**阶段一：核心特征粗聚类**

此阶段的目标是快速识别并分组高度相似的线路，以减少后续计算量。

**1. 数据预处理与特征提取：**

* **标准化POI与城市信息：** 将线路poi列表中的POI和途径国家、省份、城市标准化。例如，将POI名称匹配到唯一的POI ID，将城市名称统一。
* **提取关键序列：**
  + **每日城市停留序列：** 提取每天的城市序列，例如：[扬州市, 扬州市], [南京市]。
  + **POI序列：** 提取每日的POI访问序列，例如：[大明寺, 瘦西湖], [...]。
* **提取数值特征：** 天数。
* **提取文本特征：** 线路标题、行程特色/标签（如果可用）。

**2. 核心相似度计算：**

* **城市序列相似度：** 采用**动态时间规整 (DTW)** 衡量每日城市停留序列的相似度。DTW能有效处理序列长度不一致的情况，容忍局部差异。
* **POI序列相似度：** 同样采用**DTW**来衡量POI序列的相似度。为了提高DTW的效率，可以考虑：
  + 使用**POI类别或预训练的POI嵌入**作为DTW的局部距离度量，而非直接使用经纬度（经纬度在DTW中可能过于敏感）。
* **天数相似度：** 简单计算天数的绝对差值。
* **文本相似度：** 对线路标题等文本，使用**TF-IDF + 余弦相似度**或**词嵌入（如Word2Vec/GloVe）+ 余弦相似度**计算。

**3. 多维度相似度融合：**

* 将上述各项相似度通过**加权求和**或**机器学习模型（如决策树或逻辑回归）进行融合，得到一个综合的粗粒度相似度得分**。权重可以根据业务重要性或通过小规模标注数据进行学习。

**4. 粗聚类：**

* **分层聚类（Hierarchical Clustering）：** 使用综合相似度得分进行**凝聚式分层聚类**。这会生成一个聚类树（dendrogram）。
* **阈值剪枝：** 设置一个相对宽松的相似度阈值，剪枝聚类树，形成初步的**粗粒度簇**。这些簇内的线路彼此之间有较高的相似性，但可能仍包含一些非重复线路，或每个簇的线路数量可能偏多。

**优劣势：**

* **优势：**
  + **直观且易于理解：** 基于核心特征的相似度计算逻辑清晰。
  + **高效降维：** 初步聚类能有效减少后续阶段的计算量。
* **劣势：**
  + **DTW计算成本：** 尽管是粗聚类，但DTW对于所有线路对的计算量仍然较大。
  + **相似度融合权重确定：** 手动调整权重可能需要多次尝试。

**阶段二：高级嵌入细聚类与约束处理**

此阶段对阶段一产生的每个粗粒度簇进行精细化处理，利用高级嵌入技术进行更准确的相似度评估，并处理“类内属性之和”的约束。

**1. 线路嵌入（针对每个粗粒度簇内线路）：**

* 在粗粒度簇内部，为每条线路生成一个**线路嵌入向量**。由于你暂时不考虑多模态，我们可以专注于结构化数据和文本的融合。
* **POI序列嵌入：**
  + 使用**Word2Vec/Doc2Vec**（将POI视为“单词”，序列视为“文档”）或**基于Transformer的编码器**（如MiniBERT或T5的编码器部分，对POI ID序列进行训练）来生成POI序列的嵌入。这能更好地捕捉POI之间的顺序和上下文关系。
* **城市序列嵌入：**
  + 类似POI序列，使用**Word2Vec**或**Transformer编码器**生成城市序列的嵌入。
* **数值特征嵌入：** 天数等数值特征可以进行归一化后，通过一个小型全连接网络转换为嵌入向量。
* **文本嵌入：** 对线路标题等文本，使用\*\*预训练的语言模型（如Sentence-BERT或其轻量级版本）\*\*生成高质量的文本嵌入。
* **线路综合嵌入：** 将上述所有嵌入向量（POI序列嵌入、城市序列嵌入、数值特征嵌入、文本嵌入）进行**拼接（Concatenation）**，形成一个高维的**线路综合嵌入向量**。

**2. 细粒度相似度计算：**

* 在每个粗粒度簇内部，计算线路综合嵌入向量之间的**余弦相似度**。

**3. 精细聚类与约束处理：**

这是实现“基于空间聚类的同时，把每个类内节点某个属性之和控制在设定的范围内”的关键。

* **LSH 加速：** 对于每个粗粒度簇，如果其内部线路数量仍然很大，可以使用\*\*局部敏感哈希（LSH）\*\*对线路综合嵌入向量进行分桶，进一步缩小需要两两比较的范围，加速相似度查找。
* **定制化聚类算法（重点）：**
  + **约束K-Means/K-Medoids 变体：** 对每个粗粒度簇，运行一个修改版的K-Means或K-Medoids。
    - 在**簇分配步骤**中，当一个线路点被尝试分配给一个簇时，除了考虑与簇中心的距离（相似度），还需要检查：
      * **“类内节点某个属性之和”约束：** 如果将该线路加入簇会导致该簇的某个属性总和（例如，假设每条线路有一个“复杂性指数”或“POI总数权重”，聚类希望这些之和不超过预设值）超过预设的**上限**，则不允许将其分配到该簇，或者寻找次优的选择。
      * **“类内节点某个属性之和”下限（可选）：** 如果存在，确保每个簇的总和不低于某个**下限**。
    - **启发式搜索：** 由于严格满足约束的K-Means可能难以收敛，可以采用启发式方法：例如，首先进行无约束的K-Means，然后识别违反约束的簇，并尝试分裂或调整其成员。
  + **两阶段后处理优化：**
    - **第一阶段：** 在每个粗粒度簇内部，使用**DBSCAN**或**Agglomerative Clustering**（基于线路嵌入的余弦相似度）进行**无约束的细粒度聚类**。这会生成一系列“潜在重复”的簇。
    - **第二阶段（核心）：** 对每个生成的细粒度簇进行**约束检查和调整**：
      * **超限处理：** 如果某个簇的“属性之和”超过了设定的上限，则需要**分裂**该簇。分裂策略可以是：移除簇内与中心最不相似的线路，或将整个超限簇重新进行更严格的聚类（例如，通过递归调用K-Means或层次聚类，直到满足约束）。
      * **欠限处理（如果存在）：** 如果某个簇的“属性之和”低于设定的下限，可以尝试将其**合并**到最近的、且合并后仍满足约束的其他簇中。
      * **优先选择：** 在选择分裂或合并的线路时，可以优先处理那些“边缘”或“贡献度最小/最大”的线路。

**4. 去重策略：**

* 在每个最终的细粒度聚类内部，选择一条“代表性”线路作为保留线路。选择标准可以包括：
  + **信息完整度最高：** 拥有最详细的配图、POI拓展信息等。
  + **标题/描述最具代表性：** 语义上最能代表该类线路。
  + **中心点线路：** 聚类中心附近的线路。

**优劣势：**

* **优势：**
  + **精度高：** 线路嵌入能更准确地捕捉线路的复杂语义和结构。
  + **处理约束：** 引入了明确的策略来处理“属性之和”的约束，尽管实现复杂。
  + **可伸缩性：** 阶段性处理和LSH的应用提升了大规模数据处理能力。
* **劣势：**
  + **实现复杂：** 涉及多种嵌入技术和定制化的聚类算法，开发难度较大。
  + **计算资源：** 深度学习模型的训练和推理需要一定的计算资源。
  + **超参数调优：** 嵌入模型、聚类算法和约束阈值的超参数调优是挑战。