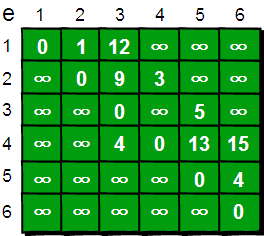
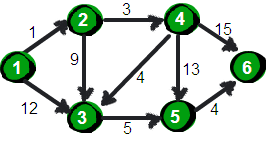
面经：<https://zhuanlan.zhihu.com/p/96008369>

1. Dijkstra算法：某点到所有点的最短路径<https://wiki.jikexueyuan.com/project/easy-learn-algorithm/dijkstra.html>



邻接矩阵表示，e[1][3]=12表示1->3的路径代价为12

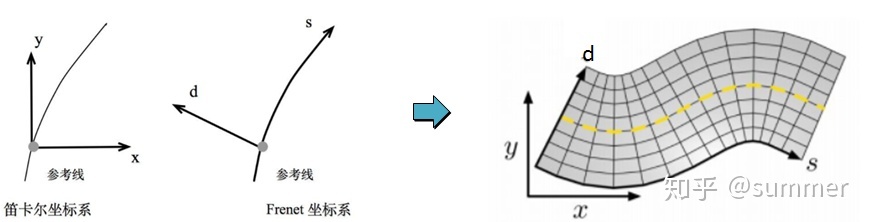
1）将所有顶点分为两部分：已知为最短路径的点集P和未知最短路径的点集Q。开始P中就只有自己

2）每次在Q中找一个不在P中，且到起点1的距离最短的点M。先把M点加入到P中，因为我们不可能找到一条路径能够从起点通过它到M的距离比当前M到起点的距离更短。然后对M所有的出边进行松弛，更新所有M出边点到起点的距离。

3）重复第3步，直到Q为空

1. A\*算法
2. 决策优化相关模型（MDP, POMDP等）
3. 车辆动力学模型
4. 配置空间：将障碍物膨胀后，机器人可以看成一个点，规划时不用考虑机器人的尺寸
5. 路径规划一般用BFS，因为DFS会一条道走到黑
6. Frenet坐标系

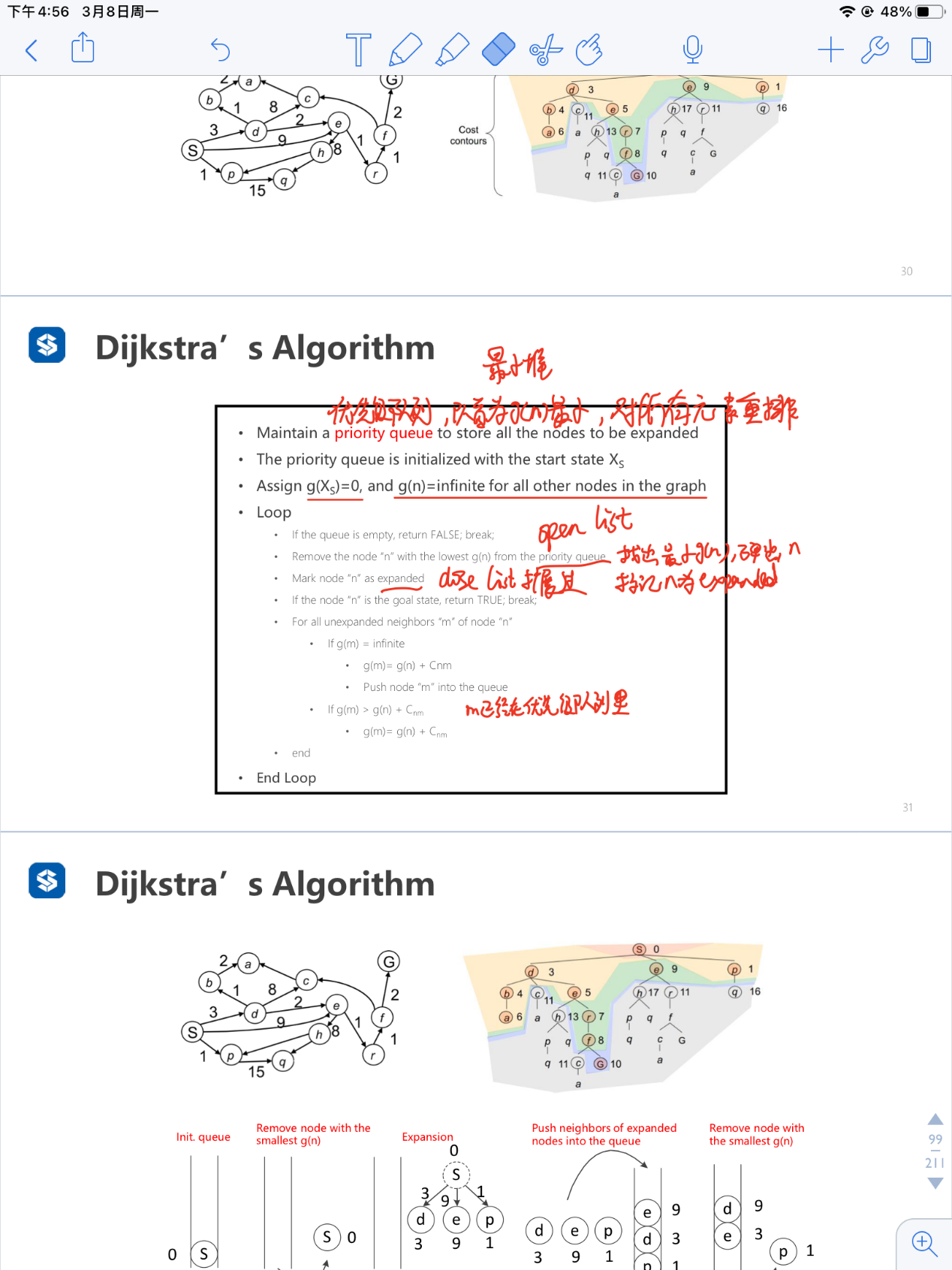
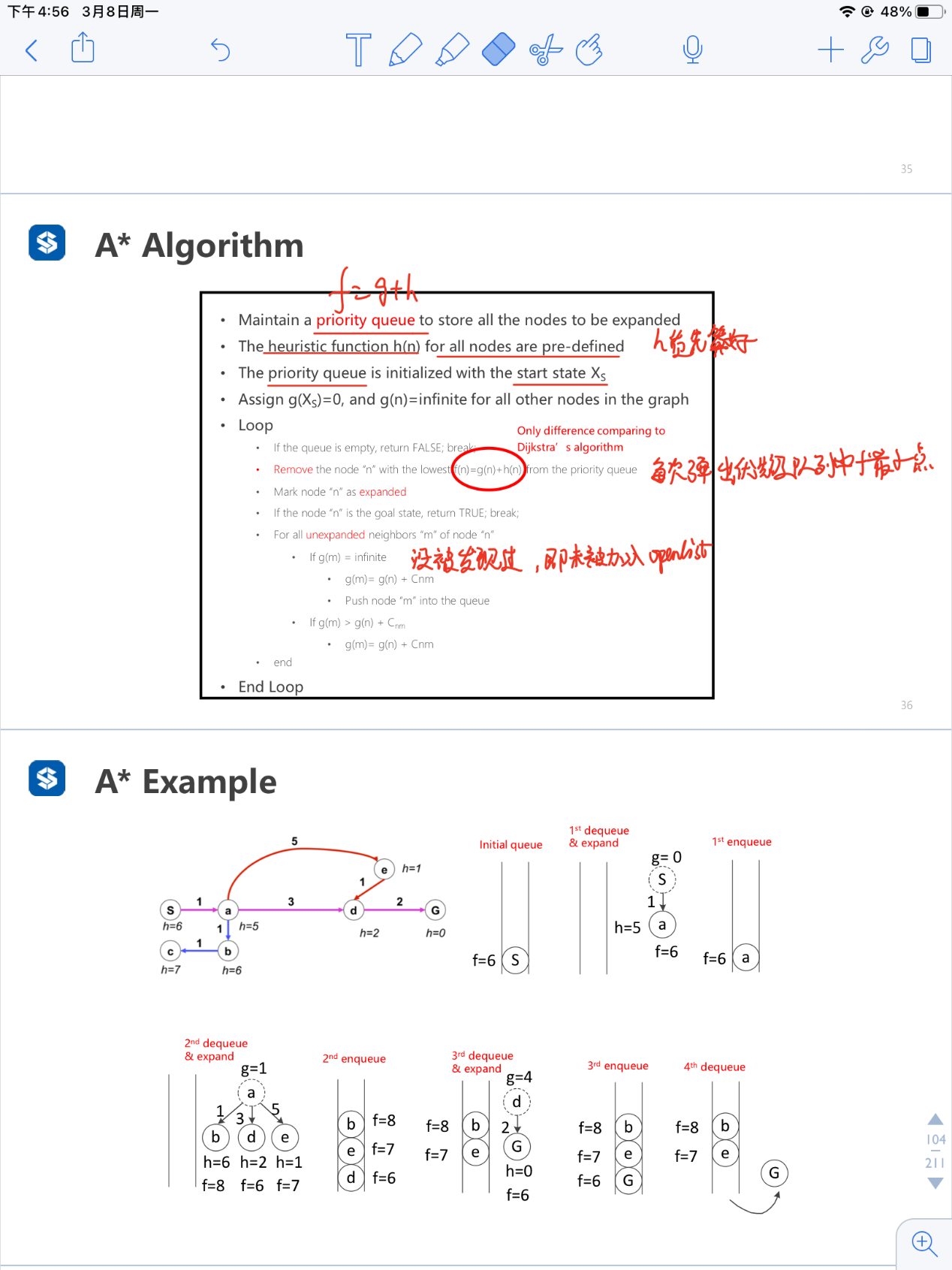
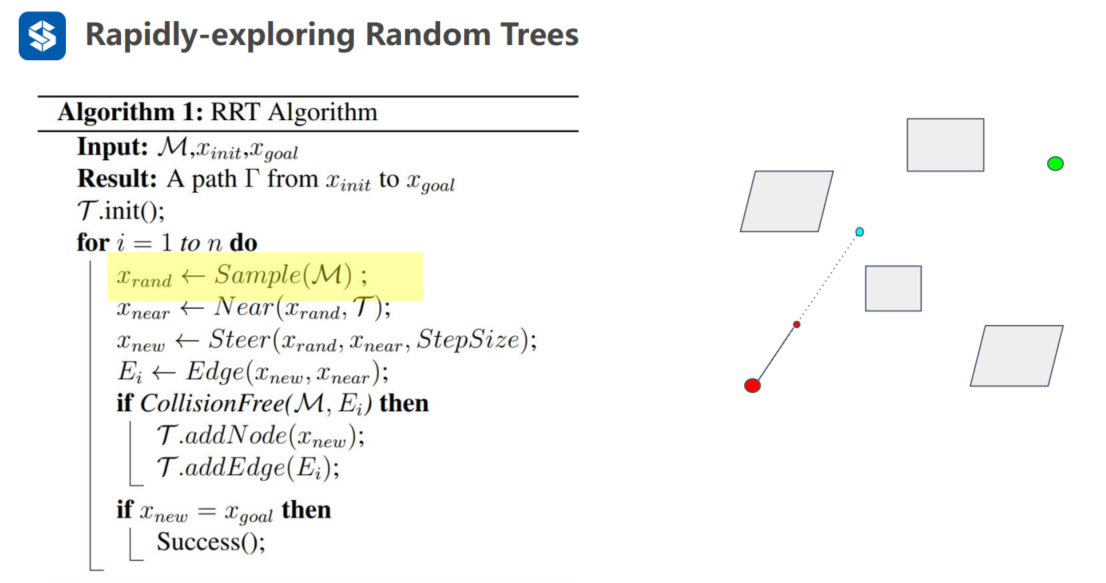
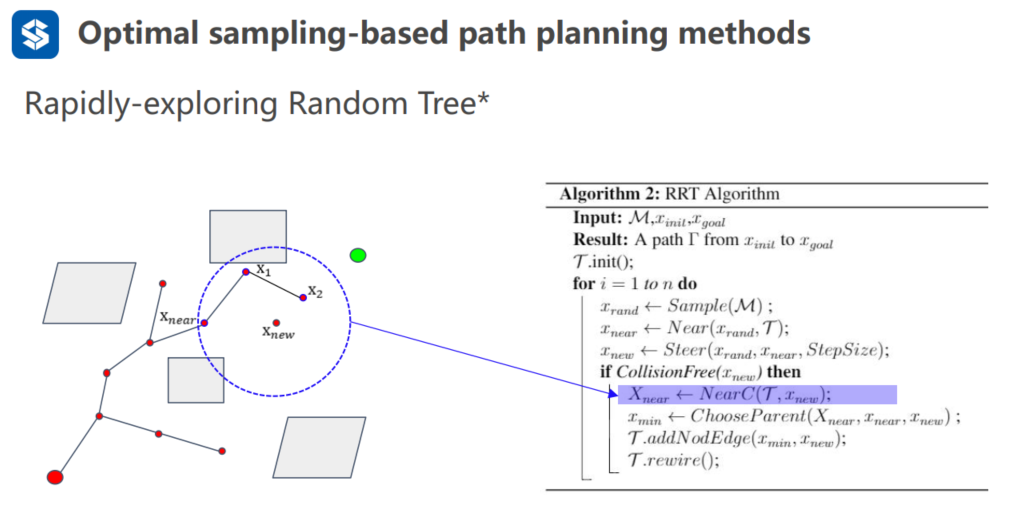
纵轴S代表沿道路的距离（纵向），横轴d代表与纵向线的位移（横向）



需求：因为控制只需要知道车辆所在车道位置以及与车道中心线的垂向距离，笛卡尔坐标系计算麻烦

优点：适用弯道的路线规划；简化了控制问题，因为在行驶中，道路的参考线很容易找到，输出横纵向距离等参数给控制层

1. Dijkstra，A\*，RRT，RRT\*

1. 路径规划算法的优缺点
2. 图搜索
   1. Dijkstra：贪心算法，计算一个点到其他节点最短路径。主要特点是以起始点为中心向外层层扩散，直到扩散到起点

优点：完备，最优

缺点：只能看到当前已经计算代价值的节点，所以需要向每个方向搜搜

没有目标点信息

* 1. A\*：启发式搜索，f(n)=g(n)+h(n)，可以更快找到路径

启发函数很重要，h(n)不要比实际到达终点的代价小太多，否则会搜索很久。

启发式距离有对角线距离（最好）、曼哈顿距离、欧式距离

需要打破对称性，减少点搜索：给每个代价加一个随机小量<1/1000

1. 随机搜索
   1. RRT：快速生成随机数

优点：可以找到起点到终点路径；更有指向性，因为随机点里面加入选中终点的概率，target\_oriented

缺点：不是最优

不高效（有优化空间，比如kdtree搜索最近点，双向搜索RRT）

在整个空间采样  
 路径拐弯太多（RRT\*）

未考虑动力学约束（kinodynamic-RRT\*）

* 1. RRT\*：比RRT增加了重新选择父节点、reWiring步骤

缺点：生成的路径不是最优

优化路径时优化空间在整个搜索空间，不高效

生成的路劲不符合动力学约束，比如说差分轮不能走直线

* 1. Informal RRT\*：RRT\*生成路径后，以始末点为焦点，路径长度为半长轴的椭圆里面 进行路径优化
  2. Kinodynamic-RRT\*：对Steer函数加入动力学约束考虑
  3. Anytime-RRT\*：终点确定，以车辆当前点为起始点，实时规划