问题描述：将27个需要配送的医院仓库划分成5组，分别由一辆货车负责配送，

找出配送方案的最优划分，并模拟该方案进行每天的模拟配送，计算出一个月的总里程

1. 首先将27个点划分成5组属于一个集合划分问题，但是27个点的划分复杂度太高，

我们把没3个距离最近的点看作一个元素，这是就是把9个点划分成5组，然后得出所有可能的划分方案，其中集合的划分核心算法如下：（采用递归算法和动态规划）

/// <summary>

/// 根据集合产生groupCount个真子集的划分

/// 动态规划缓存结果可提高效率

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="set"></param>

/// <param name="groupCount"></param>

/// <returns></returns>

public static List<List<List<T>>> SetPartition<T>(List<T> set, int groupCount)

{

int count = set.Count();

var cache = MemoryCache.Default;

CacheItemPolicy policy = new CacheItemPolicy()

{

AbsoluteExpiration = new DateTimeOffset(DateTime.Now.AddSeconds(10))

};

var key = $"Partition-{count}-{groupCount}";

var value = cache[key] as List<List<List<T>>>;

if (value != null)

{

return value;

}

List<List<List<T>>> results = new List<List<List<T>>>();

if (count < groupCount || groupCount < 1 || count < 1)

{

return results;

}

//划分为1组时

if (groupCount == 1)

{

results.Add(new List<List<T>> { new List<T>(set) });

return results;

}

//将集合最后一个元素加入临时集合

var lastElement = set.Last();

List<List<T>> tempPartition = new List<List<T>>();

List<T> tempSet = new List<T> { lastElement };

set.Remove(lastElement);

//计算n-1,m-1时的划分

var temp = SetPartition(new List<T>(set), groupCount - 1);

foreach (var item in temp)

{

item.Add(tempSet);

results.Add(item);

}

//计算n-1,m时的划分

temp = SetPartition(new List<T>(set), groupCount);

foreach (var item in temp)

{

for (int i = 0; i < groupCount; i++)

{

//深拷贝

var newPart = new List<List<T>>();

item.ForEach(p => newPart.Add(new List<T>(p)));

newPart[i].Add(lastElement);

results.Add(newPart);

}

}

cache.Set(key, results, policy);

return results;

}

1. 在产生之前划分方案的基础上对货车的列表进行全排列，全排列算法如下（采用递归算法）：

/// <summary>

/// 全排列算法

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="nodes">需要进行全排列的元素列表</param>

/// <param name="count">全排列元素数量</param>

/// <param name="temp">临时列表，长度与nodes相同</param>

/// <param name="results">全排列</param>

public static void Permutation<T>(List<T> nodes, int count, List<T> temp, List<List<T>> results)

{

foreach (var node in nodes)

{

int a = nodes.Count();

//将第count - nodes.Count()个元素放入临时列表

temp[count - a] = node;

var spare = new List<T>(nodes);

spare.Remove(node);

if (spare.Count() == 0)

{

//将完成的排列存入结果集

results.Add(new List<T>(temp));

return;

}

Permutation(spare, count, temp, results);

}

}

1. 对如上的划分和排列进行匹配遍历，即可遍历所有分组方案，在明确一个配送方案的前提下，根据方案和仓库内液氧留存量的状态计算每天的配送计划，再确定每条线路上这一天的配送节点后，可以据此计算出这一天的配送路径和最短距离，而根据配送节点计算最短路径的算法也采用全排列的方式（算法同上），而每两个仓库之间的距离采用高德地图API获取



1. 根据每天的计划和每天用量更新各个仓库的库存状态，循环30次即可计算出一个月的行驶里程。