entran en una única sala

Juego de datos

Sala 0

inicio: 16
fin: 19

Sala 1

inicio: 13
fin: 14

Charla 0
asistentes: 20
inicio: 16
fin: 17

Charla 1
asistentes: 10
inicio: 17
fin: 18

Charla 2
asistentes: 30
inicio: 18
fin: 19

Árbol de soluciones

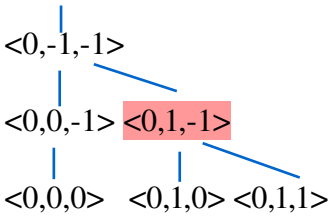
<-1,-1,-1>

<0,-1,-1> <1,-1,-1>

En la raíz se cumple la restricción implícita de la no coincidencia horaria charla-sala, ya que la sala 1 no contiene los horarios de la charla 0.
Luego no se sigue recorriendo el árbol de soluciones por los estados de los subárboles generados por <1,-1,-1>.

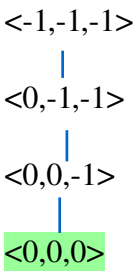
Se procede a observar el siguiente árbol

<-1,-1,-1>



En <0,1,-1> se da una restricción implícita, ya que la sala 1 no contiene los horarios de la charla 2.

Se procede a observar el siguiente árbol



en el cual la solución que garantiza mayor cantidad de asistentes es <0,0,0> y entonces la función objetivo la devuelve.

Caso en donde hay una distribución homogénea de charlas en salas

Juego de datos

Sala 0

inicio: 10
fin: 15

Sala 1

inicio: 13
fin: 15

Sala 2

inicio: 14
fin: 19

Charla 0

asistentes: 20
inicio: 10
fin: 11

Charla 1

asistentes: 30
inicio: 11
fin: 12

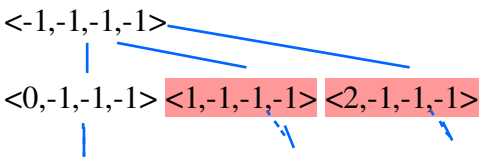
Charla 2

asistentes: 100
inicio: 14
fin: 17

Charla 3

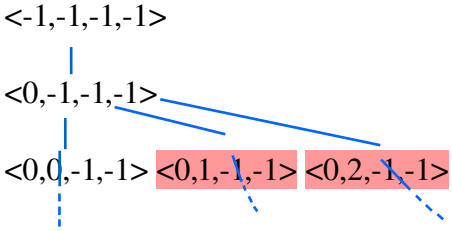
asistentes: 45
inicio: 13
fin: 14

Árbol de soluciones



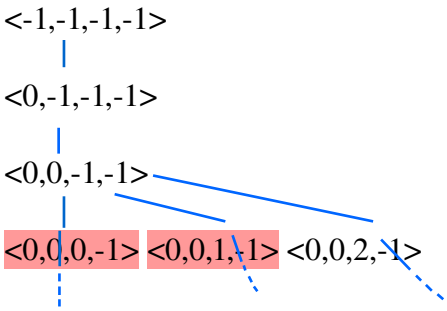
En la raíz se cumple la restricción implícita de la no coincidencia horaria charla-sala, ya que la charla 0 no está contenida por los horarios de la sala 1 y 2.

Se procede con el siguiente árbol



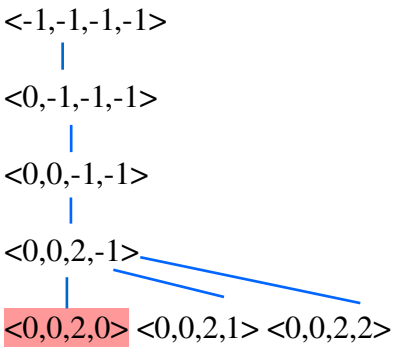
En <0,-1,-1,-1> se cumple la restricción implícita de la no coincidencia horaria charla-sala, ya que la charla 1 no está contenida por los horarios de la sala 1 y 2.

Se procede con el siguiente árbol



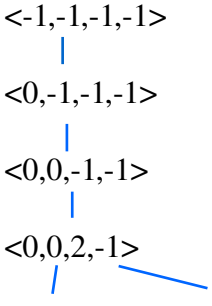
En <0,0,-1,-1> se cumple otra vez la restricción implícita de la no coincidencia horaria charla-sala, ya que la charla 2 no está contenida por los horarios de la sala 0 y 1.

Se procede a mirar el siguiente árbol



En <0,0,2,-1> se cumple nuevamente la restricción implícita de la no coincidencia horaria charla-sala, esta vez debido a que la charla 3 no está contenida por los horarios de la sala 0.

Se procede a mirar el siguiente árbol



<0,0,2,1> <0,0,2,2>

en el cual se tienen las soluciones <0,0,2,1> y <0,0,2,2>, que trivialmente se sabe que son mejores que las de niveles inferiores. Aquí, la función objetivo va a devolver de manera arbitraria entre éstas dos, ya que ambas soluciones garantizan una máxima asistencia.

Caso en donde hay una charla por sala (m=n, caso particular del anterior)

Juego de datos

Sala 0

inicio: 10
fin: 11

Sala 1

inicio: 13
fin: 14

Sala 2

inicio: 15
fin: 17

Charla 0

asistentes: 15
inicio: 10
fin: 11

Charla 1

asistentes: 30
inicio: 13
fin: 14

Charla 2

asistentes: 30
inicio: 15
fin: 16

Árbol de soluciones

<-1,-1,-1>

<0,-1,-1> <1,-1,-1> <2,-1,-1>

En la raíz se da una restricción implícita, ya que la charla 0 no es contenida ni por la sala 1 ni por la sala 2.

Se procede a observar el siguiente árbol

<-1,-1,-1>

<0,-1,-1>

<0,0,-1> <0,1,-1> <0,2,-1>

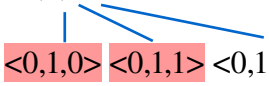
Se da una restricción implícita en <0,-1,-1>, ya que la charla 1 no es contenida ni por la sala 0 ni por la sala 2.

Se procede a observar el siguiente árbol

<-1,-1,-1>

<0,-1,-1>

<0,1,-1>



<0,1,0> <0,1,1> <0,1,2>

en el cual se produce una restricción implícita en <0,1,-1>, debido a que la charla 2 no es contenida ni por la sala 0 ni por la sala 2.

Se procede a observar el siguiente árbol

<-1,-1,-1>

<0,-1,-1>

<0,1,-1>

<0,1,2>

<0,1,2> es la solución que garantiza mayor cantidad de asistentes (única en que se llenan todas las salas) y por lo tanto es devuelta por la función objetivo.

Casos en donde no todas las charlas pueden ser asignadas a alguna sala

Caso en donde ninguna charla puede ser asignada a ninguna sala

Juego de datos

Sala 1

inicio: 17
fin: 18

Sala 2

inicio 15
fin: 19

Sala 3

inicio: 9
fin: 12

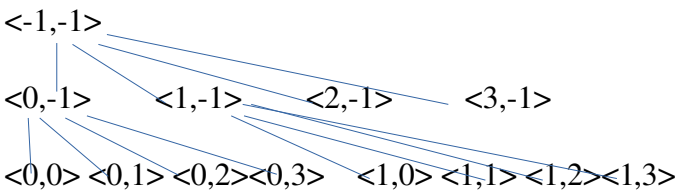
Charla 1

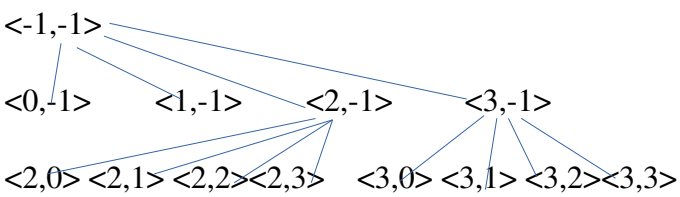
asistentes: 20
inicio: 13
fin: 14

Charla 2

asistentes: 12
inicio: 13
fin: 15

Árbol de soluciones





<0,0> es solución, no es asignada ninguna charla a ninguna sala.
<0,1> <0,2><0,3> No son solución por restricción de no disponibilidad de la sala para esos horarios.
Los árboles generados por las tuplas <1,-1> <2,-1> <3,-1> como raíz no son recorridas, por restricción de no disponibilidad de sala.
La función objetivo tiene como única solución la tupla que indica que no se pueden asignar las charlas a ninguna sala.

Caso en donde una sola charla es asignada a alguna sala

Juego de datos

Sala 1

inicio: 17
fin: 18

Sala 2

inicio 15
fin: 19

Sala 3

inicio: 9
fin: 12

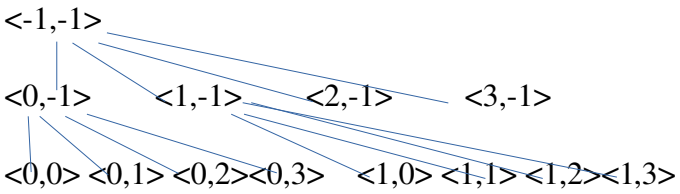
Charla 1

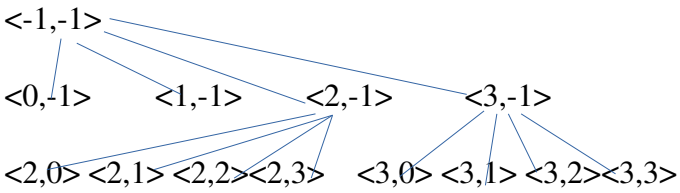
asistentes: 20
inicio: 13
fin: 14

Charla 2

asistentes: 12
inicio: 10
fin: 11

Árbol de soluciones





<0,0> es solución, no es asignada ninguna charla a ninguna sala.
<0,1> <0,2> no son solucíón por restricción de no disponibilidad de cada sala para la charla 2.
<0,3> es solución, la charla 2 es asignada a la sala 3.
Los árboles generados por las tuplas <1,-1> <2,-1> <3,-1> como raíz no son recorridas, por restricción de no disponibilidad de sala.
La función objetivo devuelve la tupla que asegura la máxima cantidad de asistentes: <0,3>.

Caso en donde hay poca diferencia entre la cantidad de charlas asignadas y charlas rechazadas

Juego de datos

Sala 1

inicio: 17
fin: 18

Sala 2

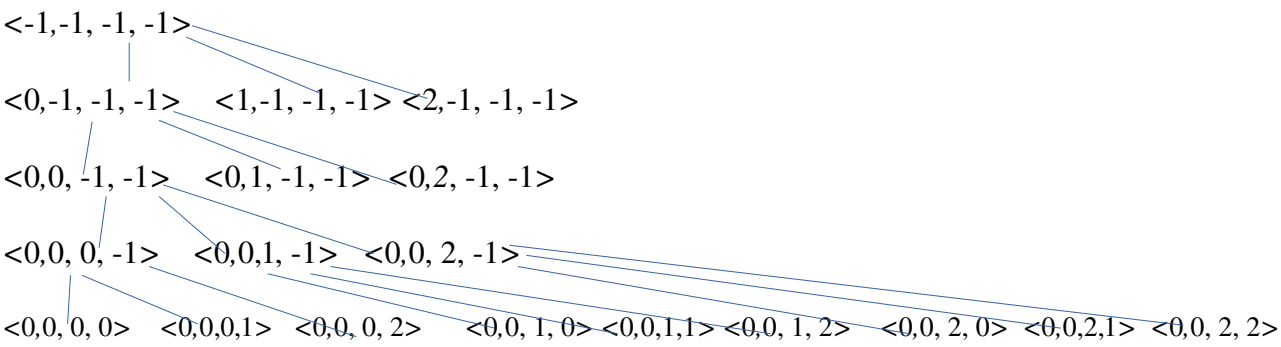
inicio 15
fin: 19

Charla 1
asistentes: 20
inicio: 15
fin: 16

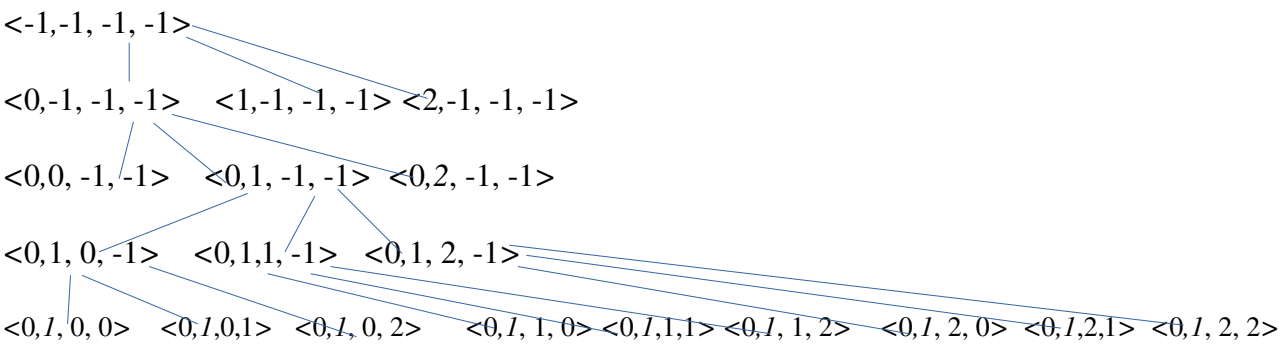
Charla 2
asistentes: 12
inicio: 10
fin: 11

Charla 3
asistentes: 15
inicio: 17
fin: 18

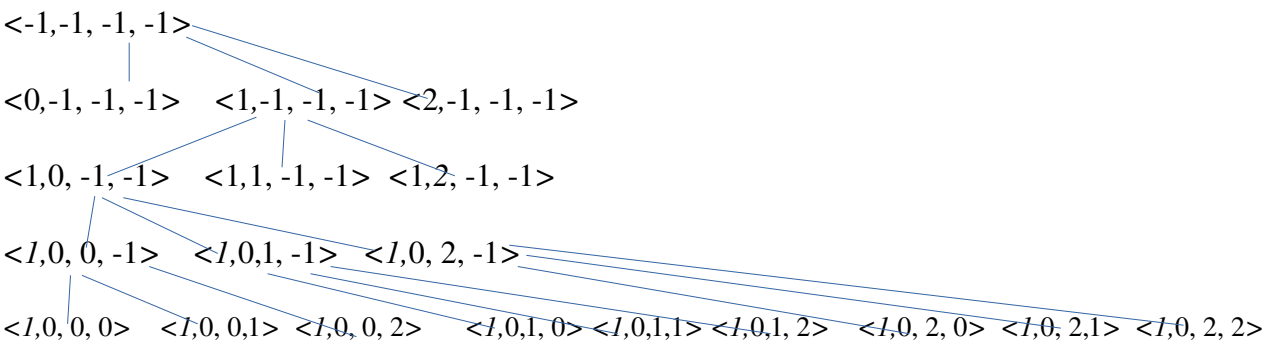
Charla 4
asistentes: 9
inicio: 15
fin: 16



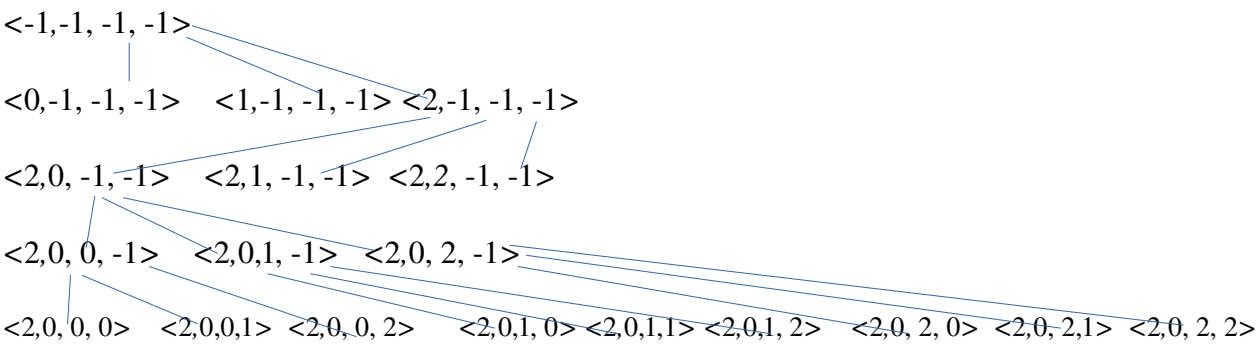
<0,0, 0, 0> es solución ninguna sala es asignada a nignuna charla.
<0,0,0,1> no es solución por restricción de no disponibilidad de la sala 1 para la charla 4
<0,0, 0, 2> es solución, se asigna la sala 2 para la charla 4.
<0,0, 1, 0> es solución, se asigna la sala 1 para la charla 3
<0,0,1,1> no cumple con la restricción de disponibilidad de la sala 1 para la charla 4
<0,0, 1, 2> es solución se asignan ambas charlas a salas distintas.
<0,0, 2, -1> ocurre aquí una poda por no estar disponible la sala 2 para la charla 3, luego no se reccorre el subárbol con esta tupla como raíz.



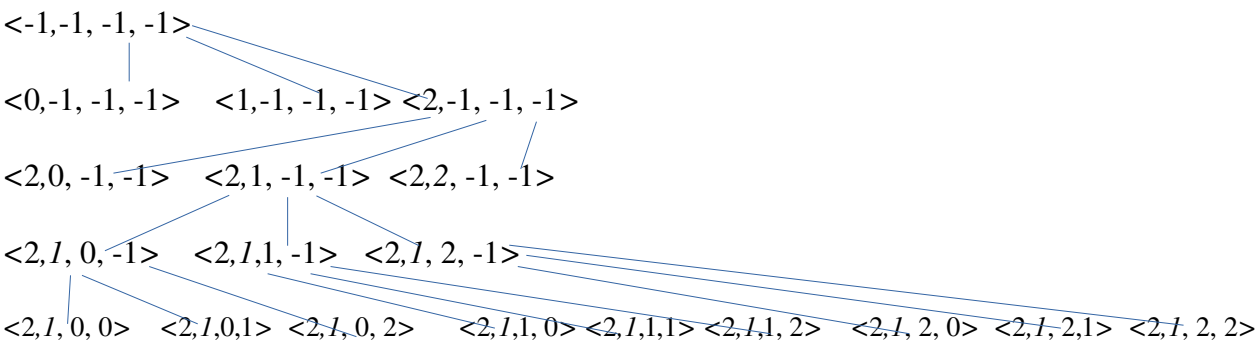
En <0,-1, -1, -1> ocurre poda por predicado cuando se va a evaluar o ubicar la charla 2 a alguna sala (es decir valor distinto de 0 en la segunda componente de la tupla), esto es porque charla 2 no se puede asignar a ninguna sala, entonces no se sigue recorriendo el espacio de soluciones por los hijos que restan recorrer <0,1, -1, -1> <0,2, -1, -1> .



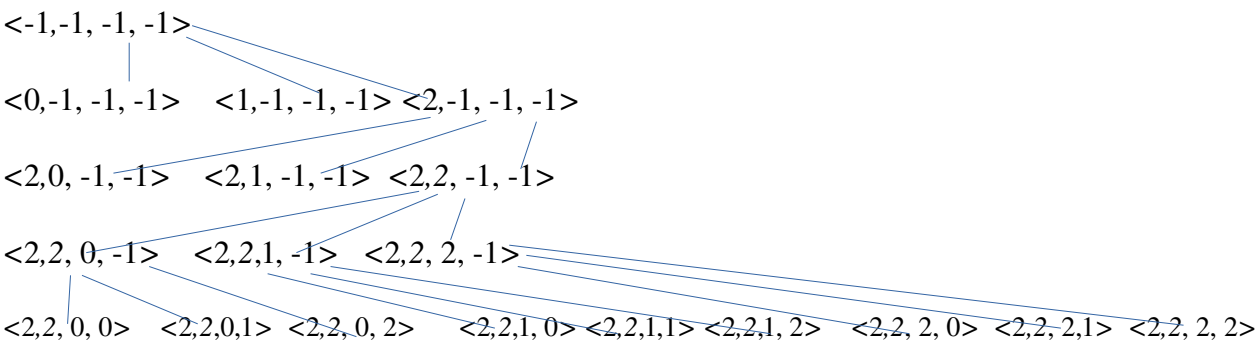
<1,-1, -1, -1> No cumple con la restricción de disponibilidad horaria de sala, por lo que el recorrido por el árbol que tiene esta tupla como solución no se realiza.



<2,0, 0, 0> Es solución la charla 1 es asignada a la sala 2.
<2,0,0,1> No es solución por restricción de no disponibilidad de la sala 1 para el horario de la charla 4.
<2,0, 0, 2> No es solución por restricción de compatibilidad de horarios entre la charla 1 y la 4.
<2,0,1, 0> Es solución asignándose charla 1 a sala 2 y charla 3 a sala 1.
<2,0,1,1> No es solución por restricción de disponibilidad horaria de la sala 1 para la charla 4.
<2,0,1, 2> No es solución por restricción de compatibilidad entre la charla 1 y la 4.
<2,0, 2, 0> Es solución las charlas 1 y 3 se dan en la misma sala.
<2,0, 2,1> No es solución por restricción de no disponibilidad horaria de la sala 1 para la charla 4.
<2,0, 2, 2>No es solución por restricción de compatibilidad entre la charla 1 y la 4.



<2,1, -1, -1> La tupla no cumple con la restricción de disponibilidad, pues la sala 1 no tiene horario disponible para la charla 2. Luego el árbol generado por esta tupla no es recorrido.



<2,2, -1, -1> Esta tupla no cumple con la restricción de disponibilidad, la sala 2 no tiene horarios disponibles para la charla 2.

Se encontraron las siguientes soluciones, para las cuales la cantAsistentes devuelve los valores indicados

Tupla t	**cantAsistentes(t)**
<0,0, 0, 0>	0
<0,0, 0, 2>	9
<0,0, 1, 0>	15
<0,0, 1, 2>	24
<2,0, 0, 0>	20
<2,0,1, 0>	35
<2,0, 2, 0>	35

La función objetivo devuelve el primer valor de tupla igual al máximo cálculado por cantAsistentes, en este caso ***<2,0,1, 0>***