# Tabajo practico entregable

## Greedy

El algoritmo greedy debe tomar la mejor decision en un momento dado con la esperanza de que cuando se terminen de tomar todas las decisiones posibles se llegue a la mejor solucion general, no siempre se encuentra la mejor solucion general de un problema ya que a tomar una decision solo evalua las opciones a “corto plazo” y si esta decision tomada no fue la mejor ya no se vuelve a considerar. El uso de greedy para resover un problema tiene como objetivo encontrar una solucion de la forma mas eficiente y buscar la optimizacion.

Inicialmente el planteo fue seleccionar pares de candidatos pero la idea de generar todos los pares alumno-libro posibles resulto poco comprensible. Tambien se considero la idea de tomar un libro como candidato, pero se descarto la idea porque por la implementacion realizada, finalizaba con un numero muy alto de iteraciones.

Finalmente los candidatos son los alumnos, el criterio para seleccionar un alumno es evaluarlos en un orden determinado para optimizar la seleccion, primero van los que tienen la nota mas alta. Una vez seleccionado se le asignaran libros hasta que apruebe o que no queden libros que pueda leer. Al seleccionar un libro para el alumno se toman en cuenta primero los de mayor puntaje, en caso de que el alumno alcanze la nota de aprobacion se busca que apruebe de la forma menos holgada posible.

Como la biblioteca puede tener miles de libros se tomaron medidas para optimizar la seleccion del libro, esta implementacion se llama *optimizacion del salto*, en el peor de los casos se evaluan todos los libros, pero en la mayoria de las situaciones y especialmente cuando hay gran numero de libros, se reduce de manera considerable la cantidad de iteraciones sobre la lista de libros sin influir negativamente de forma notoria en el resultado final de la asignacion de libros a un alumno. (ver linea 158 del archivo Greedy.java)

Otro elemento de optimizacion fue no generar una instacia de Libro diferente para cada ejemplar, sino que una unica instancia del libro sabe cuantos ejemplares tiene y se van restando a medida que se asignan a los alumnos.

Considerando que la consigna es maximizar la cantidad de alumnos aprobados y no dice explicitamente que la solucion es que aprueben todos, entiendo que hay tres posibles condiciones de finalizacion del algoritmo. La primera condicion de corte es que apueben todos los alumnos, la segunda es que se asignen todos los libros, la tercera es que no aprueben todos los alumnos pero al mismo tiempo los libros que queden disponibles no puedan ser asignados debido a que ya fueron leidos por todos los alumnos.

Es factible agregar un candidato a la solucion cuando a este se le asigna un libro, teniendo en cuenta que el algoritmo debe informar que libros se deben asignar a cada alumno.

##### Resultados dataset1.csv

Nota para aprobar asignatura: **30**

El contador finaliza con un total de 21 iteraciones.

**Resutado:** Aprueban 5 alumnos.

Nota para aprobar asignatura: **50**

El contador finaliza con un total de 17 iteraciones.

**Resutado:** Aprueban 3 alumnos.

Nota para aprobar asignatura: **100**

El contador finaliza con un total de 11 iteraciones.

**Resutado:** Aprueba 1 alumno.

##### Resultados dataset2.csv

Nota para aprobar asignatura: **30**

El contador finaliza con un total de 52 iteraciones.

**Resutado:** Aprueban 5 alumnos.

Nota para aprobar asignatura: **50**

El contador finaliza con un total de 53 iteraciones.

**Resutado:** Aprueban 5 alumnos.

Nota para aprobar asignatura: **100**

El contador finaliza con un total de 35 iteraciones.

**Resutado:** Aprueban 5 alumnos.

Nota para aprobar asignatura: **200**

El contador finaliza con un total de 45 iteraciones.

**Resutado:** Aprueban 5 alumnos.

Nota para aprobar asignatura: **500**

El contador finaliza con un total de 74 iteraciones.

**Resutado:** Aprueban 3 alumnos.

##### Resultados dataset3.csv

Nota para aprobar asignatura: **500**

El contador finaliza con 78 iteraciones. \*(sin optimizacion del salto son 2631)

**Resutado:** Aprueban 5 alumnos.

Nota para aprobar asignatura: **1000**

El contador finaliza con 111 iteraciones. \*(sin optimizacion del salto son 1965)

**Resutado:** Aprueban 5 alumnos.

Nota para aprobar asignatura: **2000**

El contador finaliza con 214 iteraciones. \*(sin optimizacion del salto son 3298)

**Resutado:** Aprueban 5 alumnos.

Nota para aprobar asignatura: **5000**

El contador finaliza con 503 iteraciones. \*(sin optimizacion del salto son 2836)

**Resutado:** Aprueban 5 alumnos.

Nota para aprobar asignatura: **20000**

El contador finaliza con 2933 iteraciones. \*(sin optimizacion del salto son 3770)

**Resutado:** Aprueban 4 alumnos.

Nota para aprobar asignatura: **30000**

El contador finaliza con 3814 iteraciones. \*(sin optimizacion del salto son 4017)

**Resutado:** Aprueban 2 alumnos.

## Backtacking