**《算法设计与分析》实验报告模板**

我保证没有抄袭别人作业!

1. **实验名称**

分支限界法

1. **问题描述**

1.01背包问题

给定3个物品，背包容量为20斤，各个物品的价值分别为

v=(25,24,15), 各物品的重量分别为：w=(18,15,10)。请给出一种物品选

择方案使得背包总价值最大，并输出该最大价值。

设计两种优先级定义

优先级：结点的价值上界，结点所获得的当前价值

2.单源最短路径问题

1. **算法设计与分析**
2. **算法选择**

分支限界法以广度优先或以最小耗费优先的方式搜索问题的解空间树.

由于采用了最小代价估值函数指导搜索，总是向最有可能成为最优解的子树上扩展.且使用限界函数杀死活结点表中不可能成为最优解的节点，提高效率。

1. **算法设计**

**算法名称**：01背包问题

**输 入：**物品价值数组p,物品重量数组w,背包容量c

**输 出：**能装在最多价值的方案solution

**算法步骤：**

1. 初始化背包容量和物品价值，重量
2. 创建根节点:表示背包为空
3. 计算并推送根节点到优先队列
4. 循环处理优先队列中的节点

弹出上界最大节点

计算该节点的左右子节点，并计算上界

若子节点可行，放入优先队列

（5）返回最大价值

**算法名称**：单源最短路径问题

**输 入：**物品价值数组p,物品重量数组w,背包容量c

**输 出：**能装在最多价值的方案solution

**算法步骤：**

1. 初始化数据结构，使用图的邻接表表示图，存储边的权重信息，创建优先队列
2. 将源点和代价0的节点入队列，每次从优先队列中取出代价最小的节点，如果是目标节点，则最短路径已找到，返回
3. 遍历当前顶点的所有邻接节点，如果能降到更低代价，则将新的定点加入优先队列
4. 队列为空时停止，如果目标定点未被访问过，则不可达
5. **算法复杂度分析**

(1)01背包

在最坏情况下，算法可能需要遍历所有的节点。对于一个包含*n*个物品的问题，每个节点代表一个决策点（选择或不选择某个物品），所以总的可能节点数最多是 2^n，即每个物品有两个选择（选或不选）。

由于优先队列的大小和操作次数与节点的数量成正比，因此最坏情况下，算法的时间复杂度为 *O*(2^*n*log2*n*)。

上界计算复杂度为O(n),共2^n个节点，时间复杂度为O(n2^n)

空间复杂度为额外的优先队列空间O(2^n)

（2）每次插入或删除操作的时间复杂度为 O(logE)，其中E是边的数量。图的所有边需要处理一次O(E),总时间复杂度为O(ElogE)

空间复杂度，用邻接表表示图，存储O(E),优先队列中有O(V)个顶点，空间复杂度为O(E+V)

1. **实验测试**
2. **测试用例和结果分析**

测试数据和用例同之前的实验一样，这里说明同之前实验。

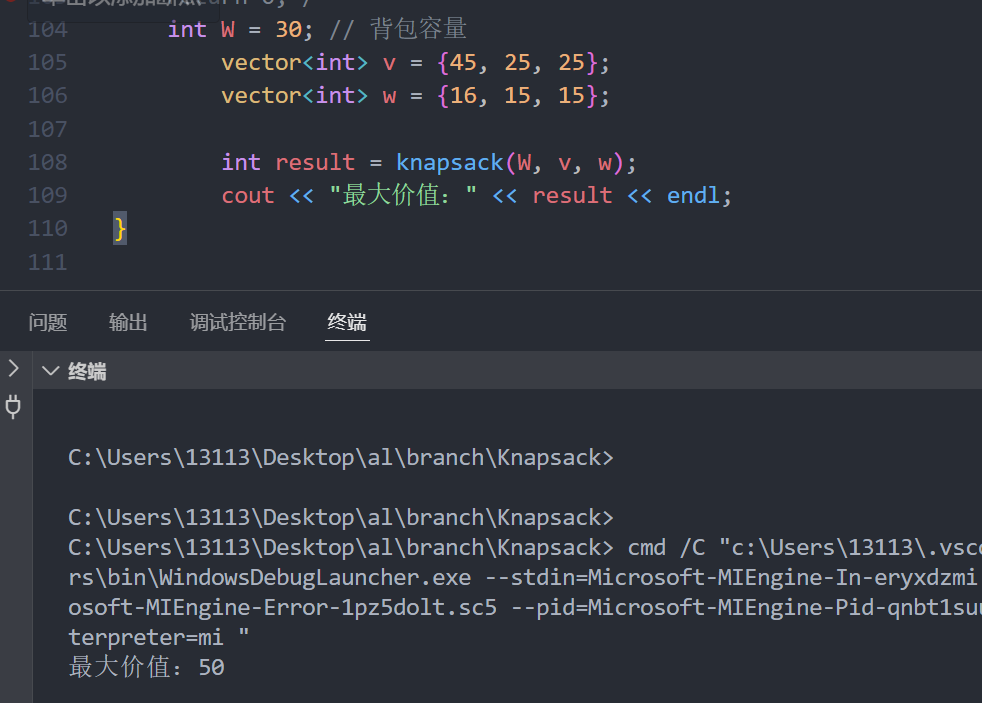
Testdata.hpp文件中的Testdata类函数生成，可以生成一维数组，乱序或有序，二维数组，以及随机字符串，可以根据数据所需求的规模设定生成测试用例。而测试时间由Timetracker.hpp中的measuretime函数实现，可以根据具体算法可以调整运行次数，以得到函数较为准确的运行时间，并且避免运行时间过久。

测试用例的数据规模会记录到datascale.txt文件中，而对应的运行时间记录在file.txt文件中，file为算法名称（与cpp文件同名），再经过Draw.py(该文件调用了python的matpilot库)将其绘制为坐标图，以更加直观的方式展现运行时间随着数据规模的变化，更好进行性能分析。

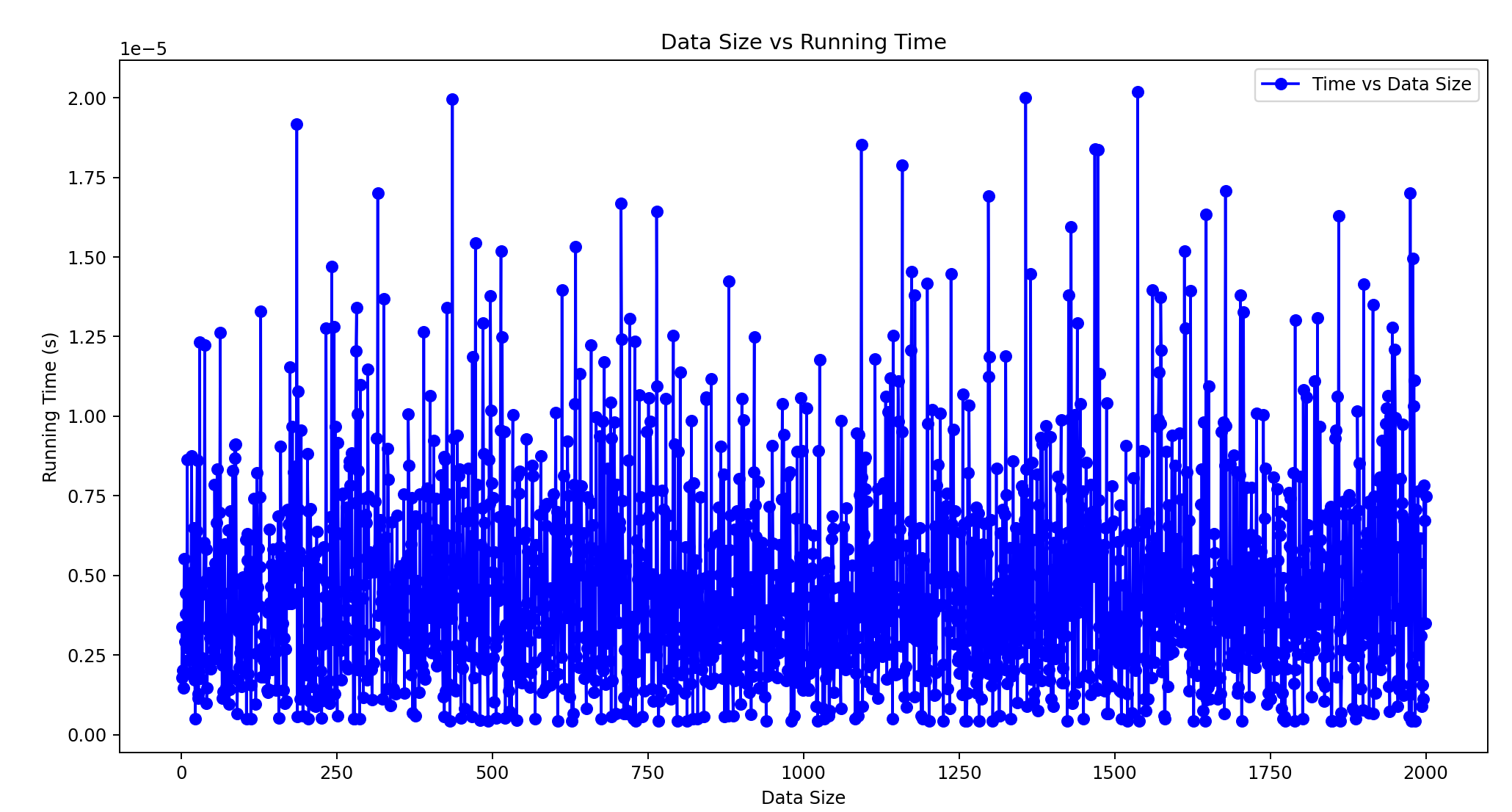
操作步骤，先运行算法.cpp文件，生成两个txt数据文件，再运行python文件根据数据文件进行绘制，即可复现以下测试用例的结果（由于测试数据每次随机生成，可能略微不同，为确保数据来源真实以及结果准确，会保留展示的测试用例数据，放在工作目录下的testdata.txt，同时便于进行算法的正确性验证）

1.01背包问题

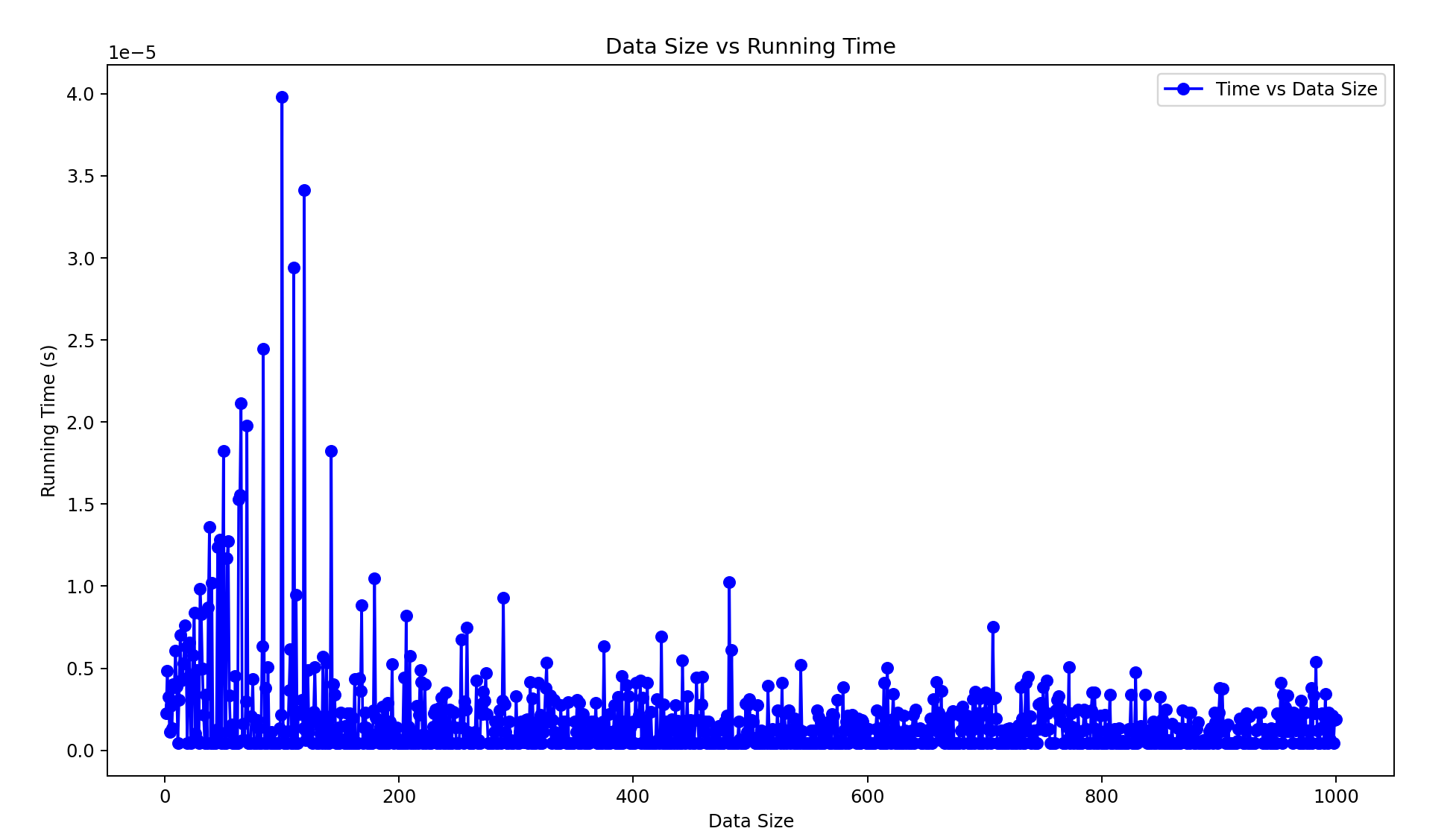
先使用课本例题进行测试,得到正确结果。后续扩大数据集规模，并将结果放入随机生成的数据后面，测试时间性能的同时，可以进一步确定其正确性。



选用的数据集规模从1-1000，并且完全随机生成，但测试结果和前面的分析的时间复杂度并不相符，且呈现一种较为均衡的情况，数据规模和对应的运行时间图如下



猜想可能因为上界估算在完全随机的情况下，可以做到比较均衡的运行时间。后面改用有序继续尝试，将物品价值按照从小打到排序，结果如下图。可以看到确实数据的随机对算法造成一定影响。在有序情况下，虽然数据规模增大，但限界法极好的缩小了运行时间，并稳定在很小的运行时间，这主要是因为上界估算处理得当，虽然数据集规模增大，仍然能避免很多不必要的试错。



2.单源最短路径

该问题需要的测试用较为复杂，在任务完成前未能实现更加普遍随机的数据测试用例，例如前面的数据规模可以连续逐渐增大实现画图的效果。这里仅选取了个别测试用例，将运行时间分别记录

用例顶点数和边数分别为(10,15),(100,500),(1000,5000),(5000,20000)而边数等的权重由之前的随机数生成函数生成,其中调用二维数组生成是生成相邻点

例如((0,3),(1,4))是生成一个从顶点0到定点3和定点1到定点4的边

，而调用一维数组是为该边赋予权重（存放在testdata.txt中）

得到运行时间分别为0.0107502,0.0070477,0.120469,2.50908

可以看到数据规模扩大运行时间不一定增加，但总体来说仍然是上升趋势。

同样测试数据规模，再次生成随机数据，使用贪心算法测试，运行时间为0.0068756, 0.0057082, 0.158741, 3.03401

和分支限界法差距并不算大

1. **实验总结**

分支界限法给人的第一感觉和回溯法有点像，但实际上区别点很多，在解决问题的实现中更有体会，分支界限法的设计需要朝着一个目标，总是选择更有希望的节点，优先去处理该分支，这样可以总体来说可以向着最优解的目标靠近。在本次实验遇到的最大困难是第二个实验要设计图的测试用例，由于时间不足，也没能向前面实验做出更大更一般情况的测试数据集，只能选择几个规模测试。这对后续的分析造成不小的影响。