Advanced Algorithms 2020 18 Dec 2020

Exercise 13

Lecturer: Mohsen Ghaffari *Teaching Assistant: Vaclav Rozhon*

1 Warmup: Hungry Cow again

Recall the Hungry Cow problem from the previous session. Prove that any deterministic algorithm for the problem gives approximation guarantee at least 3 and any randomized algorithm for the problem has approximation guarantee at least 2.

2 Online Bipartite Matching [Recommended]

In the Online Bipartite Matching problem, we are given one side U of a bipartite graph $G = (U \cup V, E)$ and the vertices of the other side V of the bipartition arrive in an online fashion, one by one. Each time a vertex arrives, it arrives together with all its incident edges and we have to decide immediately and irrevocably which of its incident edges we select into the matching or whether we do not select any of the incident edges.

(A) Determine the competitive ratio of deterministic online bipartite matching, i.e., prove tight upper and lower bounds.

Hint:
Try to play with a path with three edges. Also, the size of any maximal matching is at least half of the size of the maximum matching.

(B) Adapt your lower bound construction so that it gives a (weaker) lower bound for randomized algorithms with an oblivious adversary, too.

B+: Can you find a construction that yields at least a slightly better lower bound?

Hint:
Use Yao's principle and a path with three edges. To do better, try to find a graph with three vertices on the left and three vertices on the right.

3 Lower Bound for the k-Server Problem on the Line [Recommended]

Argue that every deterministic online algorithm for the k-server problem on the line has competitive ratio at least k.

Hint 1:
What is the smallest graph (that is a path) on which you can hope to build sufficiently bad sequences of requests?

Hint 2:
For the natural input sequence of always choosing the next request to be the (only) vertex in your path that does not host a server: How often is each edge of your path traversed? (On average?) How could one use this?

Hint 3:

Can you obtain an upper bound on the cost incurred by OPT by finding a good offline algorithm that moves only one server (after all other servers are moved where they should be)? Depending on the request sequence, which server would you choose to be the moving server?

4 Hiring Professors

Due to the increasing demand for a special course on online algorithms, ETH decides to hire a new professor covering this topic. As expected, the candidates arrive in an online fashion, one after the other, and whenever a candidate arrives, the hiring committee decides immediately and irrevocably whether the candidate is hired. The only information the hiring committee determines about an arriving candidate is whether the candidate is better than all previous candidates, or not. After all, ETH only wants to hire the very best. For the same reason, ETH also hires at most one of the candidates.

The number of candidates is n and known in advance. Show that any randomized online algorithm hires the best candidate with a probability of at most $1/n$ (on the worst-case sequence of candidates for the respective algorithm).

Hint 1:

Use Yao's Principle!

Hint 2:

It might be useful to only consider input sequences where, until the best candidate (which may not be the last) is reached, the quality of candidates increases monotonically.