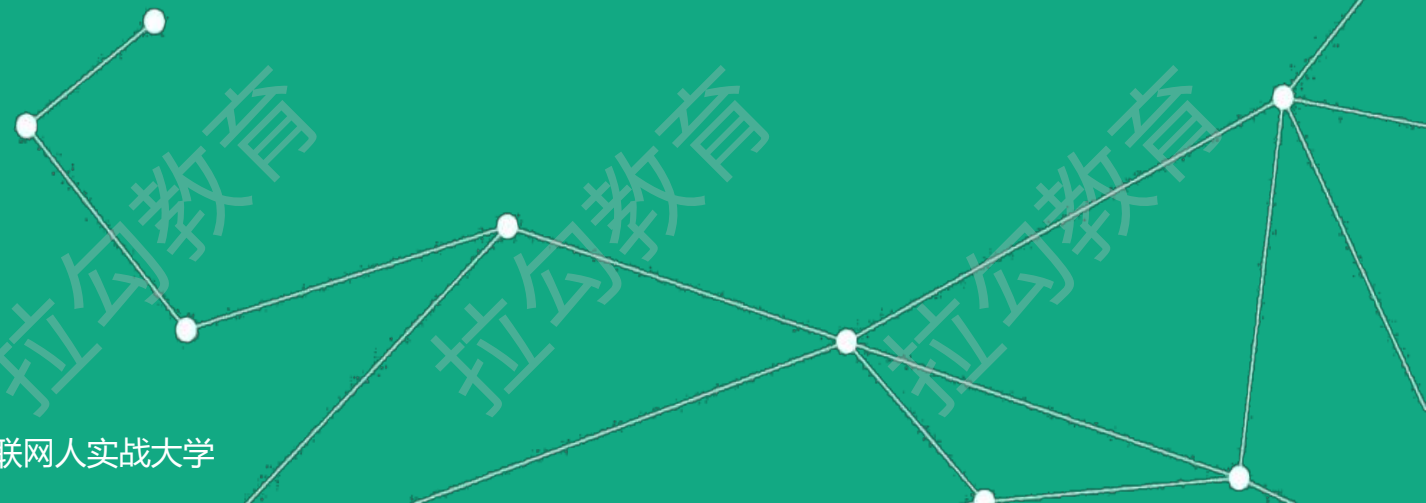
课时16

图在 Uber 拼车业务中的应用

1. 项目动机
2. 技术方案选择
3. 简单的匹配算法
4. 第三代拼车匹配系统
5. 总结



课前简介

Uber Pool 是 Uber 类似于滴滴拼车的共享乘车产品

怎样去设计一个好的拼车路线呢?

怎样才能去测量一个拼车路线是否既快捷又高效呢?

怎样才能准确预测几个乘客会共同享受这个行程?

项目动机

拉勾 LAGOU.COM

Uber 很早就已经对拼车这样的服务凯觎已久

想等到普通的打车服务的服务量达到一定规模再实行拼车服务

商业角度 是进行更冒险创新的时机

工程角度 原本的业务积累的大量数据可以为拼车服务提供数据支撑

技术方案选择

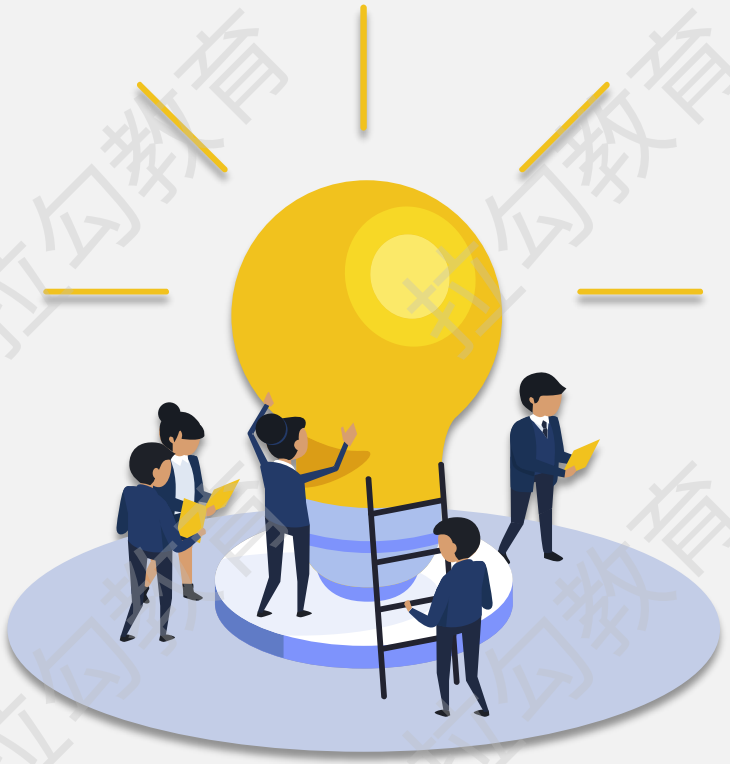
拉勾 LAGOU.COM

最开始的方案设计

根据乘客输入的起始点，会收到一个固定的报价

- Uber 拼车服务会让乘客等待 1 分钟，来匹配其他的乘客和司机
- 如果 1 分钟后没有找到匹配合适的同行乘客

  Uber 仍然会按照报价的金额派出单独的司机来进行接送



技术方案选择

拉勾
LAGOU.COM

动态窗口式的匹配系统

- 把乘客放在一个动态的匹配窗口

  比如 10 分钟的匹配窗口

  那么 8:03 和 8:09 开始叫车的乘客都会待在 8:00 ~ 8:10这样的一个窗口里等待匹配

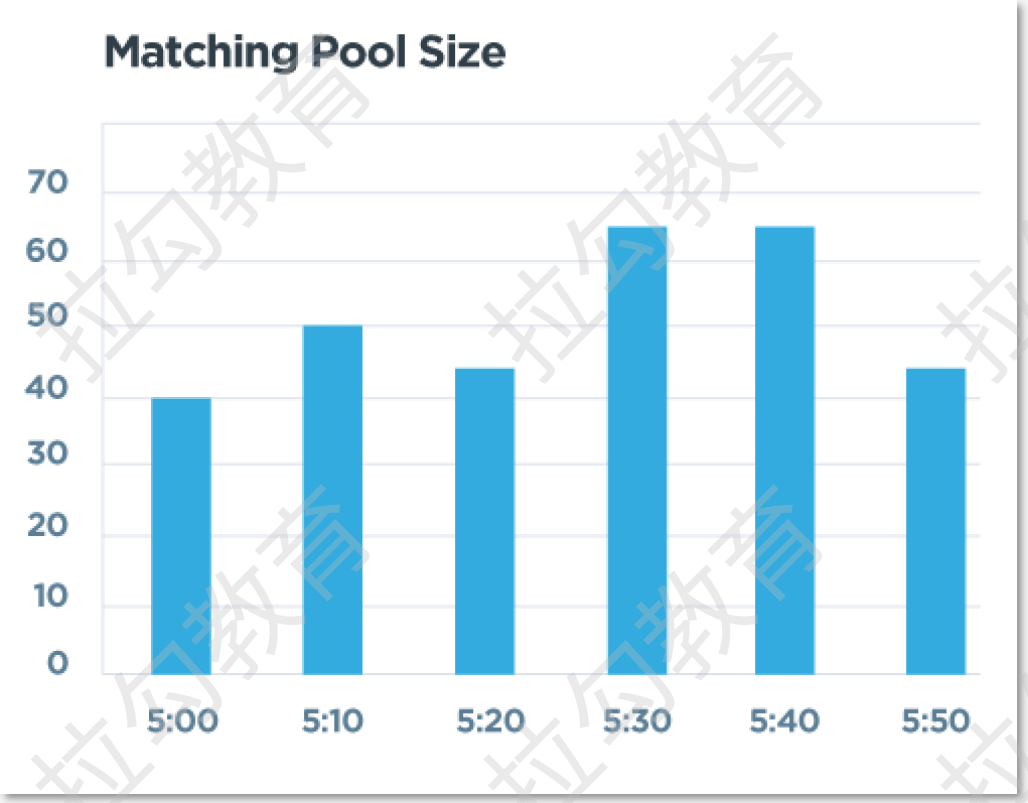
- 匹配窗口越大，系统能找到匹配的几率就越高
- 窗口不固定大小

  比如对于人口密度比较低的美国中部城市，窗口可以设到 15 分钟

  但对于人口集中的大城市，或者是上下班高峰时，窗口只需要 30 秒就能找到匹配同行的乘客了

技术方案选择

动态窗口式的匹配系统



同一个时间窗口里找到足够的司机变得很难

Uber 的系统必须要在 10 分钟窗口结束时找到拼车司机

10 分钟的窗口对用户来讲也有点长了

最终上线的 Uber 拼车服务等待时间为 1 分钟

简单的匹配算法

贪心算法的基本理念就是找到局部最优，而暂不考虑全局最优

一个贪心的匹配

就是先快速找到第一个符合条件的拼车配对组合而不考虑是否是这个时间窗口里最好的匹配

事实上贪心算法在产品上线的最初阶段是非常合适的

因为用户规模不大

往往一个时间窗口内可供选择的乘客也不多

如果非要追求全局最优则需要花费大量的时间

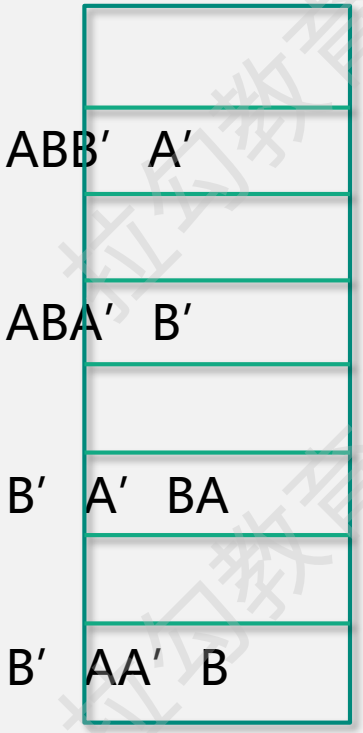
简单的匹配算法

最初的系统会便利所有的两两乘客，也就是 O(n^2) 的时间复杂度

这里面的 n 是多少呢

A 的起始点为 A 和 A′

B 的起始点为 B 和 B′



4*3*2*1 种可能性
但实际上合理的组合只有 4 种
因为 A′ 终点不可能出现在 A 起点之前，同时，AA′ BB′ 这样的组合也不存在

简单的匹配算法

对于一个有 n 个乘客的系统来说

需要遍历 4*n^2 种可能，然后再来寻找最优的组合

```
def make_matches(all_rides):
    for r1 in all_rides:
        for r2 in all_rides:
            orderings = []
            for ordering in get_permutations(r1, r2):
                if is_good_match(r1, r2, ordering):
                    orderings.append(ordering)
            best_ordering = get_best_ordering(r1, r2,
orderings)
            if best_ordering:
                make_match(r1, r2, best_ordering)
            // etc ...
```

简单的匹配算法

一个合理的拼车组合，对于每一个乘客来说

必须要保证拼车之后的绕路要在一个可接受的范围以内

对于一个 A → B → A′ → B′ 的组合

A 的绕路是 A 到 B 再到 A′ 的行程，减去 A 到 A′ 的行程，相似的也要计算乘客额外的等待接驾时间

对于 A→ B→ A′ → B′ 这样一个行程

A 的等待时间是司机→ A 的时间，B 的等待时间是司机→ A → B 的时间

一般来说第二个乘客的接驾时间更长，但是绕路时间更短

第三代拼车匹配系统

拉勾
LAGOU.COM

迭代到第三代拼车匹配系统时

Uber 的工程师已经认识到了再不使用图这个武器，已经无法完成业务需要了

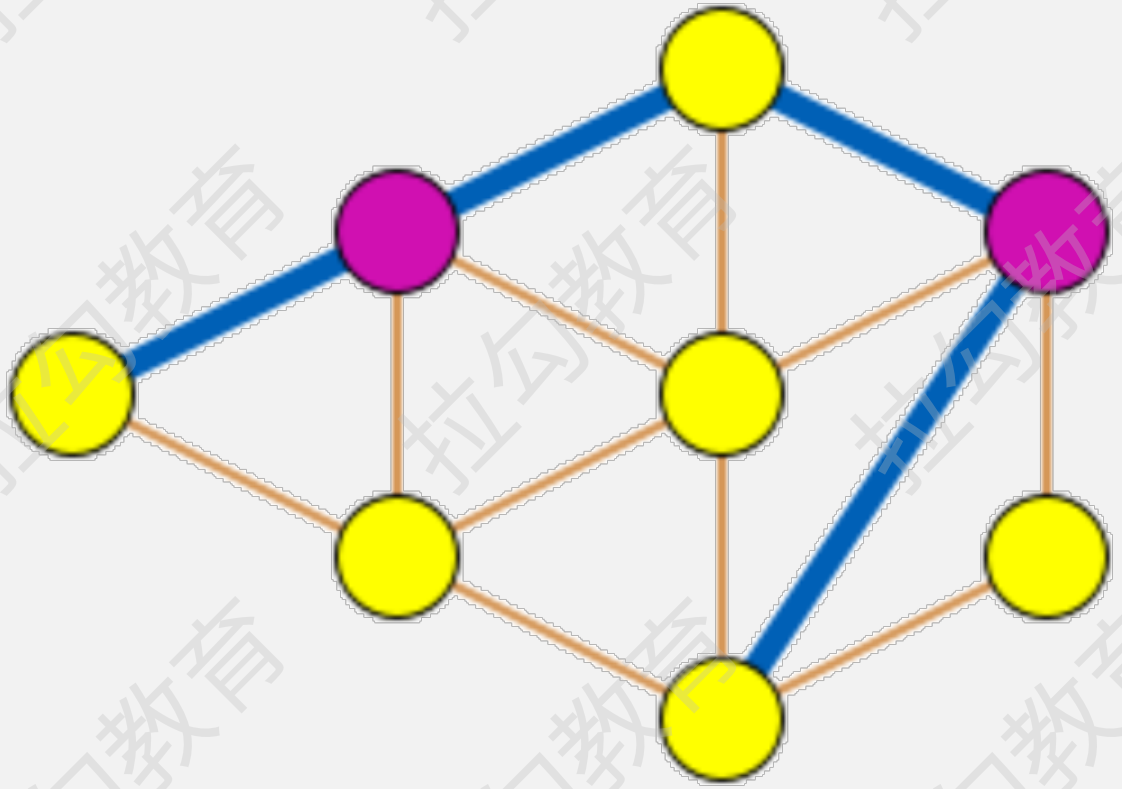
这也正是告诉我们学好数据结构的重要性

没有数据结构的深厚功力，很容易无法承担业务的重大挑战

第三代拼车匹配系统

拉勾
LAGOU.COM

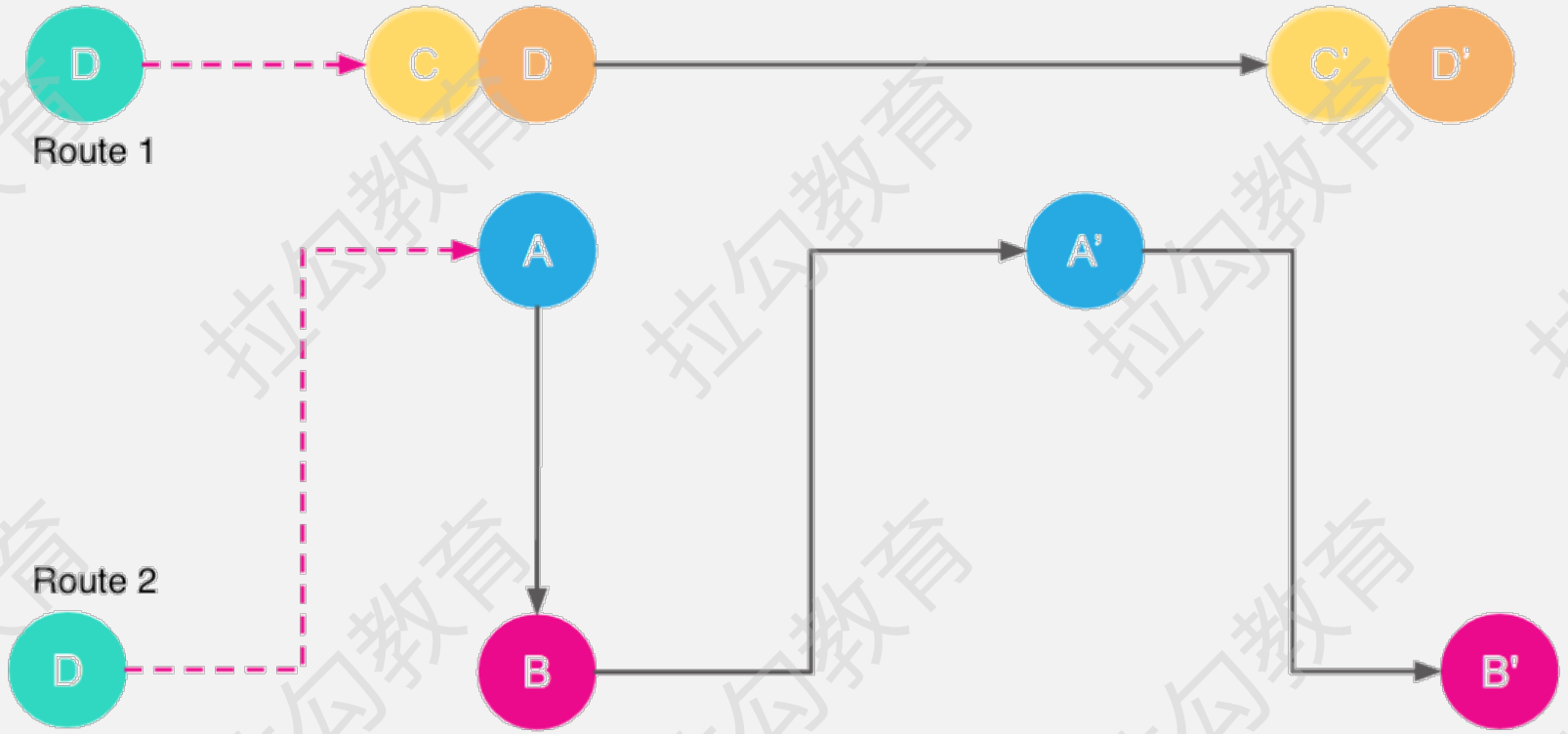
最大匹配算法



第三代拼车匹配系统

拉勾
LAGOU.COM

路线切换



总结

- Uber 的拼车系统属于图的最大匹配问题，可以看到 Uber 是怎样一步一步迭代技术方案并最终逼近最优解的过程，同时对于我们技术团队的方案设计也很有启发
- 在早期业务比较小时千万不要过度优化，而是要用最简单的方案快速验证收集更多用户和数据来不断改进系统