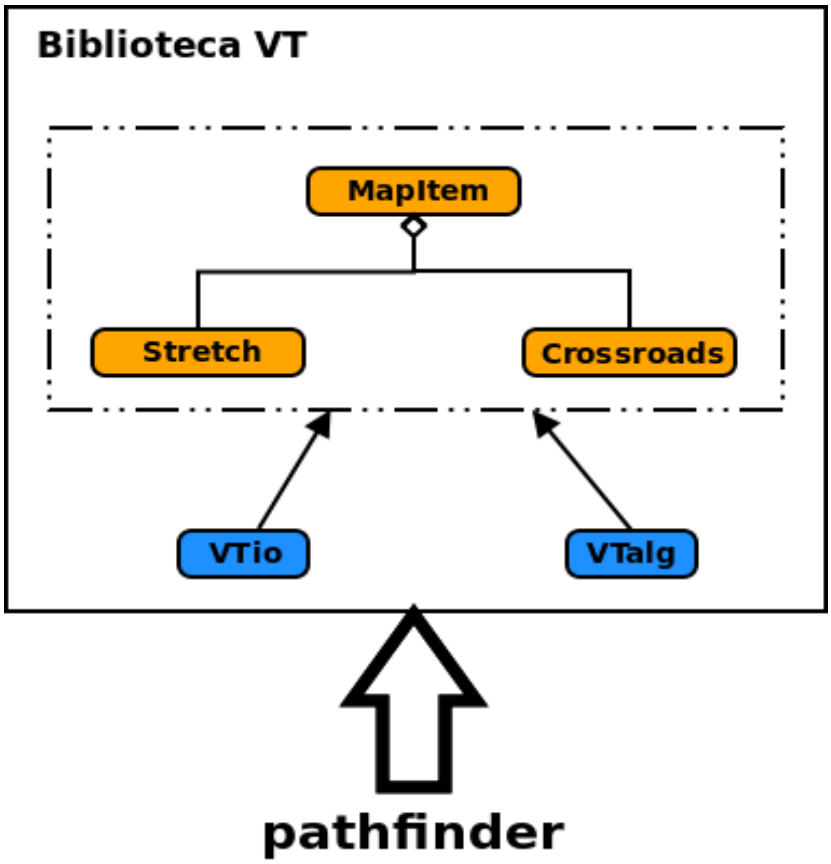
Práctica 1: El Turista Virtual

Rubén Dugo Martín
Modelos de Inteligencia Artificial
Curso 2009-2010
Universidad de Granada

Para la implementación de la práctica he optado por un diseño totalmente estructurado y modularizado, haciendo uso de las posibilidades que nos ofrece *C++*; clases, contenedores STL, herencia, etc.

Este diagrama describe gráficamente las diferentes estructura de datos y paquetes desarrollados para la práctica:



Explicaré brevemente la estructura de la biblioteca VT (de Virtual Tourist) que he desarrollado para práctica:

- ***MapItem***: Encapsula un elemento cualquiera del mapa, contiene un número y una cadena de identificación. Ficheros *MapItem.hh* y *MapItem.cc*. De esta clase heredarán varias.
- ***Crossroads***: Encapsula un cruce del mapa, contiene un par de coordenadas. Ficheros *Crossroads.hh* y *Crossroads.cc*.
- ***Stretch***: Encapsula un tramo del mapa, contiene los cruces entre los cuales está el tramo y el nombre de la calle a la que pertenece. Ficheros *Stretch.hh* y *Stretch.cc*.
- ***VTio***: Se trata de un paquete que contiene todas las funciones de entrada y salida del turista virtual. Ficheros *VTio.hh* y *VTio.cc*.
- ***VTalg***: Es el verdadero corazón de la práctica, contiene los algoritmos *BPP*, *BPA*, *BMP* y *BPM* implementados. Ficheros *VTalg.hh* y *VTalg.cc*.

Todas las funciones y estructuras de datos están comentadas en sus ficheros de cabecera correspondientes. Así como las implementaciones más importantes.

El único fichero relevante para la corrección de la práctica es el *VTalg.cc* en el cual se encuentran implementados los algoritmos de búsqueda en grafos.

También cabe destacar la estructura de datos que he implementado para realizar los recorridos en los grafos, se trata de una encapsulación genérica (plantilla) de los grafos en *C++*, sus ficheros fuente son *CGraph.hh* y *CGraph.cc*.

Aunque he implementado varios algoritmos de búsqueda en grafos (*BPP*, *BPA*, *BMP* y *BPM*) me he quedado con el *BPM* (Búsqueda Primero el Mejor, concretamente la variante *A**) ya necesita expander menor número de nodos para llegar a una solución, además esta siempre es óptima.

En esta tabla plasmo los resultados obtenidos para los diferentes problemas, todos realizados con el parámetro *W=0.5*.

Problema	**Nodos expandidos**	**Longitud del camino (m)**
1	*24*	*717.33*
2	*25*	*532451*
3	*12*	*210.814*
4	*10*	*588.95*
5	*33*	*1050.65*
6	*48*	*585.626*
7	*71*	*725.935*
8	*34*	*767.358*
9	*130*	*1621.89*
10	*138*	*1497.75*
11	*494*	*4654.35*
12	*713*	*4562.03*
13	*441*	*4397.16*

Como comentario final cabe destacar que el proyecto se entrega sin ningún fichero objeto compilado. Se puede compilar y enlazar todo el proyecto con la orden *make*, aunque ya incluyo un ejecutable en el directorio */bin* compilado para *x86* por si algo falla.