UNIVERSIDAD DE GRANADA

División en Dos Equipos

Algorítmica

Aarón Bueno Rodríguez
Bryan Moreno Picamán
Miguel Ángel Rodríguez Serrano

2º Grado Ingeniería Informática

EXPLICACIÓN DEL PROBLEMA

En este problema se desea dividir un conjunto de n personas para formar dos equipos que competirán entre sí. Cada persona tiene un cierto nivel de competición, que viene representado por una puntuación (un valor numérico entero). Con el objeto de que los dos equipos tengan una capacidad de competición similar, se pretende construir los equipos de forma que la suma de las puntuaciones de sus miembros sea la misma. Diseñar e implementar un algoritmo vuelta atrás para resolver, si es posible, este problema. Mejoradlo usando alguna técnica de poda. Realizar un estudio empírico de la eficiencia de los algoritmos.

División en dos Equipos - Algoritmo vuelta atrás -Backtracking -

Para hacer un manejo eficiente de este problema, se ha optado por separar por un lado la generación de "equipos" y por el otro la solución de la división de los mismos.

En la generación se hace uso de rand para la generación de números (como capacidades de competición) dentro de un rango, asegurándose que la suma de todos ellos sea par (si la suma de todas las capacidades de competición es impar, no se tiene solución de forma segura, por lo que se ha optado por no incluirlos), esto no asegura que exista solución, pero evita muchas ejecuciones inútiles.

Por otra parte se tiene el algoritmo de backtraking en sí mismo, el cual hace uso de la lista de equipos y su capacidad de competición, así como otra lista en la que se almacenaran que jugadores van a qué equipos.

Se ha optado por el uso de una sola lista para almacenar la selección de jugadores, ya que facilita el uso de la misma durante el problema. Ejemplo:

Lista	1	2	1	2	2	1
Equipo A	1	0	1	0	0	1
Equipo B:	0	1	0	1	1	0

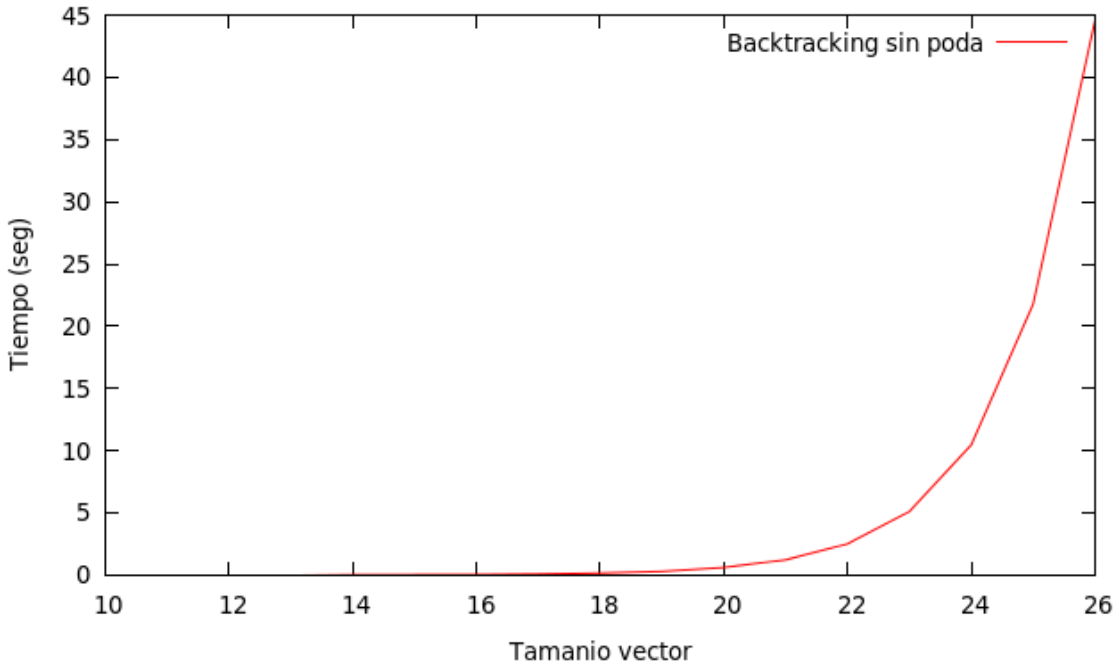
Usando esta forma de representación, en la lista final se tiene un 1 para el equipo 1, un 2 para el equipo dos, de forma que se puede descomponer fácilmente en las dos listas necesarias.

Explicación del algoritmo.

Para la resolución del problema se hace uso de un algoritmo recursivo, cuyo funcionamiento se basa en:

1- El algoritmo recibe una lista con los valores de cada jugador, y otra lista donde almacenar el reparto de los mismos.
2- Se añade primero el jugador 1 al final de la lista de reparto.
3- Se comprueba que esta lista no disponga ya de todos los jugadores posicionados (nos encontraríamos en un nodo hoja), en caso de no serlo se llamaría de nuevo a la función, que repetiría estos pasos hasta llegar a la base del árbol.
4- Si ya se está en un nodo hoja, se examina si este es solución (con la función específica para ello), en caso positivo se devuelve.
5- Ahora se empezaría el Backtracking, se retira de la lista el último elemento insertado, se mete el siguiente (en este caso un numero 2), y se vuelve a examinar si es solución. En caso afirmativo se devuelve.
6- En el paso final se saca el elemento insertado anteriormente de la lista, y se termina la llamada a la función, esto asegura que cuando se termina la llamada recursiva, el algoritmo continua en el paso siguiente, que sería quitar el último elemento, añadir el siguiente y llamar de forma recursiva de nuevo.

A continuación podemos ver una gráfica con el gasto de tiempo y como va aumentando conforme aumenta la entrada, superando al caso siguiente en el que hacemos uso de una poda, este cambio se hace más pronunciado conforme aumenta el tamaño, siendo bastante más eficiente el algoritmo que poda que el que no lo hace, como se podrá ver a continuación.



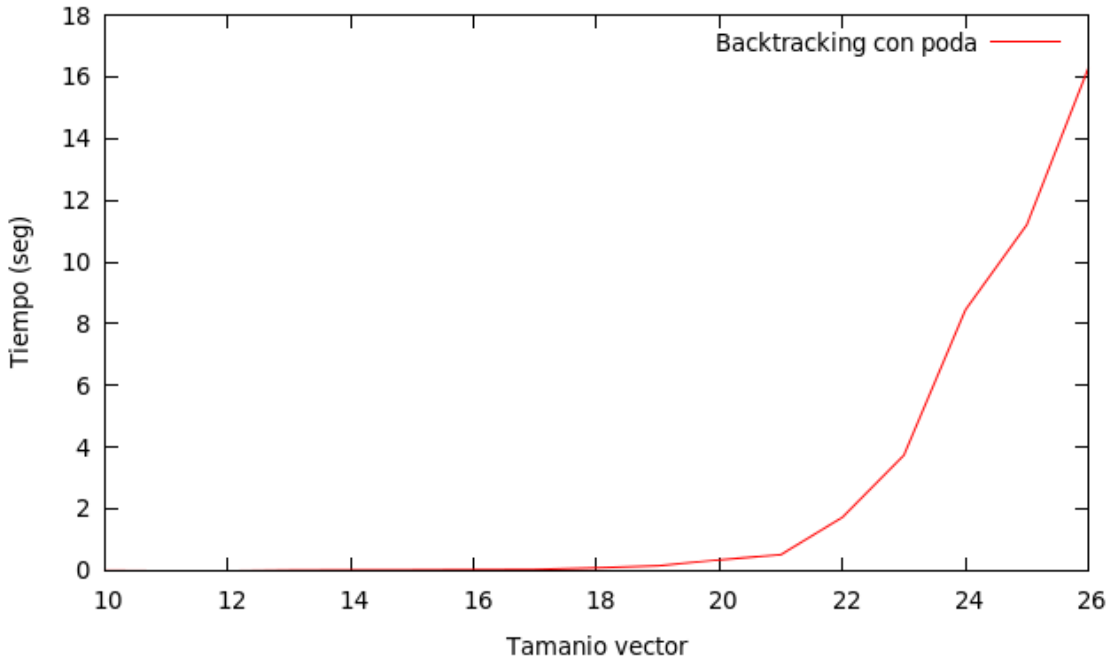
División en dos Equipos - Backtracking con Poda-

Para la realización de la poda, se ha partido del algoritmo base anterior, con las mismas estructuras y funcionamiento. A este se le ha añadido una poda, la cual se realiza de la siguiente manera:

- Se lleva una suma de las capacidades (hasta el momento) de cada uno de los equipos.
- Si alguna de estas sumas sobrepasa la mitad de la suma total de todos los números se realiza la poda, ya que es imposible encontrar una solución por este camino.

Además de esto se realiza un complemento a la poda, en el cual si se tiene solución sumando las capacidades al el equipo 1 (ya que es por el que se pasa primero siempre que se explora en profundidad), se obvia el comprobar si la va a tener en el otro, esta pequeña mejora ahorra algunos ciclos de procesador mejorando el rendimiento.

A continuación podemos ver una gráfica con el gasto de tiempo y como va aumentando conforme aumenta la entrada, no obstante no tanto como en el caso anterior sin poda, que claramente tenia resultados peores como se podrá ver a continuación.



Ejemplo de ejecución Backtracking Con y Sin Poda

Los tiempos salen algo superiores que en las ejecuciones del estudio empírico por la presencia de salidas por pantalla que ralentizan la ejecución.

Valores Originales:	**4|7|8|6|4|6|7|3|10|3|**		
Backtracking Sin Poda		**Backtracking Con Poda**	
A: 4|7|8|6|4|0|0|0|0|0|	**B: 0|0|0|0|0|6|7|3|10|3|**	**A: 4|7|8|6|4|0|0|0|0|0|**	**B: 0|0|0|0|0|6|7|3|10|3|**
A: 4|7|8|0|4|6|0|0|0|0|	**B: 0|0|0|6|0|0|7|3|10|3|**	**A: 4|7|8|0|4|6|0|0|0|0|**	**B: 0|0|0|6|0|0|7|3|10|3|**
A: 4|7|8|0|4|0|0|3|0|3|	**B: 0|0|0|6|0|6|7|0|10|0|**	**A: 4|7|8|0|4|0|0|3|0|3|**	**B: 0|0|0|6|0|6|7|0|10|0|**
A: 4|7|8|0|0|0|7|3|0|0|	**B: 0|0|0|6|4|6|0|0|10|3|**	**A: 4|7|8|0|0|0|7|3|0|0|**	**B: 0|0|0|6|4|6|0|0|10|3|**
A: 4|7|8|0|0|0|7|0|0|3|	**B: 0|0|0|6|4|6|0|3|10|0|**	**A: 4|7|8|0|0|0|7|0|0|3|**	**B: 0|0|0|6|4|6|0|3|10|0|**
A: 4|7|8|0|0|0|0|0|10|0|	**B: 0|0|0|6|4|6|7|3|0|3|**	**A: 4|7|8|0|0|0|0|0|10|0|**	**B: 0|0|0|6|4|6|7|3|0|3|**
A: 4|7|0|6|0|6|0|3|0|3|	**B: 0|0|8|0|4|0|7|0|10|0|**	**A: 4|7|0|6|0|6|0|3|0|3|**	**B: 0|0|8|0|4|0|7|0|10|0|**
A: 4|0|8|6|4|0|7|0|0|0|	**B: 0|7|0|0|0|6|0|3|10|3|**	**A: 4|0|8|6|4|0|7|0|0|0|**	**B: 0|7|0|0|0|6|0|3|10|3|**
A: 4|0|8|0|4|6|7|0|0|0|	**B: 0|7|0|6|0|0|0|3|10|3|**	**A: 4|0|8|0|4|6|7|0|0|0|**	**B: 0|7|0|6|0|0|0|3|10|3|**
A: 4|0|8|0|4|0|7|3|0|3|	**B: 0|7|0|6|0|6|0|0|10|0|**	**A: 4|0|8|0|4|0|7|3|0|3|**	**B: 0|7|0|6|0|6|0|0|10|0|**
A: 4|0|8|0|4|0|0|3|10|0|	**B: 0|7|0|6|0|6|7|0|0|3|**	**A: 4|0|8|0|4|0|0|3|10|0|**	**B: 0|7|0|6|0|6|7|0|0|3|**
A: 4|0|8|0|4|0|0|0|10|3|	**B: 0|7|0|6|0|6|7|3|0|0|**	**A: 4|0|8|0|4|0|0|0|10|3|**	**B: 0|7|0|6|0|6|7|3|0|0|**
A: 4|0|8|0|0|0|7|0|10|0|	**B: 0|7|0|6|4|6|0|3|0|3|**	**A: 4|0|8|0|0|0|7|0|10|0|**	**B: 0|7|0|6|4|6|0|3|0|3|**
A: 4|0|0|6|0|6|7|3|0|3|	**B: 0|7|8|0|4|0|0|0|10|0|**	**A: 4|0|0|6|0|6|7|3|0|3|**	**B: 0|7|8|0|4|0|0|0|10|0|**
A: 4|0|0|6|0|6|0|3|10|0|	**B: 0|7|8|0|4|0|7|0|0|3|**	**A: 4|0|0|6|0|6|0|3|10|0|**	**B: 0|7|8|0|4|0|7|0|0|3|**
A: 4|0|0|6|0|6|0|0|10|3|	**B: 0|7|8|0|4|0|7|3|0|0|**	**A: 4|0|0|6|0|6|0|0|10|3|**	**B: 0|7|8|0|4|0|7|3|0|0|**
A: 0|7|8|0|4|0|7|3|0|0|	**B: 4|0|0|6|0|6|0|0|10|3|**	**A: 0|7|8|0|4|0|7|3|0|0|**	**B: 4|0|0|6|0|6|0|0|10|3|**
A: 0|7|8|0|4|0|7|0|0|3|	**B: 4|0|0|6|0|6|0|3|10|0|**	**A: 0|7|8|0|4|0|7|0|0|3|**	**B: 4|0|0|6|0|6|0|3|10|0|**
A: 0|7|8|0|4|0|0|0|10|0|	**B: 4|0|0|6|0|6|7|3|0|3|**	**A: 0|7|8|0|4|0|0|0|10|0|**	**B: 4|0|0|6|0|6|7|3|0|3|**
A: 0|7|0|6|4|6|0|3|0|3|	**B: 4|0|8|0|0|0|7|0|10|0|**	**A: 0|7|0|6|4|6|0|3|0|3|**	**B: 4|0|8|0|0|0|7|0|10|0|**
A: 0|7|0|6|0|6|7|3|0|0|	**B: 4|0|8|0|4|0|0|0|10|3|**	**A: 0|7|0|6|0|6|7|3|0|0|**	**B: 4|0|8|0|4|0|0|0|10|3|**
A: 0|7|0|6|0|6|7|0|0|3|	**B: 4|0|8|0|4|0|0|3|10|0|**	**A: 0|7|0|6|0|6|7|0|0|3|**	**B: 4|0|8|0|4|0|0|3|10|0|**
A: 0|7|0|6|0|6|0|0|10|0|	**B: 4|0|8|0|4|0|7|3|0|3|**	**A: 0|7|0|6|0|6|0|0|10|0|**	**B: 4|0|8|0|4|0|7|3|0|3|**
A: 0|7|0|6|0|0|0|3|10|3|	**B: 4|0|8|0|4|6|7|0|0|0|**	**A: 0|7|0|6|0|0|0|3|10|3|**	**B: 4|0|8|0|4|6|7|0|0|0|**
A: 0|7|0|0|0|6|0|3|10|3|	**B: 4|0|8|6|4|0|7|0|0|0|**	**A: 0|7|0|0|0|6|0|3|10|3|**	**B: 4|0|8|6|4|0|7|0|0|0|**
A: 0|0|8|0|4|0|7|0|10|0|	**B: 4|7|0|6|0|6|0|3|0|3|**	**A: 0|0|8|0|4|0|7|0|10|0|**	**B: 4|7|0|6|0|6|0|3|0|3|**
A: 0|0|0|6|4|6|7|3|0|3|	**B: 4|7|8|0|0|0|0|0|10|0|**	**A: 0|0|0|6|4|6|7|3|0|3|**	**B: 4|7|8|0|0|0|0|0|10|0|**
A: 0|0|0|6|4|6|0|3|10|0|	**B: 4|7|8|0|0|0|7|0|0|3|**	**A: 0|0|0|6|4|6|0|3|10|0|**	**B: 4|7|8|0|0|0|7|0|0|3|**
A: 0|0|0|6|4|6|0|0|10|3|	**B: 4|7|8|0|0|0|7|3|0|0|**	**A: 0|0|0|6|4|6|0|0|10|3|**	**B: 4|7|8|0|0|0|7|3|0|0|**
A: 0|0|0|6|0|6|7|0|10|0|	**B: 4|7|8|0|4|0|0|3|0|3|**	**A: 0|0|0|6|0|6|7|0|10|0|**	**B: 4|7|8|0|4|0|0|3|0|3|**
A: 0|0|0|6|0|0|7|3|10|3|	**B: 4|7|8|0|4|6|0|0|0|0|**	**A: 0|0|0|6|0|0|7|3|10|3|**	**B: 4|7|8|0|4|6|0|0|0|0|**
A: 0|0|0|0|0|6|7|3|10|3|	**B: 4|7|8|6|4|0|0|0|0|0|**	**A: 0|0|0|0|0|6|7|3|10|3|**	**B: 4|7|8|6|4|0|0|0|0|0|**
TIEMPO:			
0.001807		**0.000815**	

Estudio Empírico de Eficiencia (Comparativa y Ejecución)

Para comprobar realmente si la poda de algunas de las ramas del árbol de decisiones mejora la ejecución del algoritmo, se ha llevado a cabo un estudio empírico de los mismos comprobando la ejecución con distintos tamaños de equipos.

A continuación se puede ver tanto en una tabla como de forma gráfica la comparativa de los tiempos de ejecución y la mejora que supone esta poda.

	10	11	12	13	14	15	16	17	18
Backtracking	0,001350	0,002927	0,005437	0,009487	0,015727	0,019604	0,036157	0,073780	0,148305
Backtracking con Poda	0,000782	0,001772	0,003386	0,004786	0,013921	0,017950	0,024254	0,032134	0,084870

	19	20	21	22	23	24	25	26
Backtracking	0,283401	0,601555	1,213900	2,498110	5,092960	10,4538	21,757200	44,564900
Backtracking con Poda	0,148310	0,344421	0,510250	1,720540	3,731900	8,443360	11,200500	16,273300