1. undirected (blind) brute-force approaches
   * **breadth-first search**
   * **depth-first search**
   * **ID-DFS search**
   * 广度优先搜索
   * 深度优先搜索
   * ID-DFS 搜索
2. Heuristic Approaches
   * **best-first search**
   * **A\* search**
   * 最佳优先搜索
   * A\* 搜索

**Programming**details:  
• You can generate your program in C, C++, Python, C#, or Java. (If you want to use some  
other language I’m willing to discuss it but you’ll need to come to me in advance.)  
• You’re given 2 data files:  
◦ The first is a list of all the cities we know about—mostly small towns in southern Kansas —and the latitude and longitude of each.◦ 第一部分是我们所知道的所有城市--大多是堪萨斯州南部的小镇--以及每个城市的经纬度。

# Names have been tweaked so that city names consisting of more than one word have an  
underscore rather than a space between the words (South\_Haven rather than South Haven), to simplify input. # 名称已作调整，因此由多个单词组成的城市名称之间将使用下划线而不是空格（South\_Haven 而不是 South Haven）。以简化输入。  
◦ A file (CSV) listing each town (pair) as a related adjacent node.

一个文件（CSV），列出了作为相关相邻节点的每个城镇（对）。  
# be aware adjacency is symmetric: If A is adjacent to B, then B is adjacent to A. This may not be listed comprehensively if your method requires that bidirectional connections be explicitly stated, you may need to generate additional pairs for the symmetrical connection to work. {That is, tell the program that it’s possible to go from listed A as adjacent to B or listed B as adjacent to A.}

注意邻接是对称的：如果 A 与 B 相邻，则 B 与 A 相邻。如果您的方法要求明确说明双向连接，则可能无法全面列出这一点，您可能需要生成更多的连接对，以实现对称连接。

也就是说，告诉程序可以将 A 列为与 B 相邻，或者将 B 列为与 A 相邻。  
--> Be sure to take this into account when setting up your program’s data structures. If adjacency is listed in either direction, it should be considered present in both directions.

在设置程序的数据结构时，请务必考虑到这一点。如果邻接关系在任一方向上都被列出，则应将其视为在两个方向上都存在。

• Your program should:

* Ask the user for their starting and ending towns, making sure they’re both towns in the database.
* - 询问用户的起点和终点城镇，确保这两个城镇都在数据库中。
* Ask them then to select the search method they wish to use to find a route to the destination.- 然后请他们选择希望使用的搜索方法，以找到前往目的地的路线。
  + If that route exists, the program should then print the route the method found in order, from origin to destination.- 如果该路线存在，程序就会按从起点到终点的顺序打印出该方法找到的路线。
  + *If you want to get fancy, you might see how the route generated looks as a map (either a map of connected states as we looked at in class, or as a projection in 2D space based on location and connectivity).- 如果你想花点心思，可以看看生成的路线在地图上是什么样子的（可以是我们在课堂上看到的连通状态的地图，也可以是根据位置和连通性在二维空间中的投影）。*

*Note that your database is very limited, and many of the real-world roads are left out for simplicity\*.*

* Your program should also:
  + measure and print the total time needed to find the route (and include a time-out).- 测量并打印寻找路线所需的总时间（包括超时时间）。
  + calculate and display the total distance (node to node) for the cities visited on the route selected.- 计算并显示所选路线上访问城市的总距离（节点到节点）。
  + (opt) determine the total memory used (scale of the arrays) to find the solution.- 选择）确定求解所使用的总内存（数组的规模）。
* Return to the search method selection and allow a new method to be selected for comparison. - 返回搜索方法选择，允许选择新方法进行比较。

*\*A real mapping application gives directions using much more complicated metrics, including variable road conditions, construction closure, fuel economy, and toll costs.*