


AyED-2018 (<https://campusvirtual.ull.es/1718/course/view.php?id=4608>)/ Práctica #6. Recursividad (<https://campusvirtual.ull.es/1718/mod/assign/view.php?id=205079>)Administración Administración
del cursoNavegación Área personal
(<https://campusvirtual.ull.es/1718/my/>)Inicio del sitio
(<https://campusvirtual.ull.es/1718/?redirect=0>)

Páginas del sitio

Curso actual

AyED-2018
(<https://campusvirtual.ull.es/1718/course/view.php?id=4608>)Participantes
(<https://campusvirtual.ull.es/1718/user/index.php?id=4608>)Tema 0.
INTRODUCCIÓN.
ALGORITMOS Y
ESTRUCTURAS DE...

Tema 1.

Universidad de La Laguna

Pabellón de Gobierno, C/ Padre Hernández, 100 | 38200 | Apartado Postal 456 | San Cristóbal de La Laguna | España | (+34) 922 31 90 00

Tema 2.

ALGORITMOS Y...

MATRICES

Tema 3.

 (<http://www.facebook.com/universidaddelalaguna>)

Práctica #6. Recursividad

El siguiente problema a resolver consiste en generar todas las cadenas de bits de un determinado tamaño. No obstante, se ha adornado con un enfoque aplicado a grafos, aunque esencialmente no deja de ser una aplicación simple.

Considérese el problema de calcular el conjunto de ciclos simples en un grafo dirigido con n vértices de menor coste, para un vector de coste con valores arbitrarios (posiblemente algunos menores que cero). Es decir, calcular todos los posibles grafos con ciclos para n vértices, y de todos ellos tomaremos aquel de menor coste.

Dentro de las muchas maneras que disponemos para representar un grafo, utilizaremos un *vector de incidencia*. Dado un grafo con n vértices esta estructura es una matriz de $n \times n$ elementos, en la que la entrada (i,j) toma el valor 1 si existe un arco desde el vértice i al vértice j . Esta matriz vendrá representada por un vector de enteros de tamaño $n \times n$. Cada posición (i,j) de la matriz corresponde con la posición $i * n + j$ del vector.

El alumno deberá computar todos los vectores posibles de incidencia, comprobar si el vector de incidencia generado representa un grafo válido, y en tal caso evaluarlo. Si el valor del grafo actual es menor al valor del mejor grafo computado hasta el momento, se actualiza el mejor valor.

Un grafo se considerará válido si el vector de incidencia no tiene ningún elemento diagonal, y si a/de cada vértice visitado entra/sale exactamente un único arco. Es decir, la suma de los elementos de cada fila i (el grado de salida del vértice i) debe ser a lo sumo 1, y la suma de los elementos de cada columna debe ser a lo sumo 1 (el grado de entrada del vértice i). Además, si el grado de salida de un vértice es 1, entonces el grado de entrada de ese vértice debe ser también 1.

En resumen, se debe generar todas las cadenas binarias de tamaño $n \times n$. Sólo aquellas que cumplan una determinada condición deben ser evaluadas.

Fase I

Diséñese una clase *minimum_cycles*, *t* que contenga los siguientes atributos:

 (<http://twitter.com/CanalULL>)

- una constante $n_$ entera que almacene el número de vértices del grafo

TIPO
ABSTRACT
O DE
DATOS
LISTA
ENLAZADA

Tema 4.

TIPO
ABSTRACT
O DE
DATOS
LISTA
DOBLEME
NTE E...

Tema 5.

TIPO
ABSTRACT
O DE
DATOS
PILA.

Tema 6.

TIPO
ABSTRACT
O DE
DATOS
COLA.


Tema 7.

RECURSIV
IDAD Y
BACKTRA
CKING

Tema 8.

ALGORITM
OS SOBRE
CONJUNT
OS

Prácticas

 Práctica
#0: Intro
a C++
(<https://campusvirtual.ull.es/1718/mod/assign/view.php?id=162048>)

- un vector *sol_* de enteros que almacene el vector de incidencia computado hasta el momento
- una variable *best_val_* entera que almacene el mejor valor computado hasta el momento

- un vector *best_sol_* de enteros que almacene el mejor vector de incidencia computado hasta el momento

El constructor de la clase debe inicializar *n_* con un tamaño determinado, el vector *sol_* con el tamaño *n_ x n_*, y valor cero, para cada elemento, la variable *best_sol_* con el valor *numeric_limits<int>::max()* (véase biblioteca *limits*), y el vector *best_sol_*, de manera similar a *sol_*.

La clase debe tener además:


- un destructor,
- una función void *write(ostream& os)* que muestre la matriz asociada al vector de incidencia,
- una función void *compute_in_degree(vector<int>& id)* que compute el grado de entrada para el vector *sol_*,
- una función void *compute_out_degree(vector<int>& od)* que compute el grado de salida para el vector *sol_*,
- una función bool *is_cycle(const vector<int>& id, const vector<int>& od)* que determine si el grafo representado en *sol_* contiene un "matching", es decir, que si un vértice es visitado en el grafo, el grado de entrada debe ser 1 y el de salida también.
- una función bool *no_diag(void)* que determine que para la solución almacenada en *sol_* no existe ningún elemento de la diagonal con valor 1,
- una función void *evaluate(const vector<int>& cost)*, que evalúe la solución *sol_* para el vector de costes *cost*, y actualice *best_val_* y *best_sol_* si fuera necesario,


- una función int *pos(int i, int j)* que devuelva la posición del arco (i, j) en el vector de incidencia,


- una función void *generate(int i, const vector<int>& cost)*, que genere todos los posibles vectores de incidencia, y que una vez calculado uno, compruebe si es válido, y en tal caso lo evalúe.

- y una función

```
void compute_best_cycle(const vector<int>& cost)
{
    generate(n_ * n_, cost);
}
```

 Práctica #1: Matrices, operaciones en punto flotante...
(<https://campusvirtual.ull.es/1718/mod/assign/view.php?id=162049>)

 Práctica #1. Asistencia.
(<https://campusvirtual.ull.es/1718/mod/assign/view.php?id=183584>)

 Práctica #2: Triángulo de Pascal y números combinados...
(<https://campusvirtual.ull.es/1718/mod/assign/view.php?id=188707>)

 Práctica

}

Fase II

Evalúese únicamente los ciclos de longitud 2




Evaluación

La calificación de la práctica dependerá de la calidad y eficiencia de la misma.

Si la práctica funciona correctamente, el alumno o alumna efectúa la modificación propuesta por el profesor, y ha concluido la fase I: hasta 7.0

Si además de lo anterior, se ha efectuado la fase II: hasta 10.0


Nota: Téngase en cuenta que se evaluará la capacidad de resolver problemas propuestos por el profesor de prácticas durante la corrección. Estos problemas tendrán una dificultad proporcionada.


-  `minimum_cycles.cpp` (https://campusvirtual.ull.es/1718/pluginfile.php/292762/mod_assign/introattachment/0/minimum_cycles.cpp?forcedownload=1)
-  `prueba.cpp` (https://campusvirtual.ull.es/1718/pluginfile.php/292762/mod_assign/introattachment/0/prueba.cpp?forcedownload=1)
-  `vector.hpp` (https://campusvirtual.ull.es/1718/pluginfile.php/292762/mod_assign/introattachment/0/vector.hpp?forcedownload=1)


Estado de la entrega

Estado de la entrega	No entregado
Estado de la calificación	Sin calificar
Fecha de entrega	domingo, 13 de mayo de 2018, 00:00
Tiempo restante	La Tarea está retrasada por: 5 días

#2.
Asistenci
a.
(<https://campusvirtual.ull.es/1718/mod/assign/view.php?id=189172>)

 Practica
#3.
Matrices
dispersa
s I
(<https://campusvirtual.ull.es/1718/mod/assign/view.php?id=162051>)

 Práctica
#3.
Asistenci
a.
(<https://campusvirtual.ull.es/1718/mod/assign/view.php?id=193165>)

 Práctica
#4.
Matrices
dispersa
s II
(<https://campusvirtual.ull.es/1718/mod/assign/view.php?id=193165>)

Última modificación -

Comentarios de la entrega ▶ Comentarios (0)

Agregar entrega

Realizar cambios en la entrega

