PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
FACULTAD DE CIENCIAS E INGENIERÍA

Algoritmia

Laboratorio 1

Semestre 2014-1

Objetivos

1.- El alumno debe ser capaz de elaborar un programa empleando la estrategia conocida como "fuerza bruta" para solucionar un problema. Además debe verificar mediante experimento que el tiempo de ejecución crece de forma exponencial a medida que el tamaño de la entrada se incrementa.

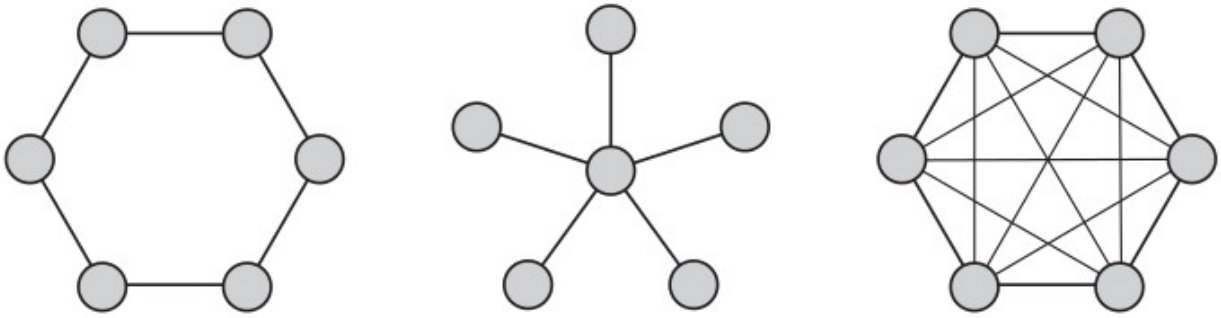
2.- El alumno debe ser capaz de elaborar un programa empleando la estrategia conocida como "búsqueda exhaustiva" para solucionar un problema. Además debe verificar mediante experimento que el tiempo de ejecución decrece cuando se conocen más restricciones.

3.- El alumno debe ser capaz de elaborar un programa empleando recursividad. Además debe deducir la taza de crecimiento mediante experimento.

Problemas

Archivo a entregar *topologia.c*

1.- **(3+2+3+1= 9 puntos) Topologías de redes**. Una red topológica especifica cómo las computadoras, impresoras, y otros dispositivos están conectados sobre una red. La figura de abajo ilustra topologías comunes de redes: anillo, estrella y completamente conectado.



Anillo Estrella Completamente conectado

A usted se le proporcionará una matriz booleana $M[0..n\text{-}1, 0..n\text{-}1]$, donde $3 < n \leq 1000$, la misma que representa la matriz de adyacencia modelando una red con una de estas topologías. Su tarea es determinar cuál de estas tres topologías, si hay alguna, representa la matriz. Elabore un programa en C, empleando la estrategia de "fuerza bruta" para esta tarea y mediante experimento determine la taza de crecimiento de este programa. El resultado del análisis así como su conclusión deben ser colocados en el mismo archivo del programa *topologia.c*, como comentario.

Archivo a entregar *fibonacci.c*

2.- **(3+1= 4 puntos) Fibonacci**. A continuación se muestra tres algoritmos que calculan el famoso número de Fibonacci.

```
long unsigned fibo1(long unsigned n) {
      long unsigned x,k;
      long unsigned y=1,t;

      x=k=0;
      while(k != n) {
            t = x;
            x = y;
            y = t + y;
            k++;
      }
      return x;
}

long unsigned fibo2(long unsigned n){
      if(n == 0) return 0;
      if(n == 1) return 1;
      if (n > 1) return (fibo2(n - 1) + fibo2(n - 2));
}

long unsigned  fibo3(long unsigned n){
      long unsigned  a,x,b,y,t,k;

      a=x=0;b=y=1;k=n;
      while(k != 0)
            if((k % 2) == 0) {
                  t = a;
                  a = a*a + b*b;
                  b = 2*t*b + b*b;
                  k = k / 2;
            } else {
                  t = x;
                  x = a*x + b*y;
                  y = b*t + a*y + b*y;
                  k = k - 1;
                }
        return x;
}
```

Modifique todas las funciones *fibo?*, de forma que la función tenga el siguiente prototipo: *fibo? (long unsigned n, long unsigned *v)* donde la función sigue devolviendo el número Fibonacci de *n*, pero además la variable *v* debe devolver el número de iteraciones o invocaciones recursivas, según sea el caso, que llevó a cabo para calcular dicho número. Además ordene los algoritmos por el número de iteraciones o invocaciones recursivas según el resultado obtenido. Para lograr este último objetivo proporcione diferentes números a cada uno de los algoritmos. Usted debe de presentar una tabla de doble entrada, como comentario en el archivo *fibonacci.c*, tal como se muestra a continuación

	4	8	12	16	20	28	32	40	48	60	64
fiboX											
fiboY											
fiboZ											

Para facilitar el trabajo se le ha proporcionado el archivo fuente conteniendo las tres funciones.

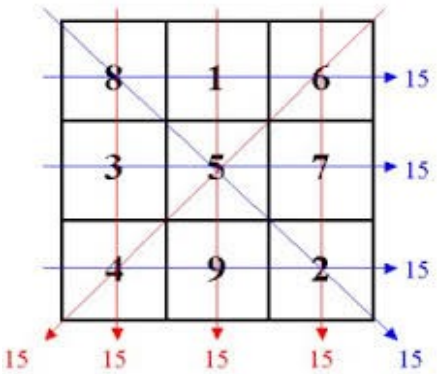
Archivos a entregar *cuadmagic1.c* y *cuadmagic2.c*

3.- **(4+3 = 7 puntos) Cuadrado mágico de 3x3.** Este problema consiste en llenar una tabla de 3x3 con nueve enteros positivos desde 1 hasta 9, de forma que la suma de los números en cada fila, columna y diagonales principales sea la misma. ¿Cuántas formas hay de llenar dicha tabla? Pensemos en llenar la tabla con un número a la vez, iniciamos colocando el 1 en cualquier parte y terminando con el 9. Hay 9 formas de colocar el 1, seguido de 8 formas de colocar el 2, y así hasta el último que es el 9, que es colocada en la última celda desocupada de la tabla. De aquí hay 9! = 9x8x7x...x1 = 362 880 formas de organizar los nueve números en las celdas de una tabla de 3x3.

Por tanto, resolver este problema por búsqueda exhaustiva debería implicar generar todas las 362 880 posibles distribuciones de los distintos enteros de 1 hasta 9 en la tabla y luego verificar, por cada distribución, si todas sus filas, columnas, y diagonales suman lo mismo. Elabore un programa en C, que siguiendo las indicaciones anteriores imprima todos los cuadrados mágicos de 3x3. Su programa también deberá imprimir el número de iteraciones y estas deben ser igual a 362 880.

0 8	1 3	2 4
3 1	4 5	5 9
6 6	7 7	8 2

Actualmente, no es difícil resolver este acertijo probando primero que el valor de la suma en común es igual a 15 y que 5 debe ser colocado en la celda central de la tabla.



Elabore un programa en C, (copie el programa anterior, cámbiele de nombre para modificarlo) que siga las últimas indicaciones. El programa deberá imprimir todas los cuadrados mágicos de 3x3 y debe de imprimir que tan rápido (en porcentaje) es respecto al primero.

Pando, 01 de abril de 2014.

Prof. Alejandro T. Bello Ruiz