UNIVERSIDAD NACIONAL DEL CENTRO
DE LA PROVINCIA DE BUENOS AIRES

Programación 3

FACULTAD DE CIENCIAS EXACTAS

GREEDY & BACKTRACKING

DOCENTES

Federico Casanova - Federico Améndola - Leandro Rocamora - Sebastián Vallejos - Matias San Román

ALUMNA

Magalí Menchón - mamenchon@alumnos.exa.unicen.edu.ar

GREEDY

1.0 Los detalles del diseño Greedy del punto anterior:

1.1 ¿Qué son los candidatos?

Consideré a los candidatos como el conjunto de empleados que se recibe como entrada.

1.2 ¿Cuál es el criterio Greedy para seleccionar el próximo candidato (incluyendo aquellos criterios que se pensaron y descartaron durante el desarrollo del trabajo)

Inicialmente pensé en un seleccionar que elija siempre el primer elemento del conjunto de candidatos, donde previamente se ha ordenado descendentemente. Esto generaba que siempre el primer grupo quedará con una mayor fuerza de trabajo.

Luego consideré en seleccionar a los candidatos de la misma forma, pero al agregar a la solución debería quedar una estructura con respecto a las posiciones del arreglo de entrada par e impar en cada grupo.

Por ejemplo:

Dada la siguiente entrada previamente ordenada.

Fuerza trabajo	200	195	190	185	170	164	160	150
Posición	0	1	2	3	4	5	6	7
Paridad	PAR	IMPAR	PAR	IMPAR	PAR	IMPAR	PAR	IMPAR

Una posible estructura de para la implementación podría ser la siguiente:

Al querer insertar, pregunto por la diferencia de tamaños entre grupo 1 y grupo 2:

Si hay diferencia de tamaños entre los dos grupos, agrego en el más chico.

Si no hay diferencia, me fijo en agregar el nuevo elemento en el grupo donde la paridad con respecto al arreglo original no sea igual. La paridad de cada grupo sería la paridad con respecto a la posición del arreglo de entrada del último empleado que agregué a ese grupo.

Posición	**Paridad**	**GRUPO 1**
0	P	200
3	I	185
4	P	170
7	I	150

Posición	**Paridad**	**GRUPO 2**
1	I	195
2	P	190
5	I	164
6	P	160

Los grupos con respecto al arreglo de entrada, quedarían plasmados de la siguiente manera, siendo los casilleros naranjas el grupo 1 y los violetas el grupo 2:

Grupo	1	2	2	1	1	2	2	1
Fuerza trabajo	200	195	190	185	170	164	160	150
Posición	0	1	2	3	4	5	6	7
Paridad	PAR	IMPAR	PAR	IMPAR	PAR	IMPAR	PAR	IMPAR

Finalmente la decisión a implementar fue directamente agregar en el grupo que menos fuerza de trabajo acumulada tiene hasta el momento, ya que de la otra forma, restringe en que queden las cantidades similares en empleados, pero no era algo pedido en el problema. Al hacerlo de la forma anterior, si se recibía por entrada empleados con fuerzas de trabajo muy distantes al inicio, la equivalencia entre fuerzas era más dispar que si se analiza desde esta última forma planteada, teniendo en cuenta que, la cantidad de empleados podría quedar muy dispareja.

1.3 ¿Cuándo es factible agregar un candidato a la solución?

Siempre es factible agregar un candidato a la solución, ya que deben agregarse todos los empleados de entrada en algún grupo.

1.4 ¿Cuándo alcanzamos una solución al problema?

Se alcanza la solución del problema cuando el conjunto de candidatos queda vacío, habiendo agregado en diferentes grupos los empleados.

1.5 Resultado del contador de candidatos:

SOLUCIÓN	1	2	3	4	5	6	7	8	9
CANTIDAD CANDIDATOS	6	4	4	4	5	6	7	8	20

BACKTRACKING

2.0 Los detalles del diseño Backtracking del punto anterior:

2.1 ¿Qué es un estado?

Un estado estaría integrado de dos partes:

- Cambiar de posición en la lista de entrada de empleados, que contiene todos los empleados.

- Las soluciones que se van formando al agregar un empleado en el grupo 1 o 2 estrictamente.

2.2 ¿Qué datos contiene?

Los datos que contendría un estado serían:

La posición del arreglo de entrada (int), donde voy a buscar al empleado de la lista de empleados de entrada), la cual irá hacia la siguiente posición en cada cambio de estado.

La solución parcial, que pertenece a una clase de tipo Solución que esta compuesta por dos ArrayLists (Grupo 1 y 2) y los datos de la fuerza total de cada grupo. La misma varía por cada rama del árbol que voy a ir generando, donde se van a alternar los empleados en diferentes grupos conforme si estoy agregando a un empleado en el grupo 1 o 2.

2.3 ¿Cuándo un estado es final y cuándo es solución?

El estado final sería cuando ya decidí dónde ubicar a cada uno de los empleados, donde quedan todas las posibles soluciones.

Es solución cuando no tenga más decisiones por tomar, y donde defino quedarme con la mejor solución (la solución encontrada es mejor que la mejor solución actual). Esto se da cuando la diferencia entre las fuerzas de ambos grupos de la posible solución actual encontrada es menor a la diferencia de la mejor solución hasta el momento, es decir, la solución óptima será la que tiene las fuerzas de grupos más equilibrada (la menor diferencia de fuerzas).

2.4 ¿Cuáles son los hijos de un estado?

Los hijos serían los dos Backtracking, donde los siguientes empleados de entrada a partir de la posición actual.

2.5 ¿Qué modificaciones debo aplicar desde un estado para ir a un estado hijo?

Para cambiar de estado se pasa a la siguiente posición de la lista de empleados.

A su vez, se analiza agregar la posibilidad en ese estado en dos posibles decisiones:

- Lo agrego al grupo 1.
- O lo agrego al grupo 2.

2.6 ¿Qué poda se le puede aplicar al problema?

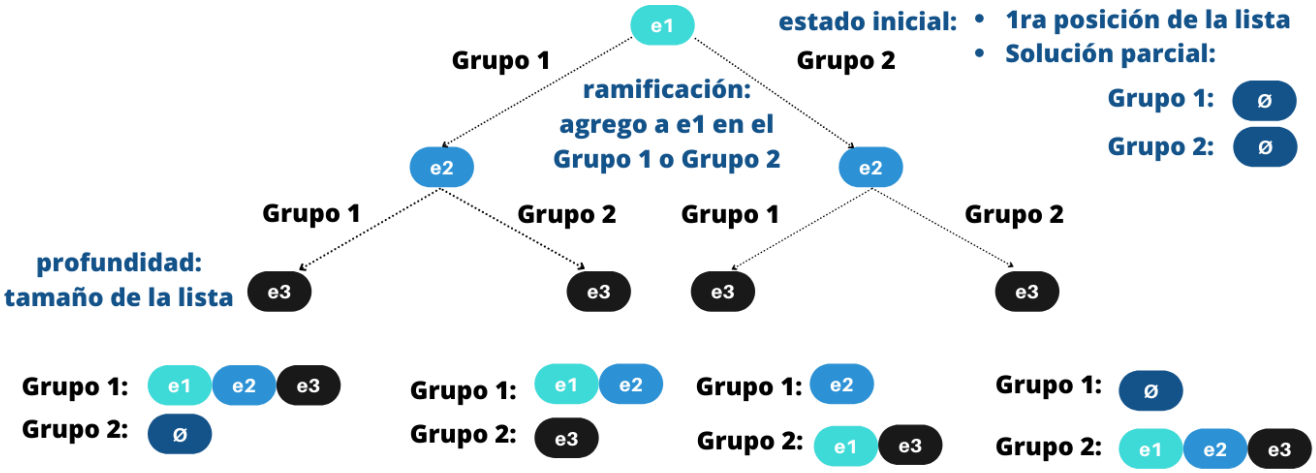
No encontré una posible poda para el problema.

2.7 Incluir un diagrama del árbol de exploración del algoritmo, indicando estado inicial, grado de ramificación y profundidad del árbol.

El estado inicial del árbol comienza con la solución vacía, por lo cual ambos grupos están vacíos, y la primera posición del arreglo de entrada, es decir el primer empleado.

El grado de ramificación es 2, ya que en el árbol de exploración puedo tener 2 posibilidades, si el empleado está en el Grupo 1 o en el Grupo 2 estrictamente.

La profundidad del árbol es la cantidad de empleados que tengo, es decir, el tamaño de la entrada de la lista de empleados.



2.8 Resultado del contador de estados:

SOLUCIÓN	1	2	3	4	5	6	7	8	9
CANTIDAD ESTADOS	127	31	31	31	63	127	255	511	2097151