Discussion 5

Greedy Algorithms: Off-line caching (Textbook Problem 16-5)

Modern computers use a cache to store a small amount of data in a fast memory. When a computer program executes, it makes a sequence $\langle r_1, r_2, ..., r_n \rangle$ of n memory requests, where each request is for a particular data element. For example, a program that accesses 4 distinct elements $\{a, b, c, d\}$ might make the sequence of requests $\langle d, b, d, b, d, a, c, d, b, a, c, b \rangle$. Let k be the size of the cache. When the cache contains k elements and the program requests the $(k+1)$st element, the system must decide, for this and each subsequent request, which k elements to keep in the cache. In other words, **cache hit** and **cache miss** will happen. The cache replacement algorithm evicts data with the goal of *minimizing the number of cache misses* over the entire sequence of requests.

Typically, caching is an on-line problem. That is, we have to make decisions about which data to keep in the cache without knowing the future requests. Here, however, we consider the off-line version of this problem, in which we are given in advance the entire sequence of n requests and the cache size k.

We can solve this off-line problem by a greedy strategy called **furthest-in-future**, which chooses to evict the item in the cache whose next access in the request sequence comes furthest in the future.

(a) Write pseudocode for a cache manager that uses furthest-in-future strategy. Assume that request sequence is of size n and cache size is k.

(b) What is the running time of this algorithm in terms of n, k.

(c) Justify its optimality.
Hint: Greedy choice property and optimal substructure property

```
FURTHEST-IN-FUTURE(R, C)
for i = 1 to R.length
    if R[i] ∈ C
        Cache Hit
    else
        Cache Miss
        if Cache is not full
            add R[i] to Cache
        else
            // tmp is used to keep track of the latest
            // appearance of cache item in the sequence
            tmp = i
            // pos is used to keep track of
            // the position in cache to be replaced
            pos = 1
            for j = 1 to C.length
                p = i
                // find the first appearance of the specified
                // cache item in the sequence
                while ______________
                    p = p + 1
                // if p reaches the end of sequence without
                // finding the specified cache item
                if ______________
                    // p = Infinity
                    pos = j
                    break from the inner for loop
                // update tmp
                if ______________
                    ______________
                    ______________
            C[pos] = R[i]
```

See slides 48 - 51 for more detailed analysis of optimality at: `http://www.cs.princeton.edu/~wayne/kleinberg-tardos/pdf/04GreedyAlgorithmsI.pdf`