Pan Zhenyang(A0078075X)

Summary

The article is mainly divided into three sections. Section one is to analyze the amortized efficiency of the move-to-front list update rule. In this section, the cost of accessing and deleting the i-th element is O (i). For inserting an element into the list with a total number of i elements, the cost is (i+1). Theorem1 is Cmf(s) <= 2Ca(s) + Xa(s) – Fa(s) – m. To prove it, he uses a potential function to estimate the differences between running the MF(move-to-front) and Algorithms A(can be any Algorithms). To access an element, we need i steps. After accessing it, there will be i-1 inversions to move it in front compared with A. The sum will be 2\*i – 1. If there is total m operations and the total cost is noted as C, we will have Cmf(s) <= 2Ca(s) + Xa(s) – Fa(s) – m, where Xa(s) is the paid exchange and Fa(s) is the free exchange. If we move an accessed or inserted element at position k at least k/d-1 units closer to front, we will have Cmf(d)(s) <= d(2Ca(s) + Xa(s) – Fa(s) – m). This is theorem2 and the proof is same as theorem1. In the second section, the article explores the limitation of theorem1. Theorem3 claims that A is a sequence s of insertions and accesses and there is another algorithm which is no more expensive than A and does not have no paid exchanges. Let i and i+1 be the elements to exchange and j be element accessed. If j does not belong to i and i+1, exchange can be moved after accessing without changing the cost. If j = i, Δf(i) will be saved on the accessing. If j = i+1, it can cost extraΔf(i), but it can perform exchange for free, saving theΔf(i). Theorem4 provides a situation the function is convex. We will have Cmf(s) <= 2Ca(s) + Xa(s) – mf(1). The proof is like theorem1; in addition we have to substitute them in the convex function f. In the last section, it is about paging. Theorem5 is Fa(s) >= (Na/(Na-Nmin+1))Fmin(s). We have that A makes Na faults while min makes Na-Nmin+1 faults. The first Na-Nmin+1 accesses are to pages in neither A’s nor MIN’s fast memory. Suppose we have a set S of Na+1 pages which either in Min or Na-Nmin+1. The combined sequence of Na accesses, A will fault every time. If this repeats many times, we will have theorem5. Theorem6 is Flru(s) <= (Nlru/(N/lru-Nmin+1))Fmin(s) +Nmin. The proof is constructing a model with three cases and using the partition to get the result.

Comments:

After reading through the paper, I still not quite sure whether move-to-front is more efficient than any other algorithms for an unsorted list. I followed the proof step by step and found there is nothing wrong. However, I still cannot get myself convinced. Maybe my initial idea is just counting the frequency of every element in the list and makes advantages of this. Then, after reading through it, I lose my idea. When I try to understand the move-to-front in the proof in details, I also lose the whole view of the algorithm. Anyway, I think if I have a large number of random real data, I will code these algorithms and figure out the real time of running them. In that way, I will get myself convinced. On the other hand, the move-to-front is really simple algorithm even a beginner can implement it. But it has lots of advantages compared to those complex ones. So, research problems should not only come from the area full of mysteries, but also the area with simple knowledge.