Lecture 21
Caching

30 September 2016

1. Introduction

Consider a scenario in which John have all 26 English alphabets and each alphabet corresponds to an Encyclopedia book. So John have 26 different Encyclopedia books and each starting from different English alphabets. He has a friend named Bob, each time Bob come and ask for an Encyclopedia book (remember Bob will only say first letter of Encyclopedia Book's name) to John. Since John can't keep all 26 Encyclopedia books with him all the time, so he keeps only 5 Encyclopedia books with him at a time and rest all to a safe place which he can access any time but accessing the rest of books take more time then his Encyclopedia books he have in his pocket.

Bob will come to John every day of October month starting from 1st to 30th October and ask for One of the Encyclopedia book (remember Bob is not keeping the book he is asking for, book still remains with John only). John is free to keep any 5 books with him in his pocket.

Say John knows what Bob is going to ask in whole month before-hand ie.. before begining of October John knows for which book will Bob ask for on 1st ,2nd and so on till 30th October.

Consider one example,
John starts with following 5 books in his pocket :

A	E	O	U	P

List of books Bob is going to ask with date (Remember this is known to John) :

A	C	U	I	H	H	N	M	V	C	R	T	Y	U	I	O	P	O	Y	R
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

S	X	V	B	M	O	I	Y	R	G
21	22	23	24	25	26	27	28	29	30

We need to find the strategy by which John can decide which books to keep with him in his pocket, so that time taken by John to full-fill Bob's all query (query is just the book for which Bob ask John to show) is minimum (remember the time taken to access books which are not in John's Pocket is greater than the books which are in his pockets) ?

First strategy can be **FIF(Farthest In Future)** in which, whenever Bob asks John for a book which is not in his pocket he removes that book from his pocket which is farthest in list of books which Bob is going to ask in future and insert the currently asked book into his pocket.

Second Strategy can be **LRU(Least Recently Used)** in which, whenever Bob asks John for a book which is not in his pocket he removes that book from his pocket which has been unused for the longest time and insert currently asked book into his pocket ie.. Here John is assuming that that books that have been heavily used in the last few days will probably be heavily used again in the next few days. Conversely, books that have not been used for days will probably remain unused for a long time.

Third Strategy can be **LFU(Least Frequently Used)** in which, whenever Bob asks John for a book which is not in his pocket he removes that book from his pocket which is the least frequently used and insert currently asked book into his pocket.

It is left to reader as exercise to show that our first strategy ie.. FIF (Farthest In Future) is the most optimum strategy for this given problem.

2. Terminology

Revisit our John and Bob first example,

Consider the situation in which Bob asks John for a book and that Book is present in John's Pocket then it is called **Cache Hit.**

Consider the situation in which Bob asks John for a book and that book is not present in John's pocket then it is called **Cache Miss.**

Consider the situation in which John wants to remove a book from his pocket and replace it with some other book then this removal of book is called **Cache Eviction.**

Consider the situation in which John can keep only 5 Books with him in his pocket, in general say he can keep k books in his pockets then k is called **Cache Size.**

Online Algorithm is an algorithm in which we assume future is not visible or more precisely complete input is not available and is given serially with time.

Comparator Ratio (COMPRAT) is ratio of any algorithm's performance to most optimum algorithm's performance. Say COMPRAT(algo1) = 2, it means time taken by algo1 will be at most twice of time taken by most optimum algorithm.

3. Points to ponder

Consider the another variant of first example of John and Bob in which Bob can see what is in John's Pocket but John have no idea for which book Bob will ask him tomorrow.

In this situation what-ever Online algorithm John follows, Bob can still make sure each time John have a cache miss !!

Let what Bob ask John each day be element of an n length string (S) contained by Bob (remember here Bob can decide each element of string at time just before he asks John for a book)

Here Bob is smart and he will always ask for a book which is not present in John's Pocket to make sure cache miss each time !!

Let Online algorithm followed by John be represented by OALG.
Hence OALG(S) = | S |
OALG(S) = n (Since | S | = n)

Consider the John and Bob first example once again in more generalised way, say number of day now extend to n and John can keep at most k books in his pockets.

What alphabets Bob should fix (more precisely in what order) for each day so that John will have maximum cache miss, given that John is smart and following most optimum strategy ie… FIF (Farthest in Future) ? (Remember it's not Online Algorithm)

It is left to reader as an exercise to show that any Online algorithm is of comparator ratio not more than k, where k is size of cache.