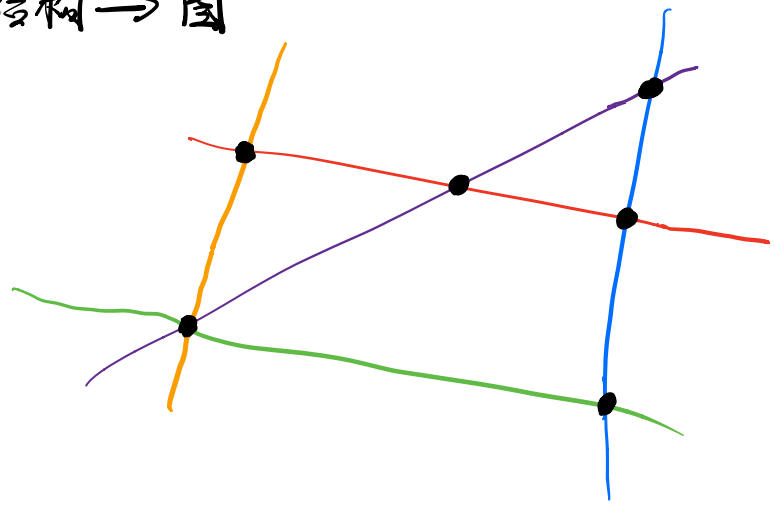
数据结构 → 图



因素：1) BFS · DFS 遍历方式

2) 开销（最小换乘、最短距离）

数据存储方式 → 图存储方式

邻接存储

Station 01 : { ~~Line 01 : [S_x, S_{x+1}, ... S_{x+n}]~~,

~~Line 05 : [S_y, S_{y+1} ... S_{y+n}]~~

}

Station 02 : { ... }

修正存储：

Station ⇒

Station 01 : { Line 01, Line 02 .. Line }

Station on : { Line x, Line y .. - Line }

Line => [$S_1, S_2, S_3, S, \ldots, S_n$]

算法 BFS Station x → Station y

while pathes:

path = pathes[0]
station x = path[-1]

lines = Lines contain station x AND not seen

for Line in Lines:

if Line contain Station y:

return path + Station y

next_stations = all - join - nale - Line

sort(next_stations, Strategy)

排序策略

pathes.addALL(path + every_next_stations)

排序策略例子

1: 最短路径

排序依据为比较各个点到当前点的路径

2: 最小换乘

在BFS中，默认即为最小换乘

再次明确数据存放格式：

1) 站点：所有站点所在的线路

String : List<String>

2) 线路：线路上所有的站点

String : List<String>

3) 线路：线路上所有的换乘点

String : List<String>

可合并

特殊处理

1) 环路 2号线 西直门↔西直门

影响点：

① 对换乘点in判断，计算可换乘点时需排除与自身线路in判断

② 对距离in判断

环路上任意两站in距离，存在两种计算方式，取较短的