PRÁCTICA 3

ALGORITMOS VORACES

Antonio Jiménez Rodríguez 2D (D2)

Problema asignado

Ejercicio 3:

Se han organizado cursos de esquí en Sierra Nevada para los que se dispone de esquíes de diversos tamaños. A cada alumno se le deben asignar unos esquís cuyo tamaño sea lo más cercano posible a su altura. Sabiendo las alturas de los alumnos y los tamaños de los esquís disponibles, es necesario hacer una asignación de los mismos tal que el error cuadrático medio sea mínimo. Suponiendo que hay N alumnos y M esquís de longitud L(j) (1≤j≤M), H(i) es la altura del alumno i, A(i) es la longitud del esquí asignado al alumno i, el error cuadrático medio se calcula de la siguiente forma:

$$E = \frac{1}{N} \sum_{i=1}^{N} (H(i) - L(j))^2$$

1. Análisis y entendimiento del problema

Nos vamos a plantear un conjunto de cuestiones:

- **¿En qué consiste el problema?**

Asignar los esquís disponibles a cada uno de los alumnos del curso minimizando el error cuadrático medio.

- **¿Todos los alumnos deben tener esquíes?**

Si, a todos los alumnos hay que darles esquís, se entiende que M >= N, ya que no tendría sentido que un alumno no pudiera impartir el curso.

- **¿Cada asignación de esquís y alumno tiene que ser exacta?**

No estrictamente pero si puede ser necesario para la minimización del problema, pero si no hay de la talla del alumno se le asigna uno de talla próxima a la suya. El objetivo de buscar una asignación correcta es minimizar el error cuadrático total de cada pareja esquí-alumno.

2. Componentes voraces del algoritmo

Un algoritmo voraz podrá aplicarse siempre que se pueda:

- **Diseñar una lista de candidatos para formar una solución**

Los errores cuadráticos de cada posible combinación de esquí-alumno (sin repetir ni esquíes ni alumnos).

- **Identificar una lista de candidatos ya utilizados**

Todos los alumnos o esquís que se hayan seleccionado para añadirlos al conjunto solución no podrán volver a ser seleccionados para la solución.

- **Diseñar una función solución para saber cuando un conjunto de candidatos es solución al problema**

Cuando su error cuadrático es el menor de las posibilidades que tiene.

- **Diseñar un criterio de factibilidad (cuándo un conjunto de candidatos pueda ampliarse para formar una solución final)**

Mientras que haya alumnos aún sin asignación de esquíes seguimos añadiendolos al conjunto solución.

- **Diseñar una función selección del candidato más prometedor para formar parte de la solución**

Vamos recorriendo alumno a alumno y seleccionando los esquíes disponibles que menor error cuadrático produzca para el resultado del algoritmo.

- **Existe una función objetivo de minimización/maximización**

La función objetivo es encontrar la asignación de esquíes a alumnos que produzca menor error cuadrático de la forma que se indica en el enunciado del problema. Es decir:

$$\text{Minimizar } \frac{1}{N} \sum_{i=1}^{N} (H(i) - L(i))^2.$$

3. Adaptación a la plantilla de algoritmos voraces

```
ALGORITMO P= AsignacionEsquies(Matriz M, n, m)
E={1...m}   // Candidatos, esquies a asignar
A={0}n      // Vector de esquies asignados a cada alumno, solucion a devolver

PARA i=1 HASTA n HACER:
    // Siendo la matriz M la matriz de errores cuadraticos
       de las parejas esqui-alumno
    j = Seleccionar esqui j (0<j<=m) en E donde M[j][i] sea minimo
    U = U\{j}
    X(i) = j
FIN PARA
DEVOLVER A
```

En la imagen anterior tenemos el pseudocódigo del algoritmo voraz. Como elementos de entrada tenemos la matriz M_{mxn}, ésta contiene el error cuadrático de cada asignación entre esquí y alumno posible. Además se le pasa el tamaño de la misma que corresponde al número de alumnos (n) y al número de esquíes (m).

El algoritmo ejecuta un bucle que recorre la matriz por columnas. Deberemos recorrer estas para cada alumno, ya que se corresponde con los esquíes disponibles, quedándonos así con el que produzca menor error cuadrático. Cada elección de esquí lo irá añadiendo al vector solución a devolver y eliminandolo de la lista de candidatos.

Una vez terminado el bucle tendremos un conjunto solución con las asignaciones de los esquíes.

4. Ejemplo

Vamos a ver el funcionamiento de nuestro algoritmo con un ejemplo práctico: $\{5, 4, 8, 7, 3, 9, 6\}_n$

Siendo i cada alumno y j cada esquí, tenemos 7 alumnos (n=7) con las alturas correspondientes de cada uno {170, 168, 181, 175, 163, 185, 173}; y tenemos 9 esquíes (m=9) con sus medidas correspondientes {160, 160, 165, 165, 170, 170, 170, 180, 190}. Con estos datos podemos generar la matriz de error cuadrático:

	i = 1	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7
j = 1	100	64	441	225	9	625	169
j = 2	100	64	441	225	9	625	169
j = 3	25	9	256	100	4	400	64
j = 4	25	9	256	100	4	400	64
j = 5	0	4	121	25	49	225	9
j = 6	0	4	121	25	49	225	9
j = 7	0	4	121	25	49	225	9
j = 8	100	144	1	25	289	25	49
j = 9	400	484	81	225	729	25	289

El error cuadrático de cada casilla lo hemos calculado tal y como indica el enunciado del problema $(H(i) - L(i))^2$.

Ahora iteraríamos e iríamos escogiendo el esquí con menos error cuadrático para alumno.

i = 1: A = $\{5\}_n$ // Escogemos el esquí número 5 para el alumno 1

	i = 2	i = 3	i = 4	i = 5	i = 6	i = 7
j = 1	64	441	225	9	625	169
j = 2	64	441	225	9	625	169
j = 3	9	256	100	4	400	64
j = 4	9	256	100	4	400	64
j = 6	4	121	25	49	225	9
j = 7	4	121	25	49	225	9
j = 8	144	1	25	289	25	49
j = 9	484	81	225	729	25	289

i = 2: A = $\{5, 6\}_n$ // Escogemos el esquí número 6 para el alumno 2

	i = 3	i = 4	i = 5	i = 6	i = 7
j = 1	441	225	9	625	169
j = 2	441	225	9	625	169
j = 3	256	100	4	400	64
j = 4	256	100	4	400	64
j = 7	121	25	49	225	9
j = 8	1	25	289	25	49
j = 9	81	225	729	25	289

i = 3: A = $\{5, 6, 8\}_n$ // Escogemos el esquí número 8 para el alumno 3

	i = 4	i = 5	i = 6	i = 7
j = 1	225	9	625	169
j = 2	225	9	625	169
j = 3	100	4	400	64
j = 4	100	4	400	64
j = 7	25	49	225	9
j = 9	225	729	25	289

i = 4: A = $\{5, 6, 8, 7\}_n$ // Escogemos el esquí número 7 para el alumno 4

	i = 5	i = 6	i = 7
j = 1	9	625	169
j = 2	9	625	169
j = 3	4	400	64
j = 4	4	400	64
j = 9	729	25	289

i = 5: A = $\{5, 6, 8, 7, 3\}_n$ // Escogemos el esquí número 3 para el alumno 5

	i = 6	i = 7
j = 1	625	169
j = 2	625	169
j = 4	400	64
j = 9	25	289

i = 6: A = $\{5, 6, 8, 7, 3, 9\}_n$ // Escogemos el esquí número 9 para el alumno 6

	i = 7
j = 1	169
j = 2	169
j = 4	64

i = 7: A = $\{5, 6, 8, 7, 3, 9, 4\}_n$ // Escogemos el esquí número 4 para el alumno 7

Al final hemos obtenido el conjunto $\{5, 6, 8, 7, 3, 9, 4\}_n$ que son los esquíes a seleccionar para cada alumno con un error cuadrático medio:

$$E = \frac{1}{N} \sum_{i=1}^{N} (H(i) - L(i))^2 = (0 + 4 + 1 + 25 + 4 + 25 + 64)/7 = 123/7 = 17.57$$

5. Análisis teórico de la eficiencia

Vamos a pasar ahora con el estudio de la eficiencia del algoritmo.

```
9  bool elegido(int *eleccion, int n, int esqui){
10   bool result = false;
11
12   for(int i = 0; i < n; i++)
13     if(esqui == eleccion[i])
14       result = true;
15
16   return result;
17 }
18
19 void asignacionEsquies(int **matriz, int n, int m, int *eleccion){
20   int min, elec, tam = 0;
21
22   for(int i = 0; i < n; i++){
23     min = -1;
24     for(int j = 0; j < m; j++){
25       if(min == -1 || min > matriz[j][i]){
26         if(!elegido(eleccion, tam, j)){
27           eleccion[i] = j;
28           tam = i + 1;
29           min = matriz[j][i];
30         }
31       }
32     }
33   }
34 }
```

Como vemos en la anterior captura tenemos 2 funciones, la principal que se llama **asignacionEsquies**() y una auxiliar para ver si un esquí ha sido ya elegido que se llama **elegido**().

El algoritmo consiste en recorrer la matriz por columnas para ir eligiendo el esquí disponible con menos error cuadrático en el momento, para lo que utilizaremos 2 bucles for.

Dentro del segundo bucle for tenemos sentencias de O(1) y una llamada a la función **elegido**(). Esta función **elegido**() también ejecuta un bucle for que se ejecutará 1 + 2 + ... + n-2 + n-1 veces, es decir k(n-1). K dependerá de cómo estén ordenadas las columnas, y de cuantos esquís haya en la solución en el momento, pero a la hora de hacer el límite para calcular la eficiencia del bucle tendremos un O(n). Entonces el algoritmo completo consta de 3 bucles for, dos de ellos tamaño n y otro de tamaño m, por lo que tenemos una eficiencia de $O(m*n^2)$.

6. ¿Es óptimo nuestro algoritmo?

El algoritmo que tenemos siempre encuentra una solución, ¿pero podemos asegurar que ésta es siempre la mejor?

Sea X = $\{x_1, x_2, \dots, x_n\}$ el conjunto de errores cuadráticos que produciría el esquí disponible seleccionado al ser asignado a cada alumno i, donde i va desde 1 hasta n, que tiene el menor error cuadrático medio posible. Sea también Y = $\{y_1, y_2, \dots, y_n\}$ otro conjunto de errores cuadráticos que produciría el esquí disponible seleccionado al ser asignado a cada alumno i. Llamaremos E al error cuadrático medio, entonces, para que nuestro algoritmo sea el óptimo, E_x debe de ser menor o igual que E_y $(E_x \leq E_y)$ para toda elección posible de y_i (sin repetir).

En el ejemplo anterior si nuestro algoritmo da la solución óptima no debería haber ningún otro conjunto de esquís para el que el error cuadrático medio sea menor.

Teníamos que E_x = 17.57.

Si seleccionamos otro conjunto Y = $\{5, 4, 8, 7, 3, 9, 6\}_n$ y calculamos su error tenemos que:

$$E_y = \frac{1}{N} \sum_{i=1}^{N} (H(i) - L(i))^2 = (0 + 9 + 1 + 25 + 4 + 25 + 9)/7 = 73/7 = 10.42 \leq E_x$$

Por lo tanto hemos comprobado que la solución que proporciona no es la mejor, existe una con menor error cuadrático, con lo que podemos concluir que nuestro algoritmo Greedy no es óptimo para solucionar este problema.