我们使用的存储结构大部分都是继承自 json 类型。

> 为什么直接写成或者说采用 json 类型?
>
> 1. 非常简单，用了这个感觉 c++能写出 其他高级语言 的快感。减少工作量。
> 2. 小工程犯不着用类来设计。但又确实存在需要继承的情况。以及需要灵活的可局部修改的数据结构。
> 3. 方便写日志，我们不用再写操作到字符串的转换函数。
> 4. 方便前后端通信

1. 定义航班表的存储结构：

```
// a flight schedule
{
    "from":"Beijing",
    "to":"Shenzhen",
    "start":[7,20],
    "end":[00:12:00],
    "prive":1190,
    "vehicle":"flight"
}
//0 refer to bus, 1 refer to train and 2 stand for flight
```

2. 定义一条行程查询请求的数据结构

```
// a requset
{
    "from":"Beijing",
    "to":"Shenzhen",
    "pathways":["Xizang"],
    "Timelimit":[::]
}
```

3. 返回的可行方案 一个可行方案是一个 schedule 的列表

```
//可以理解为 vector<json> plan, 但是为了存储还是直接定义成 json 类
json example_plan = {schedule_1,schedule_2,...}
```

4. 定义用户的行程

```
// 行程类journey 继承 plan 类，加上开始日期即可
json example_schedule = {...};
json example_plan = {example_schedule_1,example_schedule_2,...}
```

```
json journey(example_schedule);
journey["start_date"] = {"Mar","21"};
```

5. 定义用户的状态

```
// 当 location 非空时，journey 为空
{
    "location":"Beijing",
    "journey":null;
}
```

6. 定义日志格式

```
// 记录这个用户在这个程序上进行的一切活动
{
    "time":[12,00],
    "type":“”; //活动类型
    "content":string(json), // 将所有活动直接以json 类进行表示。
}
```

7. 定义绘图模块的输入

```
// 对绘图模块输入一个 plan 的列表即可
```

8. 定义等待中的行程 将等待中的还未执行的航程作以线性表进行表示。以 vector 模板存储。当进行撤销操作时，出栈即可。 当遇到在出发地与当前所在地不在同一地址时，提出警告并出栈。

```
vector<json> ScheduleList;//一个存放 schedule 的线性表
```