Tecnológico de Monterrey, Campus Monterrey

Departamento de Ciencias Computacionales

**ANALISIS Y DISEÑO DE ALGORITMOS** - Ing. Román Martínez M.

**SEGUNDO EXAMEN DE PROGRAMACIÓN**  - 14 de Noviembre de 2014

# Nombre: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Matrícula: \_\_\_\_\_\_\_\_\_\_\_\_

**Instrucciones:**

* Este es un examen que podrá ser resuelto desde la comodidad de tu casa, accesando el material del curso y cualquier otra referencia que consideres valiosa para que TÚ generes las respuestas a los problemas planteados.
* El examen deberá resolverse en forma INDIVIDUAL. Evita caer en la tentación de conversar con compañeros del grupo y/o ajenos al grupo. Este examen evalúa TUS conocimientos y habilidades y cualquier acción que realices para entregar un conocimiento o habilidad que no es tuyo, es una acción deshonesta que será penalizada fuertemente de acuerdo a nuestro reglamento. También recuerda que “tanto peca el que mata a la vaca como el que le estira la pata”, así que evita hacerte cómplice de algún compañero en afán de ayudarlo o apoyarlo.
* Para resolver este examen puedes utilizar el compilador de C++ de tu preferencia para programar tus respuestas.
* Cada problema del examen deberá programarse en un archivo propio e independiente, respetando los nombres de archivo que se indican en la redacción del problema.
* Este documento deberá imprimirse e identificarse con tu nombre, y en las hojas impresas contestar las preguntas que se indican. Esta impresión ya contestada se entregará en la sesión del martes 18 de noviembre.
* Los archivos con los códigos programados se subirán en la página del curso en el espacio indicado en la sección de exámenes.
* Una vez accesado este examen en la página del curso, contarás con un tiempo máximo de 5 horas continuas para trabajarlo y entregarlo. Considera que los archivos de respuesta deben estar entregados SIN EXCEPCIÓN antes de la sesión del martes 18 de noviembre a las 9 am.
* Para cualquier duda o aclaración del examen, envía un correo electrónico a [roman.martinez@itesm.mx](mailto:roman.martinez@itesm.mx) y se te dará respuesta lo más pronto posible.

***PROBLEMA 1.*** *(30 puntos).*

Como ya lo aprendiste en clase, una cola priorizada es una estructura de datos que se implementa utilizando el concepto de un HEAP, de tal manera que podríamos decir que una cola priorizada y un HEAP son estructuras de datos equivalentes en su nivel lógico o abstracto.

En C++, STL provee al tipo de dato **priority\_queue** para manejar colas priorizadas. En la siguiente liga: <http://comsci.liu.edu/~jrodriguez/cs631sp08/c++priorityqueue.html> encontrarás una explicación breve y concreta de cómo está definido este tipo de dato en STL y algunos ejemplos que ilustran la forma de usarlo (OJO: el primer ejemplo tiene dos errores que al compilar los podrás detectar y corregir).

Implementa el algoritmo del **Heapsort** utilizando a las colas priorizadas de STL. La implementación deberá realizarse como una función libre que recibe un arreglo de números enteros y regresa el mismo arreglo ordenado en forma ascendente. Considera que al implementar el algoritmo utilizarás a una cola priorizada que en realidad es un HEAP, y por lo tanto, NO hay necesidad de implementar las operaciones de un HEAP.

Comprueba el funcionamiento del algoritmo probándolo con algún arreglo de datos y desplegando posteriormente el arreglo ordenado.

Responde:

La implementación realizada del Heapsort ¿mantiene el orden de complejidad de **O(n log n),** lo empeora o lo mejora? Explica brevemente el análisis realizado para justificar tu respuesta.

Aunque el codigo que se implementó aparentemente parece ser O(n) no debemos olvidar que el Priority Queue del STL tiene un heap sort que en realidad es O(n log n), es por eso que el orden sigue siendo O(n log n).

El programa que generes en este problema deberá llamarse P1EX2ADA14.cpp. Documenta el código fuente con los comentarios que consideres convenientes para facilitar la revisión del examen, así como con tus datos personales (matrícula, nombre), y el comentario inicial de si tu programa funciona o no. Sube este archivo en la página del curso.

ver archivo, ya funciona

***PROBLEMA 2.*** *(30 puntos).*

El siguiente programa contiene la implementación del algoritmo de Dijkstra para encontrar el camino más corto del primer nodo vértice hacia todos los demás vértices del grafo.

Copia, ejecuta el programa y analiza el código con la prueba que contiene.

#define tam 5

int datos[tam][tam] = {{0, 999, 4, 5, 999},

{3, 0, 5, 999, 999},

{999, 999, 0, 7, 1},

{999, 6, 999, 0, 9},

{6, 2, 999, 999, 0}};

void camino (int T[], int v)

{ if (T[v] != 1)

{ camino(T, T[v]);

cout << "-" << T[v]; }

}

void dijkstra (int W[tam][tam])

{ int L[tam+1], T[tam+1], R[tam+1], min, vmin;

for (int i = 2; i<=tam; i++)

{ L[i] = W[0][i-1];

T[i] = 1; }

for (int x = 1; x < tam; x++)

{ min = 999;

for (int i = 2; i<= tam; i++)

if ( 0 <= L[i] && L[i] <= min)

{ min = L[i]; vmin = i; }

for (int i=2; i<= tam; i++)

if (L[vmin]+W[vmin-1][i-1] < L[i])

{ L[i] = L[vmin]+W[vmin-1][i-1];

T[i] = vmin; }

R[vmin] = L[vmin];

L[vmin] = -1;

}

cout << "\nCAMINOS MAS CORTOS\n";

for (int i = 2; i <= tam; i++)

{ cout << "de 1 a " << i << ": 1";

camino(T, i);

cout << "-" << i << " cuesta " << R[i] << " unidades\n";

}

}

int main()

{

dijkstra(datos);

system("PAUSE");

return EXIT\_SUCCESS;

}

Utiliza como referencia el programa anterior para implementar un nuevo programa que sirva para encontrar **los caminos con el menor número de escalas posibles del primer nodo vértice hacia todos los demás vértices del grafo**. Una aplicación de este programa podría ser con el grafo de los vuelos posibles de una aerolínea entre ciudades, y la información que arrojaría esta aplicación sería la lista con el mínimo de escalas para viajar de una ciudad a otra. Esto permitiría detectar vuelos directos y vuelos largos por tener muchas escalas. Como podrás observar, para este caso, no se considera optimizar el costo de ir a cada vértice, sino que lo que tiene más valor es pasar por menos nodos vértices para llegar al destino.

Ver codigo, creo que ya funciona

El programa que obtengas deberá tener al menos la misma eficiencia que el programa anteriormente mostrado.

Comprueba el funcionamiento del algoritmo probándolo con el mismo arreglo de datos que se usa en el programa de Dijkstra. Una vez que estés seguro que tu programa funciona, pruébalo con la matriz que se te enviará por mail y en base a tus resultados, responde las siguientes preguntas:

1. ¿Cuáles son los nodos a los que NO se puede llegar desde el nodo 1? 2 y 5
2. ¿Cuál es el número máximo de escalas para llegar desde el nodo 1 a algún otro nodo? \_\_1\_\_\_\_

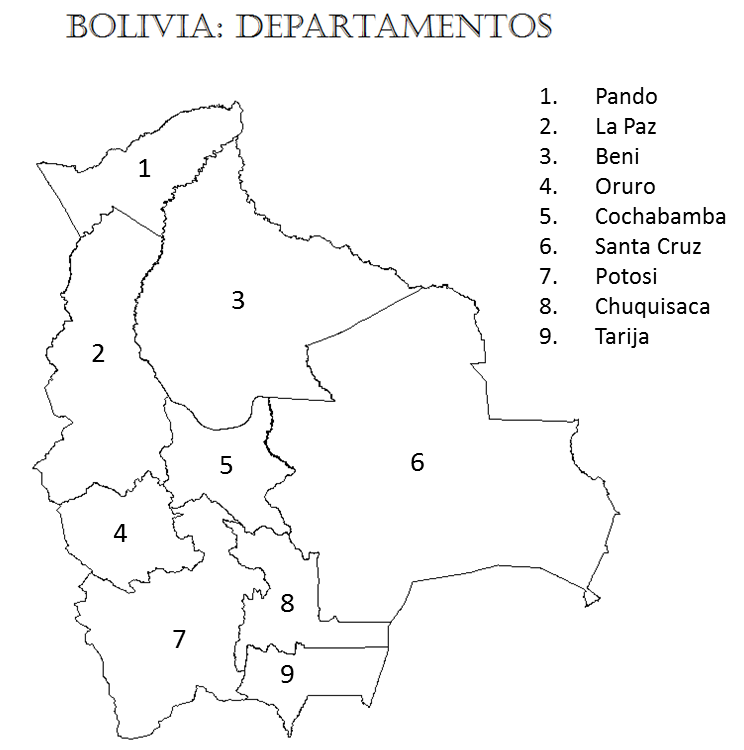
El programa que generes en este problema deberá llamarse P2EX2ADA14.cpp. Documenta el código fuente con los comentarios que consideres convenientes para facilitar la revisión del examen, así como con tus datos personales (matrícula, nombre), y el comentario inicial de si tu programa funciona o no. Sube este archivo en la página del curso.

***PROBLEMA 3.*** *(40 puntos).*

En el archivo P3EX2ADA14.cpp encontrarás un programa que tiene implementado el algoritmo de coloreado de grafos (o mapas) por la técnica de Backtracking. Revisa este código y haciendo una prueba de escritorio o rastreo, entiende cómo funciona el algoritmo sobre el ejemplo que se tiene cargado.

Una vez que hayas entendido bien el funcionamiento, trabaja en las siguientes modificaciones al programa:

1. Cambia en la matriz de adyacencias del grafo los datos para colorear el mapa de Bolivia que se muestra a continuación:



1. Agrega al programa la funcionalidad de contar cuántas son las combinaciones de colores para pintar un mapa en forma válida. Prueba con el mapa de Bolivia y responde:
   1. ¿Cuántas formas existen para pintar el mapa de Bolivia con X colores? 6 formas con 3 colores

*Toma el valor de X de la casilla [0,1] de la matriz de datos que se te envió por mail.*

1. Modifica el programa para que le pregunte al usuario la cantidad de colores con las que quiere colorear el mapa, y también le pregunte el número de color con el que desea pintar el primer estado del mapa. Considerando estos datos de entrada, adapta el programa para que genere como resultado sólo las posibles combinaciones de colores con que se colorea el mapa a partir del color inicial que dio como entrada el usuario. Responde:
   1. ¿Cuántas formas existen para pintar el mapa de Bolivia con X colores si se colorea con el color #N el primer estado? \_\_\_\_\_\_\_\_\_

*Toma el valor de X y de N de la casilla [0,1] de la matriz de datos que se te envió por mail.*

* 1. ¿Cuál es el número mínimo de colores que se necesitan para que se pueda colorear el mapa de Bolivia?\_\_\_\_\_3\_\_\_\_\_

Utiliza el mismo archivo P3EX2ADA14.cpp para hacer las modificaciones y entregarlo. Documenta el código fuente con los comentarios que consideres convenientes para facilitar la revisión del examen, así como con tus datos personales (matrícula, nombre), y el comentario inicial de si tu programa funciona o no. Sube este archivo en la página del curso.