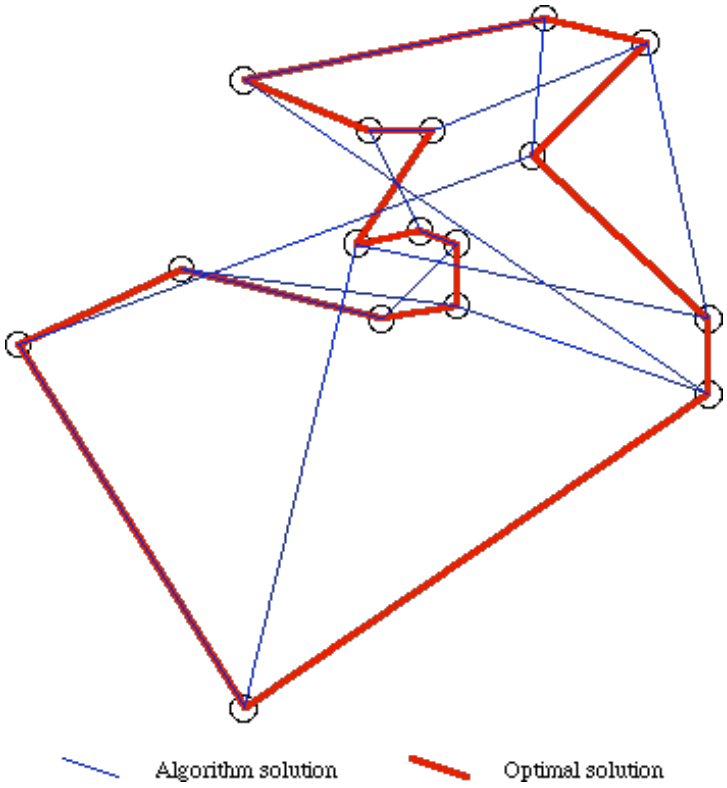
Proyectos:

1.- Polinomios

- Escribir un programa en C++ que pueda manejar polinomios de una variable
- Los exponentes de p(x) son enteros
- Los valores de x son numeros reales
- El programa (la clase) debe ser capaz de ejecutar las siguientes tareas
    - leer el polinomio de teclado y archivo
    - escribir el polinomio a pantalla y archivo
    - evaluar el polinomio *p(a)* para un valor *a* dado
    - borrar el polinomio liberando la memoria usada
    - dados 2 polinomios regresar su suma
    - dados 2 polinomios regresar su resta
    - dados 2 polinomios regresar su multiplicacion
- Se deberá implememententar con cualquier tipo de listas ligadas

2.- El Agente Viajero

El problema del agente viajero, trabajo de investigación http://es.wikipedia.org/wiki/Problema_del_viajante.
Implementar algún metodo que da una “buena” solución a el problema del agente viajero para 70 estados (puntos 2D). El costo de ir de un punto a otro es su distancia Euclidiana.



3.- El tour del Caballo

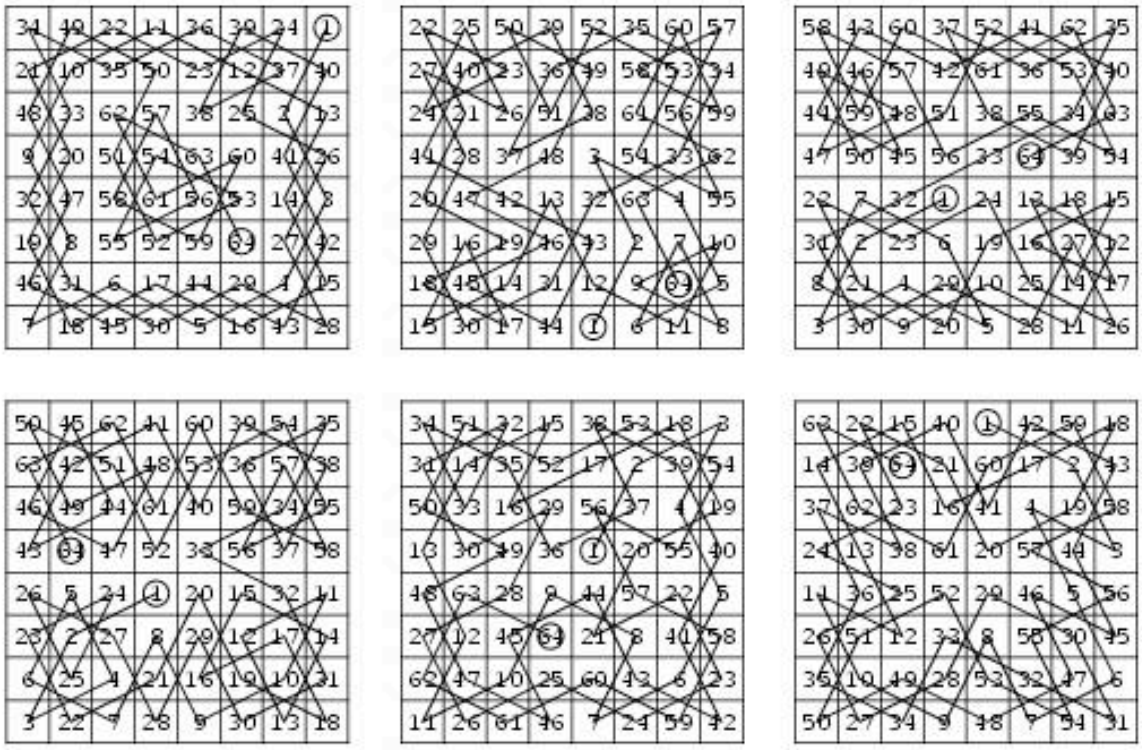
El tour del caballo (*A knight's tour*) es una secuencia de movimientos que hace un caballo de ajedrez (el cual se puede mover desplazandose una casilla horizontal y 2 verticales, o bien una casilla vertical y 2 horizontales), de tal forma que cada casilla del tablero es visitada una sola vez.

Escribir un programa que muestre un tour del caballo.

Tips

- Se puede usar usar una pila
- Se debería de aprender a hacer backtracking

Ejemplos de Tour



examples of the tour

4.- Diccionario

Escribir una programa que abra un archivo de texto y ejecute un chequeo de ortografía basado en un diccionario que se lee de otro archivo (este diccionario no es más que una lista de las palabras válidas, una por linea).

El programa debe de mandar a pantalla todas las palabras del archivo de texto y ponerles un asterisco a la izquierda a todas las palabras mal escritas. Por ejemplo: “Esto es una *emerjencia. Vendo *esterio de *manufaktura japonesa”

Se debe de usar alguno de los mecanismos de búsqueda basados en estructuras de datos de los vistos en el curso.

El que logre tener el menor tiempo en la revisión de un archivo de prueba grande (que yo coloco) puede tener puntos extra.

5.- Grandes números

Lo números enteros en C++ están limitados a números de 9 dígitos mas o menos. Para algunas aplicaciones como criptografía se necesita manejar números muy largos, de cientos de dígitos. Una manera de representar estos numeros es con listas ligadas en alguna base. En este proyecto usaremos la base 2^{32} , de tal forma que cada "dígito" puede estar entre 0 y 2^{32}-1.

Vamos a escribir los números en el siguiente orden, el "digito" en la cabeza de la lista es el menos significativo, y el dígito en la cola es el más significativo.
Para tener una representación única de cada número, el "digito" mas significativo no puede ser cero, por lo tanto, el número cero sera representado con una lista vacia.

No manejaremos números negativos.

El número que representa la lista *p* denotado como *f(p)* está dado por

p->digit + 2^{32}*p->next->digit + 2^{64} * p->next->next->digit + ...

o bien

$$f(p) = \begin{cases} 0 & \text{, si } p \text{ es vacia} \\ p\text{->digit} + 2^{32} \times f(p\text{->next}) & \text{, de lo contrario.} \end{cases}$$

Implementar la clase C++ que puede manejar estos números y que ademas soporta las siguientes operaciones:

```
void addUnsignedInt(unsigned int digit, BigNumNode *operando);
```

la cual suma un numero guardado en digit que está entre [0, 2^{32}-1]. Y

```
void addt(BigNumNode *operandoResult, BigNumNode *operando);
```

que toma 2 listas ligadas (2 bigNumbers) y regresa el resultado de la suma de ellos en el primero.

6.- Bancos

Simular las colas de espera de un banco. El número de colas “normales” es un parámetro de usuario, y hay una cola para clientes “preferenciales” (por supuesto, cuando no hay nadie en las colas “normales”, se atienden en estas a los clientes “preferenciales” para que esten menos tiempo esperando).

Cada cuanto tiempo llega un cliente, si es de tipo “normal” o “preferencial”, y cuantos minutos tarda su trámite (una vez que llega a la ventanilla) se decide de manera aleatoria.

El objetivo del programa es que dado el número de colas y las probabilidades de llegada de clientes, se debe de dejar corriendo el programa simulando, digamos, 5 horas (un periodo de trabajo).

Al final de la simulación se debe de reportar información del tiempo que pasaron los clientes dentro del banco: el promedio, el mejor y el peor, cuantas personas se quedaron encoladas al final del periodo de trabajo (cuando el banco cierra las puertas). Todo lo anterior por categorias de clientes, para los “normales y para los preferenciales”.