Proyecto de Fundamentos de análisis y diseño de algoritmos

Integrantes:
Camilo José Cruz Rivera - 1428907
Erik López Pacheco - 1430406
Robert leandro Quiceno - 1422913

Profesor:
Jesus Alexander Aranda

Universidad del valle
Facultad de ingeniería
Escuela de ingeniería de sistemas y computación EISC
Santiago de cali 15 de junio
2017

Tabla de contenido

1. Introduccíon

En el siguiente informe se evidenciara el desarrollo para el proyecto propuesto del curso de Fundamentos de análisis y diseño de algoritmos. Empezado desde la planeación de la solución a los problemas planteados, la implementación de las respectivas soluciones y un análisis de los resultados obtenidos después de aplicar los algoritmos pedidos para la solución (Algoritmo ingenuo o exhaustivo, algoritmo voraz y algoritmo dinámico)

2. Análisis de la complejidad temporal de las soluciones implementadas

Para el problema 1 (Sala de operaciones):
para la solución ingenua o exhaustiva se esperaba que está solución fuera muy ineficiente, ya que los algoritmos de este tipos por lo general solucionan el problema planteado pero haciendo comparaciones y recorriendo todos los datos de forma no apropiada, sobre usando recursos computacionales. y al implementar este algoritmo se observó que se comportó tal y como se esperaba. Está solución tuvo una complejidad de $O(n^2)$ debido a la cantidad de comparaciones realizadas para encontrar la solución

Para la solución Dinámica…

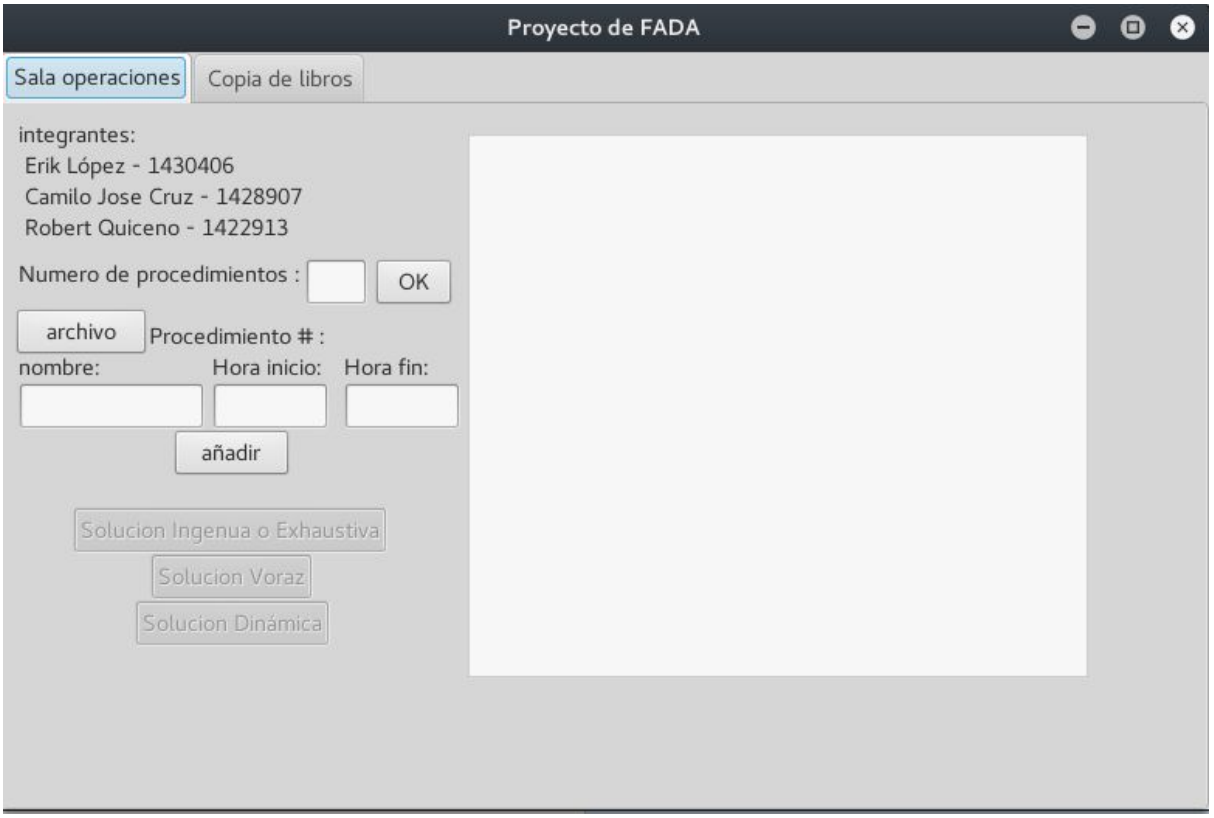
Para la solución Voraz se trabajó bajo el supuesto de que primero debería entrar el procedimiento que tiene la mayor duración para así ir garantizando el mayor uso de la sala a medida que se vayan agregando procedimientos a la solución verificando que no se crucen con los procedimientos que ya están seleccionados.
La complejidad para está solución sería de $O(cn)$ donde n es el tamaño de la entrada y c una constante que se define dentro de la ejecución del algoritmo.

es de comportamiento lineal porque solo hace un recorrido de toda la lista de procedimientos.

Para el problema 2 (Copia de libros):
#

3. Detalles principales de la implementación

Para fines didácticos y como reto personal decidimos implementar las soluciones en python, ya que no estábamos muy familiarizados con su sintaxis y nos pareció ideal aprender más de python por medio de la solución de este proyecto.
Para implementar la interfaz gráfica del proyecto, se hizo uso de la librería wx porque nos pareció más sencilla de implementar los botones con los eventos y todo lo relacionado con la GUI (para ejecutar el proyecto es necesario tener la librería en la máquina donde se vaya a ejecutar la aplicación)
Después de tener la interfaz implementada pensando en cómo se visualizará la soluciones, obtuvimos la siguiente interfaz:



Una vez teniendo la interfaz completamente funcionando, se prosiguió con las implementaciones de los algoritmos.
donde cada uno de los problemas requería la lectura de un archivo que contiene los datos necesarios para funcionar y los resultados mostrarlos en la ventada de la GUI

4. Descripción y análisis de las pruebas realizadas

```
while len(listaDePesos)>0:

    if sum == 0:
        maxx = max(listaDePesos)
        ind = listaDePesos.index(maxx)
        ProcedimientosARealizar.append(ListProc[ind])
        ListProc.remove(ListProc[ind])
        listaDePesos.remove(maxx)
        sum = sum + maxx
    else :
        maxx = max(listaDePesos)
        ind = listaDePesos.index(maxx)

        if not cruzan(ProcedimientosARealizar[aux],ListProc[ind]):
            ProcedimientosARealizar.append(ListProc[ind])
            ListProc.remove(ListProc[ind])
            listaDePesos.remove(maxx)
            sum = sum + maxx
            aux = len(ProcedimientosARealizar) - 1
        else :
            if not cruzan(ListProc[ind],ProcedimientosARealizar[aux2]):
                ProcedimientosARealizar.append(ListProc[ind])
                ListProc.remove(ListProc[ind])
                listaDePesos.remove(maxx)
                sum = sum + maxx
                aux2 = len(ProcedimientosARealizar)-1
            else:
                ListProc.remove(ListProc[ind])
```

en la ejecución de la prueba del algoritmo voraz con 8 procedimientos se demoró 0.000202894210815 sg

5. Conclusiones y aspectos a mejorar

Con la realización de este proyecto, concluimos que cualquiera de las dos estrategias voraz o dinámica son apropiadas y ambas encuentran de manera eficiente la solución, muy por encima a la técnica ingenua, y para comparar entre las dos estrategias mencionadas la voraz nos pareció más intuitiva al momento de implementarla pero la dinámica no fue sencillo construir la solución de forma recursiva ni de implementar las matrices donde se guardarán los costos de las posibles soluciones. aunque ambas soluciones son buenas dependiendo del problema el cual abordan donde se vería que una se comporte mejor que la otra.

Para mejorar consideramos que es importante mejorar la forma de modelar soluciones para la estrategia dinámica, ya que la mayoría de los problemas de computo se resuelven mejor así y en ese aspecto estamos algo quedados