E0 215 : Homework 1

Deadline : 30th August, 2022, 2pm

Instructions

- Please write your answers using LaTeX. Handwritten answers will not be accepted.
- You are forbidden from consulting the internet. You are strongly encouraged to work on the problems on your own.
- You may discuss these problems with others. However, you must write your own solutions and list your collaborators for each problem. Otherwise, it will be considered as plagiarism.
- Academic dishonesty/plagiarism will be dealt with severe punishment. Cases of academic dishonesty/plagiarism will be reported to the appropriate authorities.
- Late submissions are accepted only with prior approval (on or before the day of posting of HW) or medical certificate.

1. Anomaly for Paging: In class, we mentioned that an anomaly can happen for some paging algorithms, where on some input sequences the algorithm may perform better (i.e., incur less number of page faults) when it has a smaller fast memory. Show that LRU (Least-Recently-Used) does not incur this anomaly but FIFO (First-In-First-Out) does incur the anomaly.

2. Marking and Conservative Algorithms: An algorithm ALG is *conservative* if on any consecutive input subsequence containing k or fewer distinct page references, ALG incurs k or fewer page faults. Show that FIFO is not a marking algorithm but it is a conservative algorithm.
   Now consider the algorithm Flush-When-Full (FWF): Whenever there is a page fault and there is no space left in the cache, evict all pages currently in the cache. Show that FWF is not a conservative algorithm. However, it is a marking algorithm.

3. Mark: Prove that the algorithm Mark is H_k-competitive against an oblivious adversary when N, the total number of pages in the slow memory, is $k+1$.
   Show that, however, in general Mark is not H_k-competitive (*Hint:* construct an example for $k = 2, N = 4$).

4. pathcow:
   I) Consider the following algorithm for path-cow problem. The cow starts at the origin, moves $x = 1$ unit to the right. If the target is not found, the cow comes back to the origin and goes $x = 1$ unit to the left. If the target is not found, the cow comes back to the origin and repeats this procedure with $x = 2, 4, \ldots, 2^i, \ldots$ until the target is found.

   a) Assume the cow finds the hole at distance u from the origin on its left. Assume the largest power of 2 which is smaller than u is 2^k. What is the total distance moved by the cow?

   b) Where does the adversary place the hole in order to harm the algorithm?

   c) What is the competitive ratio of this algorithm?

II) In part I we assumed the algorithm is deterministic and the first move is to the right. Consider the same algorithm in which the first move is randomly selected to be to the right or left (each with a probability of 1/2). What is the competitive ratio of this randomized algorithm?
III) Assume instead of a path, we have a *binary tree* with each edge having a length of 1. Originally, the cow is located at the root. First, she moves to the left child; if the target is not found, she goes back to the root and then moves to the right child. If the target is still not found, she returns to the root and goes to visit all nodes of depth 2 on the left (i.e., at distance 2 from the root on the left). After checking these nodes, the cow returns to the root and repeats the same for nodes of depth 2 in the right side of the root. This procedure is repeated until at some point the target is found. Assume the cow uses depth first strategy to check nodes of depth i. In a nutshell, the algorithm works by working in rounds, where at round i it visits all vertices of depth i.
What is the competitive ratio of this algorithm? To answer, assume the target is at depth k and write the competitive ratio in terms of k. As before, you have to indicate where the adversary places the target and deduce the competitive ratio accordingly.

5. ONLINE BIDDING: We saw in the class that a simple doubling approach gives the best competitive ratio that a deterministic online bidding algorithm can achieve. That ratio was 4. In this problem, we examine the power of advice (some extra information) and randomization for this problem; basically we want to show that *advice can be stronger than randomization*. Consider two deterministic algorithms ALG1 and ALG2 , where ALG1 guesses are $1, 4, 16, \ldots 4^i$ and ALG2 guesses are $2, 8, 32, \ldots, 2 \cdot 4^i$.

   a) Consider an algorithm that flips a fair coin at the beginning and randomly chooses between ALG1 and ALG2 , and uses the guesses of the selected algorithm. What is the competitive ratio of this randomized algorithm?

   b) Assume an algorithm that receives 1 bit of advice as follows. For each instance of the problem the advice bit indicates the algorithm which has smaller cost between ALG1 and ALG2 . What is the competitive ratio of the algorithm with 1 bit of advice?