渡河问题——贪心策略

原创 李然 LOA算法学习笔记 2021-01-10 22:46

01 问题描述

一牛群需要渡河，共有n头牛。放牛人每次只能赶两头牛渡河，每头牛渡河时间为t_i，每次消耗时间由其中最慢的牛决定。注意放牛人渡完一趟后，需要骑一头牛返回对岸。

给出贪心策略和算法设计使得放牛人将n头牛全部渡完河需要的最短时间？

02 问题分析

首先，设定一个情形，还有4头及以上的牛还未渡河。设其中的四头为a,b,c,d，且所需时间a<b<c<d。现在想让c,d过河然后放牛人再返回运送其他牛。有以下两种方式：

1) 过河顺序为ac、a、ad、a，时间消耗t_1= max{a,c}+a+max{a,d}+a；

a,b,c,d …	T=max(a,c)=c →	a,c
a,b,d …	T=a ←	c
b …	T=max(a,d)=d →	a,c,d
a,b …	T=a ←	c,d

2) 过河顺序为ab、a、cd、b，时间消耗t_2=max{a,b}+a+max{c,d}+b；

a,b,c,d …	T=max(a,b)=b →	a,b
a,c,d …	T=a ←	b
a …	T=max(c,d)=d →	b,c,d
a,b …	T=b ←	c,d

即：

$$t_2 - t_1 = a + c - 2b$$

因此，选择上述两种方案中的哪一种，取决于t_2-t_1。

第一种方案的解释：利用消耗时间最小的a来分别送c，d过河，因为a耗时最短，所以每次a把放牛人送回来时间也最短，所以第一种方案可能是最优方法；

第二种方案的解释：如果c，d都需要很长时间才能过河，那么就将他们一起送过去，这样能够有效解决消耗时间较长c的。然后让先前到达对岸的b（之前a，b先到达对岸）运送放牛人回来。上述两种方案是唯一两种比较有前途的送c，d方案。

03 贪心策略

现将贪心规则设定为：在未过河牛数量n≥4的时候，我先用上述两种方法中较好的一个，把耗时最大的两个送过河（用过河时间最少的牛作为上述方法的a，第二少的作为上述方法的b）。该问题转化为寻找把剩下的n−2头牛送过河的最优策略。

循环执行此策略，直到n=2时，将最后两头牛直接送到对岸n=3时，用耗时最少的牛分别搭配把另外两个送过去为最优（三头牛过河，显然就是：过去两头，回来一头，在过去两头，两次过去两个的花费分别为：b、c，那这个回来的牛，应该是a才能最快，也就是让a分别送b、c）。

04 方案证明

4.1 首先证明，为什么先送的是最慢的两头牛?

1) 如果$a_1+a_{n-1}-2a_2<0$，则说明，用耗时最小a_1分别单独送a_{n-1}和a_n优于a_{n-1}和a_n一起过河。因此由于其他$a_i<a_{n-1}$，其中$i<n-1$， 即$a_1+a_i-2a_2<0$。所以让最小的a_1分别送所有的牛，优于让他们中的任意两头牛组合一起过河。所以这种情况下过河的总过程可以看成是，判断最慢的两头牛是否应该一起过河，然后再继续判断下一个。

2) 如果存在若干a_i，满足$a_1+a_i-2a_2>0$，那么就说明，满足这个条件的这些a_i都有一个特点，让它和任意一头耗时比它长的一起过河，都应该是要优于让耗时最小的牛分别送它和那个比它大的过河。

那么现在需要说明的是，对于满足这些条件的ai，应该如何组合，才能得到最优结果：

设$x=2a_2-a_1$，那么有$a_1\le a_2\le a_3\le\cdots\le a_{i-1}\le x\le a_i\le\cdots\ \le a_n$

现在可以清晰看出，x就是一个分界点，x左边的，让a_1分别送最好，x右边的，任意两个组合都比这两个单独让a_1送要快。现假设，让a_i和a_n一起，a_j和a_{n-1}一起，那么他们的时间花费为：a_n+a_{n-1}。如果让a_i和a_j一起，a_n和a_{n-1}一起，时间花费为$\max(a_i,a_j)+a_n\le a_n+a_{n-1}$。由此我们得出结论，让$a_n$、 a_{n-1}在一起一定优于让它们分别和其他的任意一头组合一起过河。

4.2 再来证明，为什么如果要用最快的两头牛帮助运送其他的牛?

如果每一头耗时最长的牛单独过，显然让最快的a_1 来送。然后要说明的是：如果是要送的两头耗时最长的牛要一起搭配渡河更优，那么也显然是借用最快的

a_1，a_2来往返过河最快。用类似的方法设用a_i和a_j进行船的调度，然后可以证明该选择不会比选a_1，a_2更好（只会更惨）。同时，选用a_1，a_2也可以使得$x=2a_2-a_1$的值尽可能的小，这样也就能让更多的a_i可以选择与其他更慢的牛一起渡河。

05 算法设计

```
Function cow(vector<int> nums, vector<int> speed )
{
    sort(speed,speed+nums);//将每头牛消耗时间进行sort排序
    int start =nums, ans=0;
    while (start)
    {
        if(start == 1){
                ans  += speed[0];
                break;}
            else if(start == 2){
                ans += speed[1];
                break;}
            else if(start == 3){
                ans += speed[2]+speed[0]+speed[1];
                break;}
            else{
                ans += min(speed[1]+speed[0]+speed[start-1]+speed[1],
                  speed[start-1]+2*speed[0]+speed[start-2]);
                  //比较两种方案最优
                start -= 2;}
            //将最慢的两头牛送到对岸后，再取倒数第三、第四慢
        }
        printf("%d\n", ans);}
    return 0;}
```

06 思考与改进

综上所述，证明了贪心策略的正确性，同时还发现，可以对该策略进行改进。改进如下：

每次判断最慢的两头牛一起运是不是要比分别由最快的送要快。如果是，让这两头牛一起过去，然后在继续循环判断，直到某一次，发现用最快的牛分别和其搭配要更快，然后之后所有的牛，都让最快的牛分别搭配（即采用方式一），一个一个的运。

07 作者感悟

完成上述问题后，我们可以清晰的理解到，使用贪心算法时切忌想当然。尽量从代数或者逻辑的角度进行证明，否则很有可能是错误的。描述贪心策略的时候一定要明确表明，你的贪心策略的局部最优解是什么，该局部最优解是否可以得到全局最优解。（上述改进后的贪心算法的局部最优解就是把“最慢”的两头牛送过去所需的最少时间）也就是要明确你的贪心方法做的每一步都是正确的。

在证明贪心策略的正确性的时候，有一个很技巧性的说法，想办法证明不会有什么选法比你的贪心策略的选法更优（在某种特定情况，某些特定选法可能和你的贪心策略一样优）。