7.11 Competitive Lists with Move-to-Front

Consider a list L containing n items, for example the collection of your favorite records. Whenever an item x in L is requested the list is scanned from the front until x is found. Therefore the cost of accessing x is k if x is the k^{th} item in the list. In order to better respond to subsequent requests, the position of any two adjacent items in L may be swapped. Such a swap also causes cost 1. Requests to items in the list L arrive in an on-line fashion.

The on-line algorithm Move-to-Front (M2F) adheres to the following simple rule: Whenever item x is requested, M2F moves x to the front. The cost to access x when x is the the k^{th} item in L is thus k for the initial scan, and $k-1$ swaps to move it to the front, i.e., the total cost is $2k-1$. Note that M2F does not change the relative order of items different from x. As usual, we would like to know how M2F compares to an optimal off-line algorithm OPT that knows the entire sequence of requests in advance. In the remainder of this section we establish the following theorem.

Theorem 7.16. *The algorithm Move-to-Front is strictly* 4*-competitive.*

Denote by OPT an optimal algorithm. We keep track of two lists L_{M2F} and L_{OPT}, i.e., the list L as it is maintained by M2F and OPT, correspondingly. Initially $L_{M2F} = L_{OPT} = L$. For the two lists L_{M2F} and L_{OPT}, an *inversion* is a pair of items (x, y) which appear in different order in L_{M2F} than in L_{OPT}.

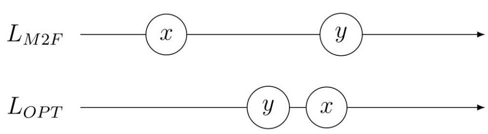

Figure 7.6: The inversion (x, y) between L_{M2F} and L_{OPT}.

Our competitive analysis of M2F is carried out using the *potential method.* The potential function Φ is defined as follows.

$$\Phi := 2 \cdot (\text{number of inversions between } L_{M2F} \text{ and } L_{OPT})$$

The potential method. A potential function Φ is a tool used in *amortized analysis.* The idea is to model the *amortized cost* amortized(*op*) of some operation *op* by

$$\text{amortized}(op) := \text{cost}(op) + \Delta\Phi(op),$$

where cost(*op*) is the *actual cost* of *op*, and $\Delta\Phi(op)$ is the change of potential caused by *op*. For the competitive analysis of an on-line algorithm $\mathcal{A}$, the total actual cost is bounded by $\mathcal{A}$'s the total amortized cost.

Initially the potential $\Phi = 0$ since the lists are equal. In every step, Φ is non-negative since the number of inversions is non-negative. Thus the total cost of M2F is upper bounded by the total amortized cost of M2F. It therefore suffices to show that M2F's amortized cost is at most 4 times the cost of OPT. We will in fact establish this bound after every request was handled, which implies that the bound also holds for the entire request sequence.

Fix a sequence of requests and a request r in that sequence, and denote by x the item requested by r. Denote by j and k the position of x in L_{OPT} and L_{M2F} before handling r, respectively.

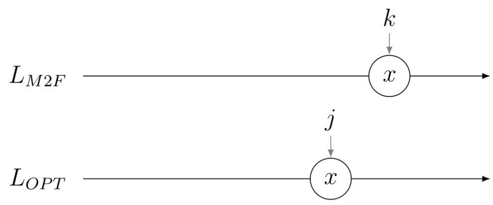

Figure 7.7: Item x in L_{M2F} and L_{OPT} before handling request r.

The cost amortized(r) for M2F consists of the actual cost(r) and the change in the potential function $\Delta\Phi(r)$. Recall that cost(r) $= 2k - 1$. The change of potential is completely determined by the inversions that are created or destroyed by the list maintenance performed by M2F and OPT, in other words $\Delta\Phi(r) = \Delta\Phi_{M2F} + \Delta\Phi_{OPT}$.

Let us first look at the contribution $\Delta\Phi_{M2F}$ to $\Delta\Phi$ caused by M2F's list maintenance. Since M2F does not change the relative order of non-requested items, all affected inversions must involve item x. Furthermore x is only swapped with items y that precede x in L_{M2F}. Let y be an item preceding x in L_{M2F} before M2F's list maintenance. We say that item y is *bad* if y precedes x also in L_{OPT}, otherwise y is *good*. If y is bad, then a new inversion is created, otherwise an inversion is destroyed. There are at most $j-1$ bad items, and therefore at least $(k-1)-(j-1)$ good items. Recalling that Φ counts each inversion twice, we conclude that

$$\Delta\Phi_{M2F} \leq 2 \cdot \Big(j - 1 - \big((k-1) - (j-1)\big)\Big) = 4j - 2k - 2.$$

We still need to account for the list maintenance of OPT. Denote by s the number of swap-operations performed by OPT while handling request r. Every such swap increases $\mathrm{cost}_{OPT}(r)$ of the optimal algorithm by exactly 1. Recall that the cost for finding item x in L_{OPT} is j, and therefore

$$\mathrm{cost}_{OPT}(t) = j + s$$

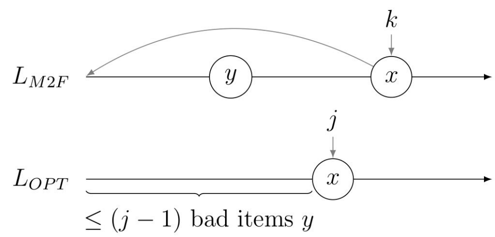

Figure 7.8: Items x, y in L_{M2F} and L_{OPT} before handling request r.

Furthermore, every swap performed by OPT creates at most one new inversion. The contribution $\Delta\Phi_{OPT}$ to $\Delta\Phi$ is thus at most $2s$, and we can bound amortized(r) as

$$\begin{aligned}
\text{amortized}(r) &= \text{cost}(r) + \Delta\Phi_{M2F} + \Delta\Phi_{OPT} \\
&\leq 2k - 1 + 4j - 2k - 2 + 2s \\
&= 4j - 3 + 2s \\
&< 4j + 2s \\
&\leq 4 \cdot (j + s) = 4 \cdot \text{cost}_{OPT}(r).
\end{aligned}$$

$\square$