# INFORME TRABAJO ENTREGABLE GREEDY

## PROGRAMACIÓN 3 - TUDAI 2020

Chiesa Conrado Esteban

**Problema planteado:**

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.

*Por ejemplo, la familia Pérez tiene 3 miembros, y sus preferencias son [33, 25, 10, 62, 48, 53, 66, 78]. El día 33 es el día preferido para los Perez, de no ser posible ir el 33, el segundo día preferido es el 25, y así sucesivamente.*

Dado que no todas las personas podrán asistir el día que prefieren, Elon decide crear un bono compensatorio para las familias que no son asignadas al día que eligieron como prioritario.

**Primera aproximación al problema:**

Como primer solución se consideró la idea de ordenando los las familias por día de preferencia, luego recorriendo los días agregar familias a estos hasta completar el cupo de personas, esto se realizaría en una copia de la lista original de familias para borrarlas de la lista una vez que fueron agregadas a un día. Una vez terminada esta primera etapa los días deberían estar llenos con las familias que tenían ese día de preferencia y en la lista quedarían las familias que no van su día de preferencia.

En la segunda etapa se itera a las familias restantes y se les trata de asignar el día de mayor preferencia. En este bloque debemos asignar el costo de no llevar a la familia en su día de preferencia.

Conclusión: La solución planteada se acerca a una solución posible, lo primero que se detectó es que era indistinto ordenar a las familias ya que al algoritmo planteado no le cambia la funcionalidad.

Esta solución no cumple con lo solicitado ya que 2 familias quedan sin asignar y el costo en bonos es de U$S 33075

La solución concluye en invalida pero con buena aproximación.

Código del algoritmo:

```
  public void distribuirFamilias () {
     ArrayList<Familia> copiaFamVis = new ArrayList<>(this.familiasVisitantes);
     int bono = 0;
     Iterator<Familia> itFamilia = copiaFamVis.iterator();
     while (itFamilia.hasNext()) {
        Familia actual = itFamilia.next();
        if (actual.miembros() < diasDeVisita.get(actual.diaPreferido()-1).lugaresDisponibles())
{
           diasDeVisita.get(actual.diaPreferido()-1).agregarVisitantes(actual.miembros());
           itFamilia.remove();
        }
     }

     if (!copiaFamVis.isEmpty()) {
        itFamilia = copiaFamVis.iterator();
        boolean agregada = false;
        while (itFamilia.hasNext()) {
           Familia actual = itFamilia.next();
           for (int i = 1; i < 8; i++) {
              if (diasDeVisita.get(actual.preferenciaEn(i)-1).lugaresDisponibles() >=
actual.miembros()) {
                 agregada =
diasDeVisita.get(actual.preferenciaEn(i)-1).agregarVisitantes(actual.miembros());
                 bono += 25 + (10 * actual.miembros()) + (5 * i);
                 break;
              }
           }
           if (agregada) {
              itFamilia.remove();
              agregada = false;
           }
        }
     }

     System.out.println(copiaFamVis.toString());

     System.out.println("U$S " + bono);
  }
```

RTA:

Familias no agregadas:

Familia: id=4846, miembros=4, preferencias=[1, 32, 66, 47, 25, 60, 4, 11]
Familia: id=4894, miembros=4, preferencias=[26, 24, 40, 47, 31, 52, 4, 1]

bonos: U$S 33075

**Segunda aproximación al problema:**

La próxima propuesta consiste en ordenar las familias según su cantidad de miembros de menor a mayor y utilizando el mismo algoritmo, de esta manera mayor cantidad de familias se suman antes a los días.

RTA:
Las familias fueron todas asignadas a alguno de sus días preferidos.
El costo en bonos es de U$S 38555

A diferencia de la aproximación anterior se ordenaron las familias con un criterio

**Tercer aproximación al problema:**

La siguiente resolución consiste en ordenar las familias en orden ascendente en cuanto a sus miembros para asignar la mayor cantidad de familias en su día preferido. Se sigue utilizando el mismo algoritmo para asignar las familias a los diferentes días.

RTA:
Las familias fueron todas asignadas a alguno de sus días preferidos.
El costo en bonos es de U$S 29965

**Cuarta aproximación al problema:**

En orden a mejorar el algoritmo se intentó llevar el segundo bloque de asignación al bloque anterior sumando un else al if del primer iterador pero el resultado fue más costo en cuanto a valor de bono pero se reduce el tiempo de procesamiento.
A continuación se muestra el bloque modificado:

```
if (actual.miembros() < diasDeVisita.get(actual.diaPreferido()-1).lugaresDisponibles()) {
  diasDeVisita.get(actual.diaPreferido()-1).agregarVisitantes(actual.miembros());
  itFamilia.remove();
} else {
  for (int i = 1; i < 8; i++) {
    if (diasDeVisita.get(actual.preferenciaEn(i)-1).lugaresDisponibles() >=
actual.miembros()) {
      agregada =
diasDeVisita.get(actual.preferenciaEn(i)-1).agregarVisitantes(actual.miembros());
      bono += 25 + (10 * actual.miembros()) + (5 * i);
      break;
    }
  }
  if (agregada) {
    itFamilia.remove();
    agregada = false;
  }
```

}

RTA:

Las familias fueron todas asignadas a alguno de sus días preferidos.
El costo en bonos es de U$S 36660

**Quinta aproximación al problema:**

En base a el debate realizado en la clase del Miércoles 03/06 se plantea agregar a un arraylist las familias que no fueron asignadas en su día preferido. al contrario de removerlos de la lista.

RTA:

Es una operación menos costosa ya que no realiza la copia de la lista porque no los remueve sino que las familias que no son agregadas en su día preferido luego son iteradas con la misma operación que venimos trabajando.

**CONCLUSIÓN:**

Ordenar las familias en orden ascendente según su cantidad de miembros permitió acomodar mayor cantidad de familias en los días abiertos y de esta forma minimizar el costo en bonos. Cambiar el remove() por un add() considero que fue una muy buena optimización ya que no crea una copia y solo las familias que no fueron asignadas en su día preferido son copiadas a la lista que se itera a continuación en el algoritmo

**MEJORA:**

Como mejora al greddy se propone hacer una las familias que hayan sido asignadas en un día de preferencia mayor al del índice 3, luego se trata de agregar a esta familia (familia actual) en su día preferido, si no es posible porque no hay cupo suficiente se quita a la última familia (familia de intercambio) agregada ese día y si queda lugar con esta modificación las 2 familias son quitadas del día el que habían sido asignadas y se las vuelve a asignar comenzando por la familia actual y siguiendo por la familia de intercambio.