Report

1. The subproblem of this greedy algorithm is to keep the most useful data in the cache. We are trying to optimize the hit ratio in order to decrease the miss ratio.

2. We are calculating furthest distances for all separate requests. When we are creating this array, the time complexity depends on both n and m. O(n*m)

On the second part, the complexity is O(n) so we can sum them up n*m + n = O(n*m).

3. In this algorithm, we are dealing with elements array and its' nodes. If we increase the number of elements higher than k + 1, at some point, elements will be missed too much.

a) Elements [0, 1, 2, 3, 4, 5], capacity: 4, requests: [5, 4, 3, 2, 1, 0, 5, 4, 3, 2, 1, 0]

When the cache checks 5 in itself, there will be a miss, so it will remove 4.

When the cache checks 4 in itself, there will be a miss, so it will remove 3.

…

When the cache checks 0 in itself, there will be a miss, so it will remove 5.

When the cache checks 5 in itself, there will be a miss, so it will remove 4.

Misses will occur infinitely!

b) We can use a different algorithm such as FIFO (First In First Out) or LRU (Least Recently Used). LRU is better in general which tracks all elements' occurrence in requests.