
Solución de un problema mediante el algoritmo Avance Rápido

Universidad de Murcia
2º Grado en Ingeniería Informática
Asignatura: Algoritmos y estructuras de datos II
Curso 2017/2018

Proyecto realizado por:
Nicolás Enrique Linares La Barba
Miguel Ángel Lucas Barceló

Índice

1. Problema	3
2. Diseño	3
3. Explicación de las variables.....	3
4. Explicación de las funciones	4
5. Estudio teórico del tiempo de ejecución	5
6. Estudio experimental	6
7. Contraste del estudio teórico y experimental	6

1. Problema

Dado un conjunto de N alumnos representados en base a un valor que expresa el nivel de amistad y trabajo entre cada pareja. Encontrar la mejor combinación de parejas de alumnos.

2. Diseño

Para el diseño de este problema se parte del esquema general de un algoritmo voraz, el cual tiene la siguiente estructura:

```
voraz (C: CjtoCandidatos; var S: CjtoSolución)  
  S:= Ø  
  mientras (C ≠ Ø) Y NO solución(S) hacer  
    x:= seleccionar(C)  
    C:= C - {x}  
    si factible(S, x) entonces  
      insertar(S, x)  
    fin si  
  fin mientras  
  si NO solución(S) entonces  
    devolver "No se puede encontrar solución"  
  fin si
```

Hemos respetado tanto la estructura, como los parámetros pasados a la función de *avanceRapido* y la utilización de las funciones y procedimientos que se muestran en el esquema para su mayor comprensión y semejanza.

3. Explicación de las variables

En cuanto a las variables utilizadas, se comienza por almacenar en variables globales los casos a resolver en "ncasos", el número de alumnos "alumnos" y las matrices "amistad" y "trabajo", donde se encuentran los datos del problema. La variable global "total" almacenará el beneficio total obtenido para cada uno de los casos de prueba.

Se llama al algoritmo pasándole como parámetros el conjunto de candidatos "candidatos" y el conjunto solución "solucion", ambos de tamaño "alumnos", es decir el número de alumnos de la clase. El primero va a ser el array de alumnos candidatos y el segundo un array que va a almacenar las parejas que conforman la solución.

Cada índice del array “solucion” es uno de los dos miembros y cada valor almacenado es el compañero asignado. Se inicializa a -2 que significa que ese alumno no ha sido comprobado. Y a medida que se van emparejando alumnos, ponemos un -1 al compañero asignado para indicar que ya tiene un compañero. Como se indica a continuación.

S:

0	1	2	3	4
2	3	-1	-1	-2

El ejemplo muestra una solución con 2 parejas y un alumno solo, donde el alumno 0 está emparejado con el 2, el alumno 1 con el 3 y el 4 no puede tener pareja porque es impar.

En el caso del array “candidatos” va a ser de tipo booleano para llevar cuenta de los alumnos escogidos, donde true significa que el alumno está libre y no tiene compañero.

4. Explicación de las funciones

Funciones y procedimientos principales:

- *AvanceRapido(c, s)*: algoritmo voraz que nos proporcionará el resultado buscado.
- *Solucion(s)*: va a indicar si el conjunto “s” forma ya una solución válida, es decir, si todos los alumnos han sido seleccionados, pudiendo quedar uno solo. Para hacer esta comprobación se recorre el array contando cuantos alumnos sin escoger hay (con -2), por lo que si este valor es 0 o 1 ya formaría una solución.
- *Seleccionar(c, alumnoActual)*: esta función va a devolver el alumno no elegido anteriormente que mayor relación tiene con el “alumnoActual” pasado como parámetro. Se recorre el conjunto “c” de candidatos (aquellos con valor true) guardando el de mejor resultado y una vez escogido nuestro candidato, no se podrá deshacer.
- *Factible()*: en este caso no se ha añadido la función factible, pues siempre va a ser factible la solución que vamos a construir. Por lo tanto, es innecesaria para este problema.
- *Insertar(alumnoActual, pareja, s)*: añade a la posición “alumnoActual” el candidato “pareja” seleccionado previamente y pone -1 en las posiciones contrarias para que cuando se recorra el conjunto solución no repita las parejas.
- *Objetivo()*: esta función está implícita en *seleccionar()*, donde se va almacenando el grado de compenetración total de todas las parejas en la variable “total”, con lo cual ya no es necesaria esta función.

Otras funciones y procedimientos utilizados:

- *LeerEntrada()* : almacenará los datos proporcionados para cada problema, inicializando los valores de las matrices "amistad" y "trabajo".
- *MostrarParejas(s)*: recorre el conjunto solución imprimiendo las parejas que se han formado.

5. Estudio teórico del tiempo de ejecución

$\begin{cases} j = \text{alumnoActual} \\ n = \text{alumnos} \end{cases}$

$$t_{AR}(n) = \overbrace{\sum_{i=1}^n (3) + 1}^{\text{Inicialización}} + \sum_{j=1}^{n/2} (1 + \overbrace{S_0 + 1 + S_e + I_n}^{\text{Selección}} + \overbrace{\sum_{j=1}^n (2) + 2 + 1 + S_d}^{\text{Insertar}})$$

$$* \underbrace{t(n)}_{\text{Solución 0}} = 1 + \underbrace{\sum_{i=1}^n (2)}_{\text{for e if}} + \underbrace{\sum_{i=1}^{n-j-2} (1)}_{n++} + 1 = \boxed{2 + 3n - 2j}$$

cada vez que se ejecuta hay dos espacios menos que comprobar.

$\{K = \text{candidato}\}$

$$* \underbrace{t(n)}_{\text{Selección}} = 3 + \underbrace{\sum_{k=1}^n (2)}_{\text{for e if se ejecuta siempre}} + \underbrace{\sum_{k=1}^{n-j-2} (2)}_{\text{aux e if}} + \underbrace{\frac{1}{j} \cdot 2 + 4}_{\text{comparación mejor alumno}} = \boxed{4 \cdot n - \frac{4j^2 + 2}{j} + 7}$$

$$* \underbrace{t(n)}_{\text{Insertar}} = \boxed{2}$$

$$* \underbrace{t(n)}_{\text{Solución 1}} = 1 + \sum_{i=1}^n (2) + 1$$

Siempre hay solución en este problema se puede omitir

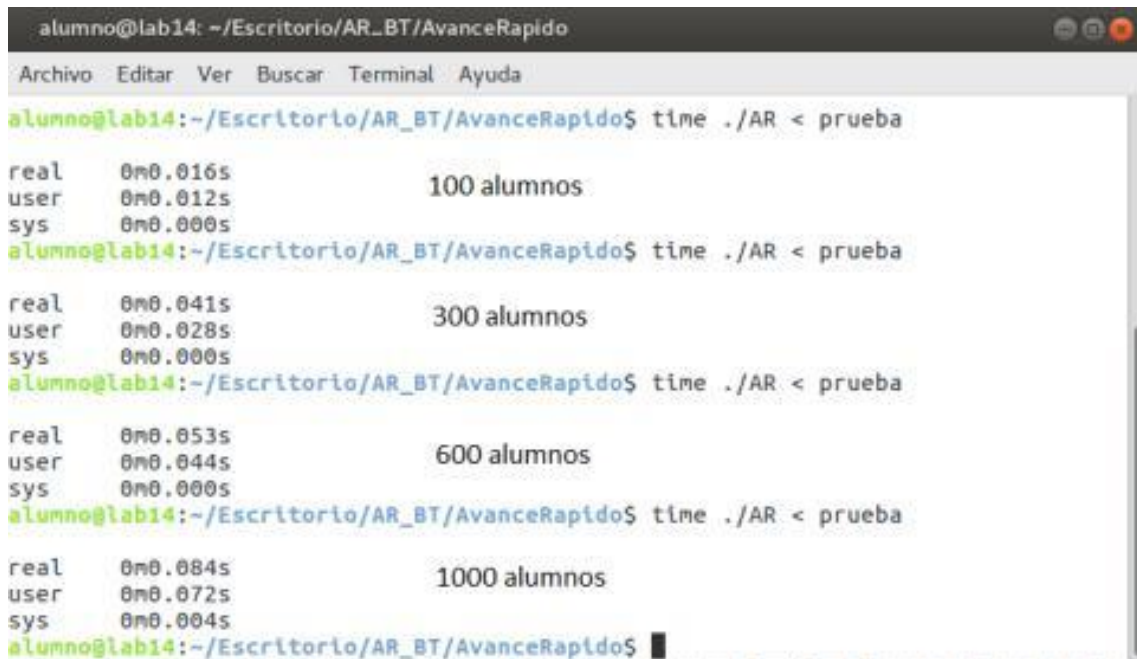
$$t_{AR}(n) = 3n + 1 + \frac{n}{2} \cdot (2 + (2 + 3n - 2j) + (4n - \frac{4j^2 + 2}{j} + 7) + 2) + 2n + 3$$

$$= 5n + 4 + \frac{n}{2} \cdot (13 + 7n - 6j + \frac{2}{j}) = \boxed{\frac{7n^2}{2} - \frac{6j \cdot n}{2} + \frac{n}{j} + \frac{23n}{2} + 4}$$

$$\boxed{\frac{7}{2} n^2 - \frac{6j \cdot n}{2} + \frac{n}{j} + \frac{23n}{2} + 4} \approx \underline{\underline{O(n^2)}}$$

6. Estudio experimental

Para el análisis experimental se ha estudiado cuatro casos diferentes. Cada problema trabaja con unos datos entradas distintos, que varían en el número de alumnos que hay que emparejar y los valores para las relaciones entre ellos de amistad y trabajo. Para obtener mejor resultado se ha quitado la salida del programa donde muestra los resultados de la solución. Podemos verlo en la siguiente imagen.



```
alumno@lab14: ~/Escritorio/AR_BT/AvanceRapido
Archivo Editar Ver Buscar Terminal Ayuda
alumno@lab14:~/Escritorio/AR_BT/AvanceRapido$ time ./AR < prueba
real    0m0.016s          100 alumnos
user    0m0.012s
sys     0m0.000s
alumno@lab14:~/Escritorio/AR_BT/AvanceRapido$ time ./AR < prueba
real    0m0.041s          300 alumnos
user    0m0.028s
sys     0m0.000s
alumno@lab14:~/Escritorio/AR_BT/AvanceRapido$ time ./AR < prueba
real    0m0.053s          600 alumnos
user    0m0.044s
sys     0m0.000s
alumno@lab14:~/Escritorio/AR_BT/AvanceRapido$ time ./AR < prueba
real    0m0.084s          1000 alumnos
user    0m0.072s
sys     0m0.004s
alumno@lab14:~/Escritorio/AR_BT/AvanceRapido$
```

7. Contraste del estudio teórico y experimental

Podemos observar cómo se va incrementando el tiempo de ejecución a medida que aumentamos el número de alumnos. Tardando hasta 5 veces entre el primer caso y el último, esto es debido al valor de N , es decir, al valor del número de alumnos del problema, hay que destacar que los valores de los vínculos vía amistad y trabajo entre ellos también pueden variar el tiempo de ejecución, pues los alumnos que ya hayan sido escogidos no van a volver a ser comprobados, por lo que el orden en el que comprobemos cada alumno afectará al resultado.

En general, a medida que utilizamos más alumnos, más aumentará el tiempo de ejecución del algoritmo, que es el resultado obtenido en el estudio teórico, por lo que no encontramos discrepancias.