online algorithm introduction

-PROMISE-

Friday 1st April, 2022

1 乘客分配问题

• 这是一个引入问题，离线算法是指算法运行时可以充分利用输入数据之间完整的关联特性，即算法运行前所有输入数据均是已知的。在线算法是指算法必须在输入数据不是完全可知的情况下，完成相应的计算并输出计算结果。在线算法均是近似算法，引入问题如下：

给定 m 辆车和他们的路线，n 个乘客，其中第 j 辆车的容量是 k_j，算法要求决定乘客上那辆车使得可接受的乘客数量最多。(因为考虑的是在线算法的情况，所以不能获得全部的车辆和乘客信息，而是分步获取，所以我们主要是用贪心等思想去解决问题)

• 解答：

-贪心算法 Most Remaining Seats First (MRSF)

每遇到一个乘客，在可以达到它目的地的车辆里选择剩余容量最大得，如果容量小于等于 0，拒绝该乘客，否则接受该乘客

Algorithm 1 Most Remaining Seats First (MRSF)

Input: $m, (S_1, \ldots, S_m), (k_1, \ldots, k_m)$.
Output: The passenger set in each bus $B_1, \ldots, B_m$.

for $j \leftarrow 1$ **to** m do
  $B_j \leftarrow \emptyset$
  $R_j \leftarrow k_j$
while a new passenger (i, d_i) arrives **do**
  $j^* \leftarrow \arg\max_{j: d_i \in S_j} R_j$
  if $R_{j^*} \leq 0$ **then**
    reject the passenger (i, d_i)
  else
    $B_{j^*} \leftarrow B_{j^*} \cup \{i\}$; $R_{j^*} \leftarrow R_{j^*} - 1$
return $B_1, \ldots, B_m$

-对偶算法 Improved Primal-Dual (IPD)

取一个对偶因子作为标准计算，每个对偶因子都有对应的车

Algorithm 2 Improved Primal-Dual (IPD) Algorithm

Input: $m, (S_1, \ldots, S_m), (k_1, \ldots, k_m)$.
Output: The passenger set in each bus $B_1, \ldots, B_m$.

```
for j ← 1 to m do
    B_j ← ∅;   z_j ← 0
while a new passenger (i, d_i) arrives do
    j* ← argmin_{j: d_i ∈ S_j} z_j
    if z_{j*} ≥ 1 then
        reject the passenger (i, d_i)
    else
        B_{j*} ← B_{j*} ∪ {i};   y_i ← 1 − z_{j*}
        z_{j*} ← z_{j*}(1 + 1/k_{j*}) + 1 / (k_{j*}[(1 + 1/k_{j*})^{k_{j*}} − 1])
return B_1, ..., B_m.
```

2 竞争比 Competitive Ratio

• 竞争比是一种评估在线算法质量的概念，可以理解为，当我们的在线算法与离线算法的最优算法保持代价同步时，这个在线算法是一个较优的算法。

• OPT($\mathcal{I}$) 为实例 $\mathcal{I}$ 可行解的最优代价，在在线算法中，这个实例是分步输入的。

Minimum：当在线算法 $\mathcal{A}$ 的代价对于问题 $\mathcal{I}$ 的所有实例，都不大于 $c\dot{O}PT(\mathcal{I})+\alpha$

Maximum：对于在线算法 $\mathcal{A}$，对于 $\mathcal{I}$ 的所有实例，他的算法代价至少是 $\frac{OPT(\mathcal{I})}{c} - a$

$c \geq 1$ 且 α 独立于请求序列，此时我们称满足上面条件的算法 (A) 是 c-competitive，当 a=0 时，是 strictly c-competitive。

• 竞争度分析总结：算法 A 是一个问题 A 的在线算法。对于任意实例 $\mathcal{I}_A$，$OPT(\mathcal{I}_A)$ C_{on} $C_{on}(\mathcal{I}_A) \leq c\dot{O}PT(\mathcal{I}_A)+\alpha$ 对任意实例 $\mathcal{I}_A$ 成立，其中 α 是与输入规模 $|\mathcal{I}_A|$ 无关的常量，在则称算法 $\mathcal{A}$ 的竞争比为 c, 若不存在更小的 c，则该算法是最优的。

• 用 Paging Algorithm 为例说明：

考虑四种页面替换算法：

LRU (Least Recently Used): On a fault, evict the page in fast memory that was requested least recently.

FIFO (First-In First-Out): Evict the page that has been in fast memory longest.

LFU (Least Frequently Used): Evict the page that has been requested least frequently.

OPT (Furthest in the Future): On a fault, evict the page whose next request occurs furthest in the future.

注：页面置换算法 https://www.jianshu.com/p/18285ecffbfb

- 先说结论：LRU 和 FIFO 是 K-competitive 算法，K 是分配的内存块数量
- 证明：假设我们有一组页面请求序列:

$\sigma = 1, 2, 3, 2, 4, 2, 5, 2, 1, 2, 1, 3, 1, ...$

内存块数量为 3.

因为我们的目的是比较 LRU，FIFO 和 OPT 的代价，而在这个例子中，代价就是页面替换的次数，所以我们首先需要找到页面在哪些位置发生了替换，对整个序列进行分割考虑的处理，因为有三个内存块，所以我们很合理的想到，根据三个内存块占满的情况分割序列（即一个块中只有 3 种不同的页面）

分割序列: $\underbrace{1, 2, 3, 2}_{\sigma_1}, \underbrace{4, 2, 5, 2}_{\sigma_2}, \underbrace{1, 2, 1, 3, 1}_{\sigma_3}...$

由此我们发现，对于每一个模块 σ_i，其中有三个不同的页面，并且 σ_{i+1} 中的第一个页面一定不与 σ_i 中的任意一个页面重复。有了这样的模块分割，我们就可以继续之后的代价分析了。

考虑 LRU 和 FIFO 的原理，我们假设模块种的三个页面都是目前没有在内存当中的（由于 LRU 和 FIFO 的执行规则，我们替换掉的不可能是该模块内的页面，所以最多是 3 次），则最多需要替换 3 次，可以理解为代价是 3;

考虑 OPT 的情况，假设 P_i 是 σ_i 中的第一个页面，而 P_{i+1} 是 σ_{i+1} 中的第一个页面，由我们分割的规则可以得到，P_{i+1} 一定不在 $\sigma_i - P_i$ 中，并且由于首先处理了 σ_i 中的页面，此时内存块中一定是 σ_i 中的那三个不同页面，因此 OPT 算法在新分割的模块 $\sigma_i^{'}$ 中一定需要替换一次页面。

综上，设 t 是模块 σ 的个数，则 $OPT \geq t - 1$，$Cost_{LRU,FIFO} \leq 3 \cdot t$.

更一般的情况，内存为 K，则有 $OPT \geq t-1$，$Cost_{LRU,FIFO} \leq K \cdot t$，所以有 LRU 和 FIFO 是 K-competitive 的.

- **如何判定在线算法是否是最好的呢？**（也就是说如何判定在线算法至少是 K-competitive 的）

其实我们只需要找到一个特例，在该特例中，所有在线算法 $\mathcal{A}$ 都至少是 K-competitive 的

特例如下：

我们取一个模块 σ，其中包括 K+1 个不同页面，且假设所有页面在内存中都处于 missing 状态，所以对于在线算法 $\mathcal{A}$ 来说，所有的页面都触发一次缺页处理，代价是 K+1。对于 OPT 算法而言对于所有的缺页他替换的页面因为是未来最久不使用的，所以至少是 K 个页面以后的某个页面被替换（由假设知道这 K+1 个页面都缺页，所以都不处于内存中），故在 OPT 算法下，最多缺页处理一次，可以理解为代价是 1. 所以 OPT 在每 K 页至多缺页一次，而 $\mathcal{A}$ 每次都缺页，则 $\mathcal{A}$ 是至少 K-competitive 的.

3 线性查找（Linear Search）

- 题目可以理解为是在一条线上找一个点 X，该点与起点 O 之间的距离为 D，且只要经过该点就里刻停止，算作找到。从起点出发，有左右两个方向可以选择。
- 解答：总的来说就是左右走，每次距离原点的距离指数增长

Algorithm

```
d=1
"current side" = right
repeat:
    Walk distance d on "current side"
    if find the pasture then
        Exit
    else
        Go back to the starting point
        d = 2·d
        Flip "current side"
```

- 接下来我们分析它的竞争比：

因为我们设置的在线算法 $\mathcal{A}$ 是指数增长的，为了计算方便，设置 $D = 2^j + \epsilon$，则 $OPT = 2^j + \epsilon > 2^j$，而对于 $\mathcal{A}$，有 $ALG = 2 \cdot (1 + 2 + 4 +$

$... + 2^j + 2j + 1) + 2^j + \epsilon = 9 \cdot (2^j + \epsilon) - (8 \cdot \epsilon + 2) < 9 \cdot OPT$, 所以这个算法是 9-competitive 的

4 读取字符并依次存入数组（Read the Symbols into the Array one by one）

• 题目主要的问题是我们不知道要开辟多长的数组，如果太长会导致数组浪费，太短会不够用。

• 解决方式是每当数组满的时候，将数组大小翻倍

• 证明竞争比是 3：

设长度是 $2^j + \epsilon$，则 $OPT = 2^j + \epsilon \geq 2^j$, 而 $ALG = (1 + 2 + 4 + ... + 2^{j+1}) = 2^{j+2} - 1 = 3 \cdot 2^j + 2^j - 1 \leq 3 \cdot 2^j$, 所以是 3-competitive 的

5 美团优惠券