Tecnológico de Monterrey, Campus Monterrey
Departamento de Ciencias Computacionales
ANALISIS Y DISEÑO DE ALGORITMOS - Ing. Román Martínez M.
SEGUNDO EXAMEN DE PROGRAMACIÓN - 18 de abril de 2015

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.
- Los archivos con los códigos programados se subirán en la página del curso en el espacio indicado en la sección de exámenes.
- Adicionalmente, se te pedirán 2 videos que deberás generar como respuesta a este examen y los deberás entregar en la página de Facebook del curso a través de un inbox al profesor. Trata de que estos videos no excedan de 5 minutos de duración.
- NO hay un tiempo específico para contestar el examen. Lo importante es que entregues tus resultados SIN EXCEPCIÓN, con la información que hayas generado como respuesta, a más tardar el LUNES 20 DE ABRIL 2015 a las 8:00 am.
- Como complemento a estas instrucciones, deberás ver el video que el profesor ha colocado en la página de Facebook.
- Para cualquier duda o aclaración del examen, envía un correo electrónico a roman.martinez@itesm.mx y se te dará respuesta lo más pronto posible.

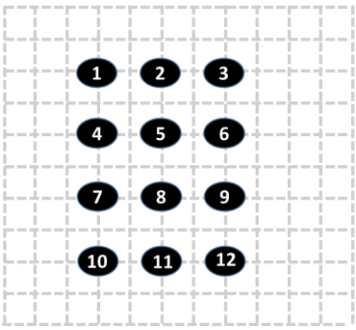
PROBLEMA 1. *(50 puntos).*

Muchos profesores de materias en las que se enseñan conceptos y procedimientos muy específicos, aplican exámenes en los que las respuestas a lo que se pregunta son exactas. Estos profesores buscan generar exámenes diferentes para prevenir la copia de alumnos que están sentados cerca en salones en los que la densidad es alta.

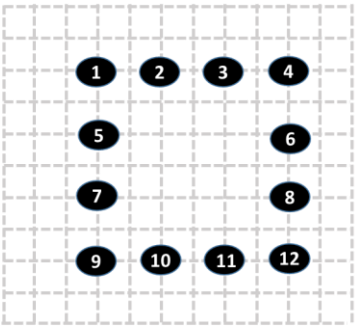
El programa que deberás realizar en este caso, es una aplicación que ayudará a estos profesores a decidir cuántos tipos de exámenes realizar para un determinado grupo, y cómo distribuir los exámenes dependiendo de la configuración de los alumnos en el salón. La regla a cumplir es que no puede haber exámenes del mismo tipo entre alumnos que se encuentren a un metro o menos de distancia en el salón. El programa tendrá precargados los datos de la configuración de los alumnos en el salón, y en base a eso, le solicitará al profesor la cantidad de tipos de exámenes que está dispuesto a hacer. Con este dato, el programa procesará la información e indicará en pantalla si es posible que con esa cantidad de tipos de exámenes se cumpla la regla de distribución que se ha mencionado. Si fuera posible, el programa también desplegará qué tipo de examen se asignará a cada alumno. Los tipos de exámenes se identificarán con las letras mayúsculas a partir de la A, y los alumnos con números enteros a partir del 1.

Para efectos de probar tu programa, dependiendo de tu matrícula, usarás alguna de las siguientes configuraciones de salón para 12 alumnos (cada cuadro del mapa equivale a 0.5 metros):

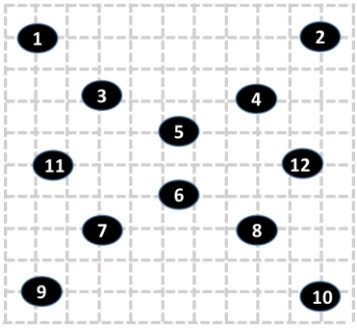
CONFIGURACIÓN H



CONFIGURACIÓN O



CONFIGURACIÓN X



Si tu matrícula termina en 0, 1, 2 ó 3 usarás la configuración H; si termina en 4,5 ó 6 usarás la configuración O y si termina en 7, 8 ó 9 usarás la configuración X.

Para resolver este problema es muy importante que pienses y respondas las siguientes preguntas:

1. ¿Cuál es la técnica de diseño de algoritmos que conviene aplicar para solucionar este problema? ¿qué problema de los expuestos en la técnica te puede servir para solucionar este problema?
2. ¿Cómo se pueden modelar los datos de la configuración del salón para poder cargarlos a memoria en el programa? ¿qué estructura de datos puede representar la información para facilitar el manejo del algoritmo?
3. Dependiendo de la técnica que hayas seleccionado, ¿cuál es el diseño de la solución en términos de la base que requiere la técnica (ej. el árbol de búsqueda de soluciones en Backtracking, o la fórmula recursiva en programación dinámica)?

NO caigas en la tentación de resolver el problema para el caso específico de los datos asignados. Tu programa debe de funcionar para resolver cualquier otro caso de configuración de salón, incluso para salones masivos en donde podría haber cientos de alumnos. Para eso, respeta la estrategia de la técnica de diseño de algoritmos que elijas. Si el tiempo te lo permite, puedes probar con las otras configuraciones aunque no se te hayan asignado.

Como resultado de este problema deberás entregar lo siguiente:

1. El código con la solución implementada y que deberá llamarse P1EX2ADAEM15.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.
2. Un video en el que menciones lo siguiente (trata de que el video no exceda de 5 minutos de duración):
    a. Cómo fue tu proceso de solución al caso, respondiendo a las preguntas de estrategia de solución que se mencionaron anteriormente.
    b. Si tu programa funciona correctamente y cuál es la solución que da el programa a la configuración específica de salón que se te asignó. Para esto, muestra la pantalla de tu programa en ejecución, y los segmentos de código que consideres importantes explicar.
    c. Cuánto tiempo tardaste en solucionar el caso y cualquier otro comentario que consideres importante considerar para la revisión.

PROBLEMA 2. *(50 puntos).*

El problema del árbol de extensión mínima, es un caso que viene de la historia previa a la computación. En 1926, cuando aún no existían las computadoras, Otakar Borůvka propuso un algoritmo para construir la red de electricidad en la ciudad de Moravia (hoy Republica Checa) en forma eficiente. En honor a él, este algoritmo lleva su nombre. Posteriormente, con la llegada de las computadoras, este algoritmo fue tomado como base para su implementación computacional y en honor a quien lo trabajó en los años 60's, también se le conoce como el algoritmo de Sollin. Este algoritmo es muy fácil de paralelizar, por lo que es muy utilizado en computadoras de múltiples procesadores.

El algoritmo de Boruvka o Sollin es un algoritmo voraz que algunos autores consideran que es una mezcla de la estrategia de los algoritmos de Kruskal y Prim. Para entender cómo funciona este algoritmo, puedes consultar múltiples referencias que hay en Internet, incluyendo videos y animaciones gráficas. Esta referencia es un muy buen resumen que tomaremos como base para trabajar en este problema: http://en.algoritmy.net/article/43803/Boruvkas-algorithm. Date tiempo para asegurarte de entender muy bien la filosofía de este algoritmo, pues la tarea en este caso será hacer la implementación del mismo.

El programa a realizar deberá tener cargada en forma constante la matriz de adyacencias de un grafo tal como se muestra en el anexo. El programa desplegará en pantalla cuáles son los arcos que corresponden al árbol de extensión mínima, el valor de cada arco y además indicará el costo total mínimo de la conexión lograda con el árbol. Los nodos del grafo estarán identificados con los valores índices de la matriz (del 0 en adelante) y esta identificación se usará para describir a los arcos en el formato X-Y donde X es el nodo origen y Y el nodo destino.

El algoritmo de Boruvka que implementarás deberá estar basado en el pseudocódigo que presenta la referencia indicada y que es el siguiente:

```
01.    /**
02.    * Boruvka's algorithm
03.    *
04.    * @param graph graph (each edge has a different weight)
05.    * @param weight weights of all the edges
06.    * @return minimal spanning tree
07.    */
08.    boruvkaAlgorithm(graph, weight)
09.      SpanningTree tree //spanning tree
10.      components = graph.vertices //at the beginning every vertex is a component
11.      while components.size > 1 do
12.         for each component in components
13.            edge e = connectWithMinimalEdge(component, weight) //connect with some other component
                                                                    using a minimal edge
14.            tree.add(e) //add the edge into the spanning tree
15.      return tree
```

Como podrás observar, este pseudocódigo de alto nivel está usando algunos objetos como: `tree`, `components` y `edge` que será importante que decidas cómo implementarlos y representarlos en memoria. También será relevante analizar que debe de hacer la función `connectWithMinimalEdge` e implementarla.
En tu implementación puedes cambiar la parametrización que este pseudocódigo para que recibas al grafo en la propia matriz de adyacencias. También hay libertad de utilizar cualquier objeto y algoritmo que provee STL y que consideres útiles y necesarios. Si consideras que un HEAP te puede servir en este caso, utiliza la cola priorizada de STL que manejaste en la tarea #13, pero NO es obligatorio usar HEAPs.

Como resultado de este problema deberás entregar lo siguiente:

1. El código con la solución implementada y que deberá llamarse P2EX2ADAEM15.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. Incluye también en esta documentación la lista de referencias en Internet que consultaste. Sube este archivo en la página del curso.
2. Un video en el que menciones lo siguiente (trata de que el video no exceda de 5 minutos de duración):
    a. Cómo fue tu proceso de solución al caso. Explica cómo están implementados los tipos de datos que permiten manejar a los arcos, los componentes y al árbol de extensión mínima. También comenta cómo implementaste la función `connectWithMinimalEdge`, mostrando en pantalla el código implementado.
    b. Si tu programa funciona correctamente y cuál es la solución que da el programa al grafo que se utilizó para la prueba. Para esto, muestra la pantalla de tu programa en ejecución.
    c. Tu reflexión respecto a la eficiencia de este algoritmo y si puede considerarse mejor a los algoritmos de Prim y Kruskal.
    d. Cuánto tiempo tardaste en solucionar el caso y cualquier otro comentario que consideres importante considerar para la revisión.

ANEXO – Matriz de adyacencias del grafo para la prueba del problema 2.

```
int grafo[][] = {{0,5,3,999,999,4,5,4,999,999,7,8,999,5,6},
                 {2,0,999,999,999,6,999,4,6,999,999,9,3,4,1},
                 {5,2,0,1,999,6,999,999,8,1,999,4,2,999,999},
                 {2,8,999,0,1,6,999,3,3,999,999,999,7,999,999},
                 {999,999,6,999,0,999,1,7,1,999,3,1,999,1,999},
                 {999,5,3,999,999,0,999,8,999,999,999,999,2,7,4},
                 {6,999,999,1,999,999,0,4,999,4,9,9,999,1,999},
                 {2,999,5,8,4,7,999,0,999,999,999,999,1,4,1},
                 {9,999,6,9,4,1,1,2,0,9,999,999,3,999,4},
                 {999,999,6,3,999,7,6,999,9,0,999,999,999,999,2},
                 {3,8,999,999,999,6,7,4,1,4,0,5,999,999,999},
                 {2,2,9,8,1,999,4,999,2,999,4,0,999,2,999},
                 {999,7,999,1,999,7,999,1,6,999,2,7,0,8,999},
                 {8,7,7,4,4,9,999,6,6,6,999,999,7,0,999},
                 {999,999,999,999,999,4,5,999,5,4,4,999,4,8,0}};
```