data.py为数据读取及相关数据结构构造的class

主要包括

1、数据读取函数ReadData()，负责将sch文件读取并构造所有的数据存储结构

2、根据给定的mode序列（即每个活动选择哪个mode）来获取对应的数据，如mode的持续时间，使用各类资源数据等等

3、根据给定的mode序列（以及每一个mode的开始时间序列）来判断是否满足资源限制

4、根据给定的mode序列以及活动列表，来判断该序列是否有可能满足各个工序之间的时间限制

举个最简单的例子

act= [0,1,2,3,4,5,6]

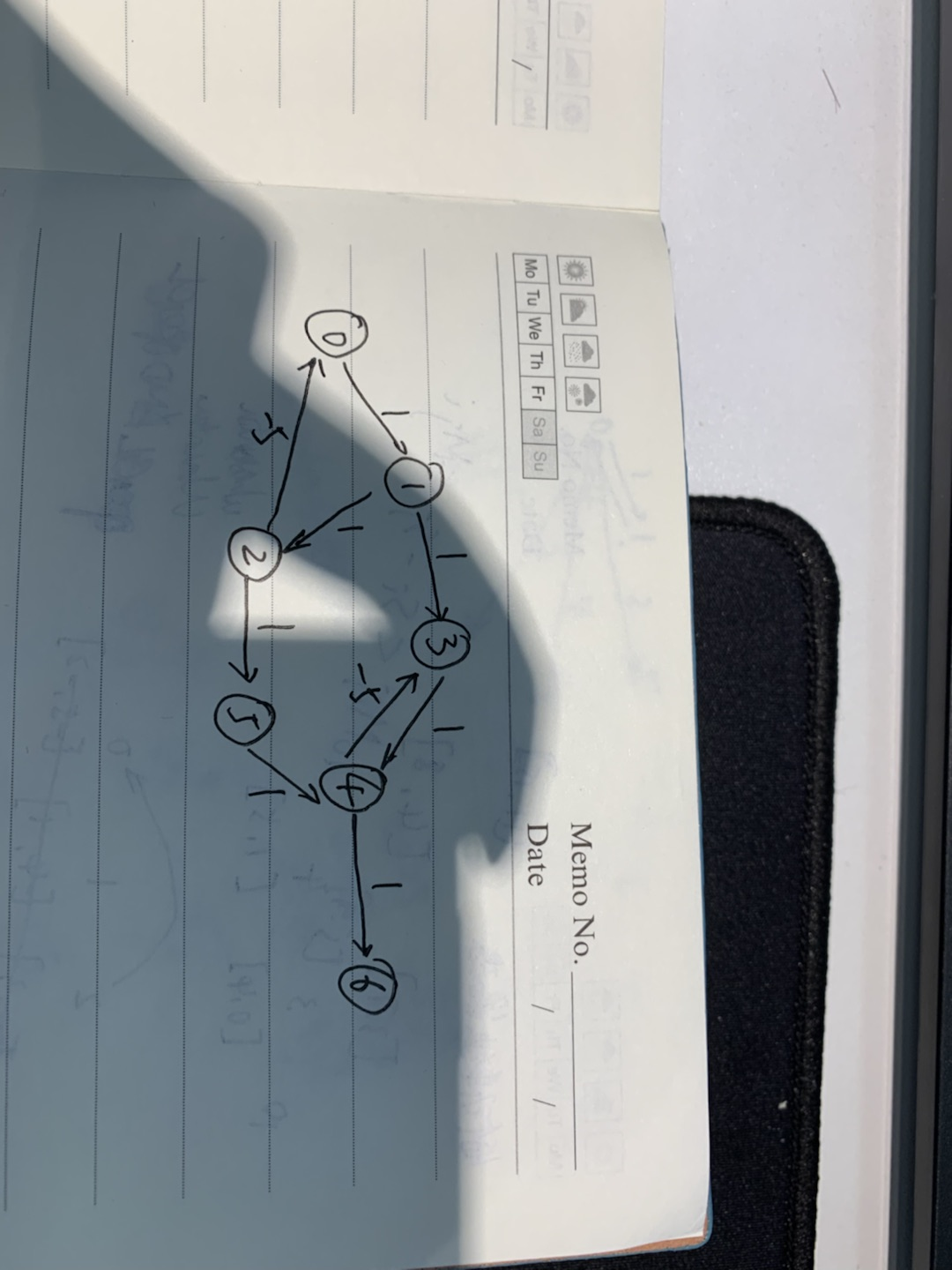
mode\_number=[1,2,2,2,2,2,1]

假设我选择的mode序列为[1,1,2,1,2,1,1]，根据给定的函数可以获取到所有的数据结构，这里假设该序列下的时间限制矩阵如下

sslimit=

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 | -inf | 1 | -inf | -inf | -inf | -inf | -inf |
| 1 | -inf | -inf | 1 | 1 | -inf | -inf | -inf |
| 2 | -5 | -inf | -inf | -inf | -inf | 1 | -inf |
| 3 | -inf | -inf | -inf | -inf | 1 | -inf | -inf |
| 4 | -inf | -inf | -inf | -5 | -inf | -inf | 1 |
| 5 | -inf | -inf | -inf | -inf | 1 | -inf | -inf |
| 6 | -inf | -inf | -inf | -inf | -inf | -inf | -inf |

有向图如下图



有向图边上的权值代表两个节点之间开始时间的限制，他代表的含义是ti-tj>sij其中sij为节点i指向节点j的边上的值，如上图则各个活动之间需要满足时间关系如下：

t1-t0>1,

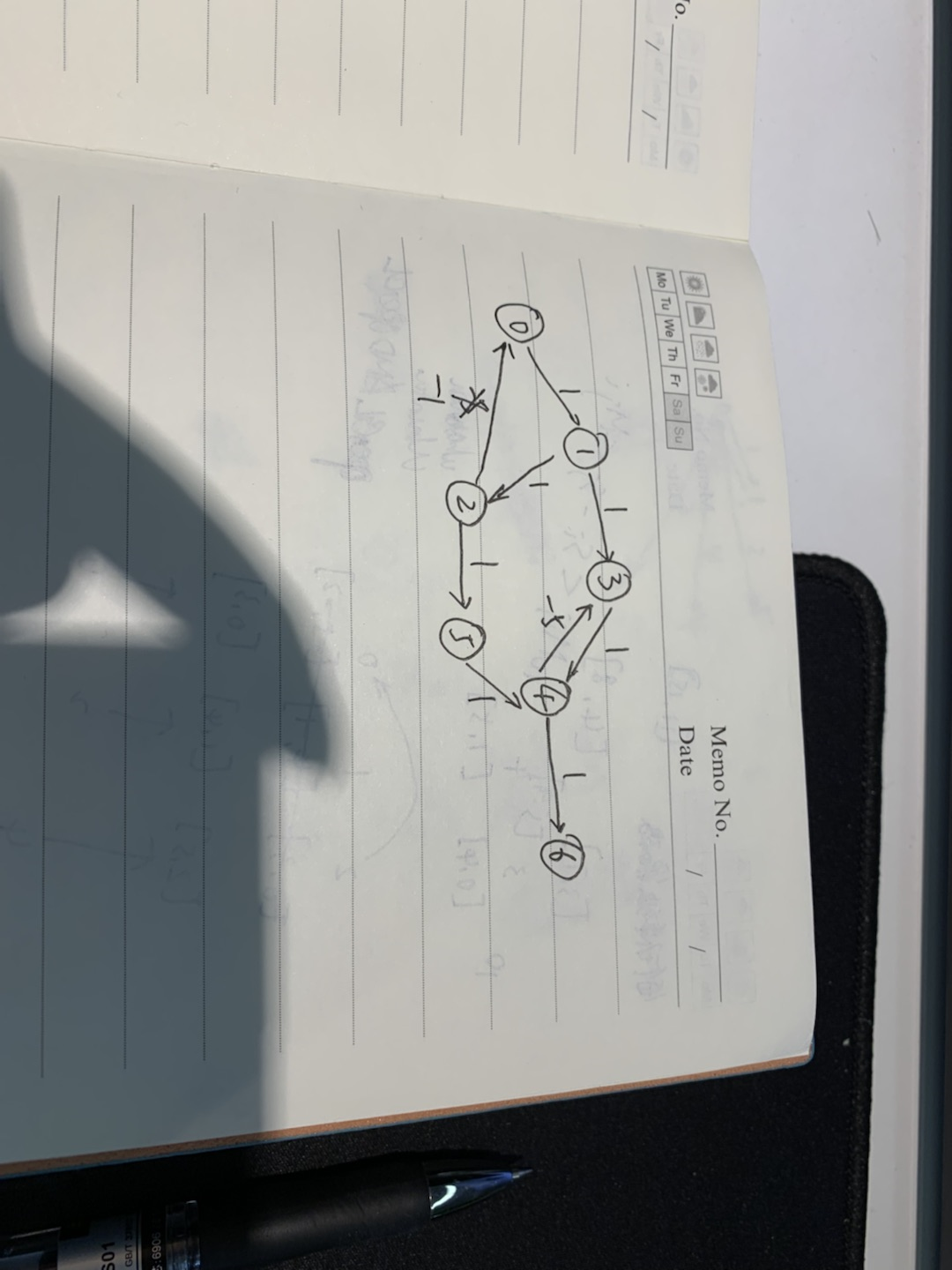
t2-t1>1,

t0-t2>-5,也是t2-t0<5

…………

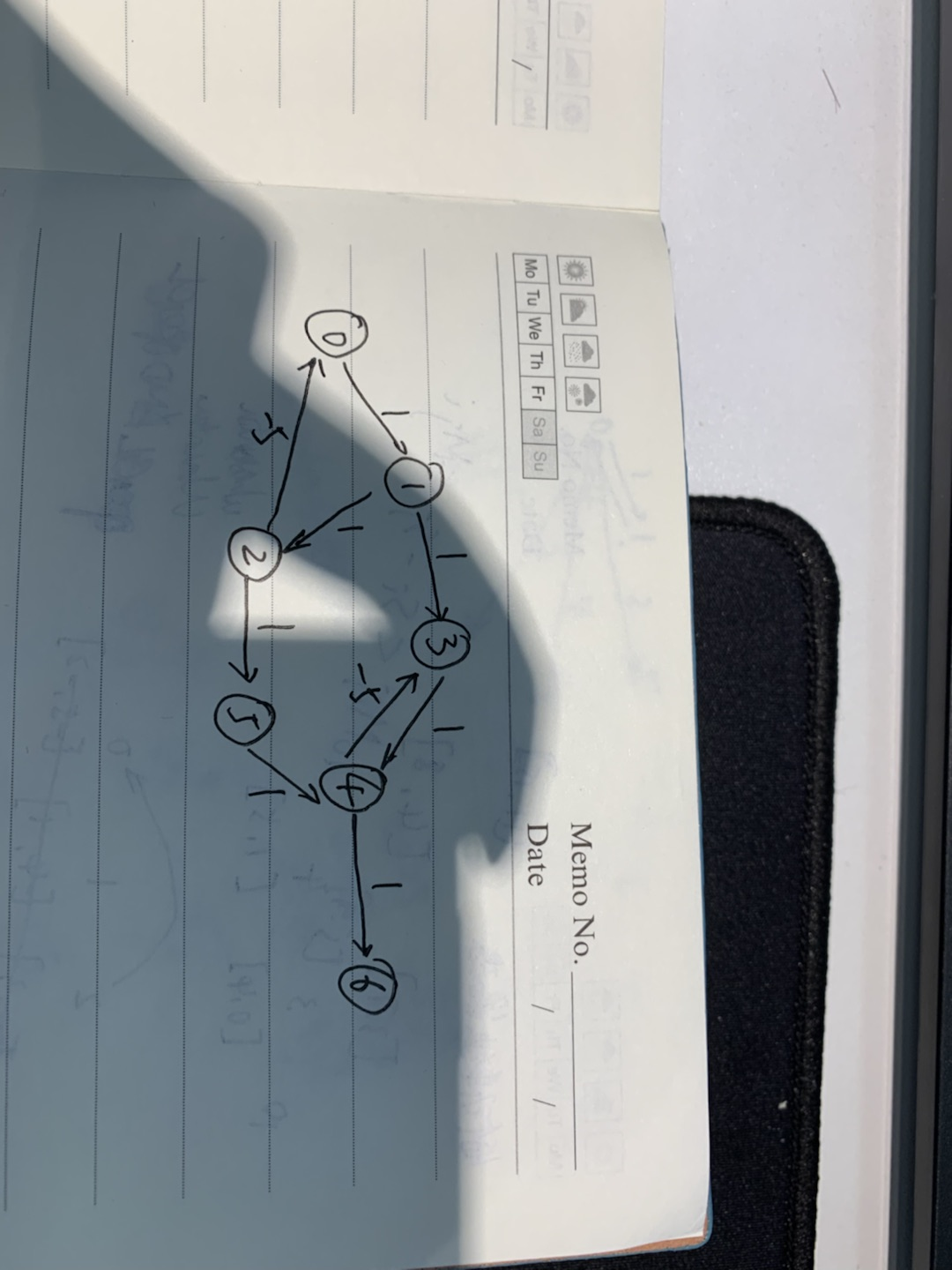
在给定活动（act）之间的关系后有向图就确定，选择不同的模式仅仅改变有向图边上的值。我们需要根据时间限制矩阵sslimit来判断这种模式选择是否可能存在，上图很明显这种时间限制下是存在可能的开始时间序列满足要求，那么我们就找到一个满足时间限制的模式序列[1,1,2,1,2,1,1]。

那么不存在的样子是什么样呢，如下图，假设我们对第二个act不选择mode 2，选择mode 1，假定此时的时间限制如下：



仅仅将t2-t0<5换成t2-t0<1很明显可以发现没有任何一种开始时间满足时间限制要求，那么这种模式序列[1,1,1,1,2,1,1]就是无法满足时间限制的模式序列，应该被丢弃。

那么在找到满足时间限制的模式序列后，我们需要确定每一个活动的开始时间，当然，在允许范围内，这个开始时间序列有很多情况，还是以上图为例



时间编码可以为[0,1,2,2,4,3,5]，也可以为[0,2,4,5,7,6,10]等等，只需要满足ti-tj>sij其中sij为节点i指向节点j的边上的值。

5、函数randomMode可以自动生成一系列的满足上述要求的mode序列，无需参数，返回的是mode序列

6、在获取到满足要求的mode序列后，需要对每一个mode确定开始时间，函数decodeTimeSequece可以完成该功能，输入函数randomTime生成的时间编码，函数ShortestPath生成的M矩阵以及人工设置的项目允许的最大延时d\_ba，即可生成一个满足时间限制的开始时间序列，具体使用如下：

d\_ba = 50

#待分析文件路径

filepath = r'G:\help\Archive\testset\_mm100\psp1-1.sch'

#最紧缺资源优先选择（'min'）or综合资源优先选择（'sum'）or完全随机选择（'none'）

para = 'min'

#构造readData类的实例个体，同时可以使用类中的各类方法以及数据

p = readData(filepath,para)

#读取数据

p.ReadData()

#生成一条mode染色体

state,mode\_chromosome = p.randomMode()

sslimit = p.buildGraph(mode\_chromosome,p.activities)

M = p.ShortestPath(sslimit)

#判断是否满足不可更新资源

if not p.Is\_NonRenewable\_Resource\_Feasible(mode\_chromosome):

continue

time\_chromosome = p.randomTime()

#解码获取实际的活动开始时间

state, start\_timesequence,\_ = p.decodeTimeSequece(time\_chromosome,M,d\_ba)

T = p.getProjectTime(start\_timesequence,mode\_chromosome)

#判断是否满足可更新资源限制

if not p.Is\_Renewable\_Resource\_Feasible(mode\_chromosome, start\_timesequence, T):

continue

能继续走下来的mode序列以及start\_timesequence开始时间序列，均是满足时间限制、资源限制的，可以参与代价函数cost计算。

具体使用方法参考GA-1中的initializePopulation函数