

¿Que se puede ver en este documento?

En esta breve documentación se explica el objetivo del ejercicio entregable, sus características, y se describe la solución implementada.

Objetivo del ejercicio: Utilizar lo aprendido en las clases sobre algoritmos **Greedy** para intentar conseguir una aproximación a la mejor solución posible de un problema complejo.

Consigna:

Elon Musk decide abrir su taller espacial para permitirle a las familias que puedan ir a conocerlo. Luego de abierta la inscripción, 5000 familias se anotan para realizar la visita. Debido a la gran demanda de público decide establecer un número de 100 días durante los cuales aceptará visitas. Sabiendo que el taller tiene una capacidad diaria acotada (de **340** personas), Elon debe distribuir a los visitantes durante los 100 días. Para esto, solicita a cada familia que le informe **8** días (en orden de preferencia) en los cuales preferirían realizar la visita, y cuántas personas conforman el grupo familiar.

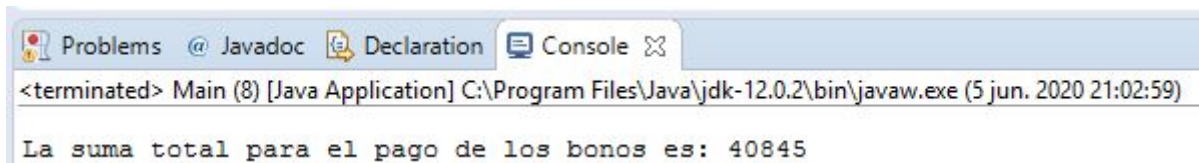
Dado que no todas las personas podrán asistir el día que prefieren, se otorga un bono compensatorio para las familias que no puedan asistir en su día preferido. El bono se calcula como $\text{U\$S } 25 + (\text{U\$S } 10 * \text{Grupo Familiar}) + (\text{U\$S } 5 * \text{Dia asignado})$.

Se pide un programa que permita determinar qué familias deberán ser invitadas en cada uno de los 100 días para **minimizar el monto total** de los bonos compensatorios, asegurándose de que todas las familias sean asignadas a uno de sus días de preferencia.

Se sabe que, para el caso de prueba provisto, el mínimo costo total del bono compensatorio es **29035**.

¿Que algoritmo se utilizó para la solución?

En este caso personal, se utilizó en esencia un algoritmo **Greedy**, el cual intentaba asignar a cada familia en su día de mayor preferencia posible. Si bien ya está aclarado que solo se busca una solución aproximada al problema, a simple vista es notorio que por sí solo este algoritmo no logra obtener una buena aproximación, sin mencionar que elegir el día más preferido posible no garantiza que todas las familias puedan asistir al evento.

A screenshot of a Java IDE's console window. The window has tabs for 'Problems', 'Javadoc', 'Declaration', and 'Console'. The 'Console' tab is active, showing the output of a Java application. The output text is: '<terminated> Main (8) [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (5 jun. 2020 21:02:59)' followed by a blank line and then 'La suma total para el pago de los bonos es: 40845'.

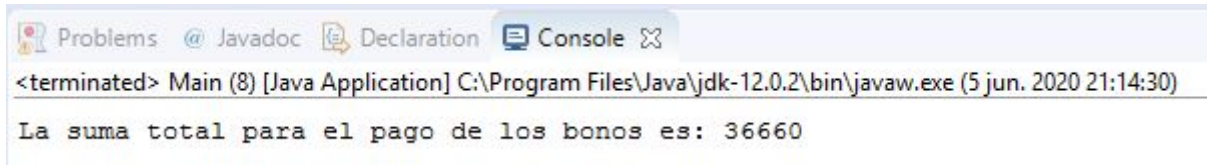
```
<terminated> Main (8) [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (5 jun. 2020 21:02:59)

La suma total para el pago de los bonos es: 40845
```

Como puede observarse en consola la aproximación obtenida está lejos de la solución óptima.

¿Existe alguna manera de disminuir la suma a pagar en indemnizaciones?

La respuesta es **SÍ**. Usando un poco el sentido común se puede notar que lo más conveniente es priorizar primero a las familias menos numerosas, esto como resultado disminuye en cierta medida la cantidad de familias indemnizadas. Debido a que las familias se encuentran registradas en una colección de datos (ArrayList), podemos ordenarlas en base a su cantidad de miembros gracias al método **sort** incluido en este tipo de estructuras de almacenamiento, por lo que no requiere una modificación en el algoritmo Greedy.



```
Problems @ Javadoc Declaration Console
<terminated> Main (8) [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (5 jun. 2020 21:14:30)
La suma total para el pago de los bonos es: 36660
```

Evidentemente esta pequeña modificación obtuvo una mejor aproximación, aunque todavía no esta lo suficientemente cerca de la solución óptima.

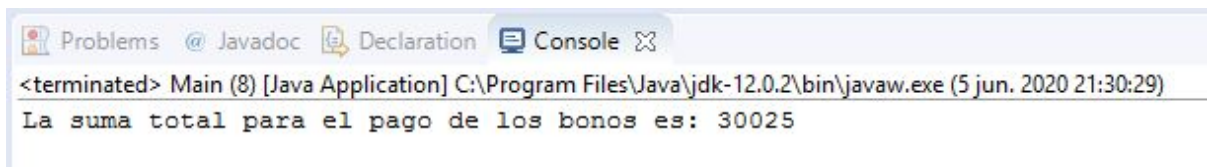
¿Es posible aproximarse aún más a la solución óptima?

Para lograr esto se utilizo un metodo de ordenamiento (Burbuja), el cual asigna a las familias a través de un criterio de selección más riguroso que simplemente su cantidad de miembros.

Este criterio determina **cuando** dos familias pueden **intercambiar** los dias en los que asistiran a la exposición, para esto el nuevo algoritmo debe tomar la solución anterior y “acomodarla” en base a los siguientes factores:

- Que el resultado de intercambiar de dia a las dos familias no genere que se sobrepase el cupo máximo permitido en ninguno de los dos días.
- Que ambas familias resulten ubicadas en alguno de sus días de preferencia.
- Que la suma de sus bonos luego de la reasignación, sea menor a la suma previa.

Este ordenamiento de complejidad $O(n^2)$ permitió refinar aún más la solución ya que agrega nuevas consideraciones. Finalmente se obtuvo un programa que prioriza pagar la menor cantidad de bonos posibles, al mismo tiempo que reacomoda a las familias que reciben dichos bonos de manera que su bono sea menor, y por último nos garantiza que puedan concurrir a la exposición el mayor número de familias posible.



```
Problems @ Javadoc Declaration Console
<terminated> Main (8) [Java Application] C:\Program Files\Java\jdk-12.0.2\bin\javaw.exe (5 jun. 2020 21:30:29)
La suma total para el pago de los bonos es: 30025
```

Esta vez obtuvimos una aproximación muy cercana a la solución óptima.

¿Cómo trabaja Greedy?

El denominado “**Algoritmo Greedy**” “*algoritmo voraz*” sigue una estrategia sencilla. Simplemente, se trata de elegir la opción óptima en cada paso local, con la esperanza de llegar a una solución general óptima. Este algoritmo no se preocupa de los pasos que restan hasta encontrar la solución. Un algoritmo de este tipo nunca deshace una decisión ya tomada: una vez incorporado, un “*candidato a la solución*” (Cuál es el mejor día posible para ubicar a una familia en este caso) formará parte de la solución. Y cada candidato rechazado es eliminado definitivamente.