Served，FCFS），则很难优化寻道时间。然而，当磁盘负载很重时，可以采用其他策略。很有可能当磁盘臂为一个请求寻道时，其他进程会产生其他磁盘请求。许多磁盘驱动程序都维护着一张表，该表按柱面号索引，每一柱面的未完成的请求组成一个链表，链表头存放在表的相应表目中。

给定这种数据结构，我们可以改进先来先服务调度算法。为了说明如何实现，考虑一个具有40个柱面的假想的磁盘。假设读柱面11上一个数据块的请求到达，当对柱面11的寻道正在进行时，又按顺序到达了对柱面1、36、16、34、9和12的请求，则让它们进入未完成的请求表，每一个柱面对应一个单独的链表。图5-28显示了这些请求。

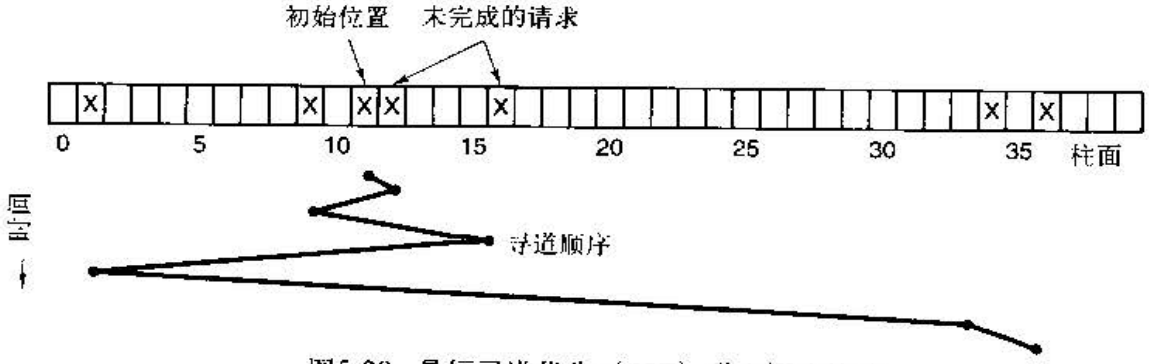

图5-28 最短寻道优先（SSF）磁盘调度算法

当前请求（请求柱面11）结束后，磁盘驱动程序要选择下一次处理哪一个请求。若使用FCFS算法，则首先选择柱面1，然后是36，以此类推。这个算法要求磁盘臂分别移动10、35、20、18、25和3个柱面，总共需要移动111个柱面。

另一种方法是下一次总是处理与磁头距离最近的请求以使寻道时间最小化。对于图5-28中给出的请求，选择请求的顺序如图5-28中下方的折线所示，依次为12、9、16、1、34和36。按照这个顺序，磁盘臂分别需要移动1、3、7、15、33和2个柱面，总共需要移动61个柱面。这个算法即*最短寻道优先*（Shortest Seek First，SSF），与FCFS算法相比，该算法的磁盘臂移动几乎减少了一半。

但是，SSF算法存在一个问题。假设当图5-28所示的请求正在处理时，不断地有其他请求到达。例如，磁盘臂移到柱面16以后，到达一个对柱面8的新请求，那么它的优先级将比柱面1要高。如果接着又到达了一个对柱面13的请求，磁盘臂将移到柱面13而不是柱面1。如果磁盘负载很重，那么大部分时间磁盘臂将停留在磁盘的中部区域，而两端极端区域的请求将不得不等待，直到负载中的统计波动使得中部区域没有请求为止。远离中部区域的请求得到的服务很差。因此获得最小响应时间的目标和公平性之间存在着冲突。

高层建筑也要进行这种权衡处理，高层建筑中的电梯调度问题和磁盘臂调度很相似。电梯请求不断地到来，随机地要求电梯到各个楼层（柱面）。控制电梯的计算机能够很容易地跟踪顾客按下请求按钮的顺序，并使用FCFS或者SSF为他们提供服务。

然而，大多数电梯使用一种不同的算法来协调效率和公平性这两个相互冲突的目标。电梯保持按一个方向移动，直到在那个方向上没有请求为止，然后改变方向。这个算法在磁盘世界和电梯世界都被称为*电梯算法*（elevator algorithm），它需要软件维护一个二进制位，即当前方向位：UP（向上）或是DOWN（向下）。当一个请求处理完之后，磁盘或电梯的驱动程序检查该位，如果是UP，磁盘臂或电梯舱移至下一个更高的未完成的请求。如果更高的位置没有未完成的请求，则方向位取反。当方向位设置为DOWN时，同时存在一个低位置的请求，则移向该位置。

图5-29显示了使用与图5-28相同的7个请求的电梯算法的情况。假设方向位初始为UP，则各柱面获得服务的顺序是12、16、34、36、9和1，磁盘臂分别移动1、4、18、2、27和8个柱面，总共移动60个柱面。在本例中，电梯算法比SSF还要稍微好一点，尽管通常它不如SSF。电梯算法的一个优良特性是对任意的一组给定请求，磁盘臂移动总次数的上界是固定的：正好是柱面数的两倍。

对这个算法稍加改进可以在响应时间上具有更小的变异（Teory，1972），方法是总是按相同的方向进行扫描。当处理完最高编号柱面上未完成的请求之后，磁盘臂移动到具有未完成的请求的最低编号的