Graded Homework 2, exercise 2

Marco Milanta

December 4, 2021

Maximizing Profit (25 points)

Given a stream of n elements $x_1, \cdots, x_n \in \{1, \cdots, n\}$ all distinct (i.e., the input is a permutation). We have to choose x_i, x_j with $j \geq i$ on arrival, meaning that when an element arrives, we must immediately decide if we include it in one of the two elements. The profit function is given as $x_j - x_i + 1$. (Note that any sensible strategy will have $x_j - x_i + 1 > 0$ as we can choose $i = j = n$.)

1. Find a deterministic algorithm that achieves competitive ratio at most $\sqrt{n}$.
2. Show that any randomized algorithm has competitive ratio at least $\Omega(\sqrt{n})$

Solutions

1. We propose the following algorithm. Note that we index arrays starting from 1.

Algorithm 1: Online algorithm

Input: $X \in$ permutations of $\{1, \ldots, n\}$
Output: $i, j, x_j - x_i + 1$

```
Seen ← [0, ..., 0] ∈ ℕ^n
Missing ← [1, ..., n] ∈ ℕ^n
ChosenElements ← 0
x_i ← x_j ← 0
for k ∈ 1, ..., n − 1 do
    if ChosenElements = 0 then
        Missing[X[k]] ← 0
        optFutu ← max Missing
        if (optFutu − X[k] + 1) ≥ √n then
            x_i ← X[k]
            i ← k
            ChosenElements ← 1
    else if ChosenElements = 1 then
        if X[k] = max Missing then
            x_j ← X[k]
            j ← k
            ChosenElements ← 2
if ChosenElements = 0 then
    x_i ← x_j ← X[n]
    i ← j ← n
return i, j, x_j − x_i + 1
```

Legality of the algorithm. Before analyzing the competitive ratio of this algorithm, I would like to notice that it has the correct structure. Notice these two facts:

- In the $k-$th iteration of the loop, we only look at $X[k]$. This means that our algorithm doesn't cheat by looking into the future. Finally, we look at the last element after the loop.
- x_i and x_j are assigned only once, always x_i before x_j, and they are assigned to be $X[k]$ at the $k-$th iteration. Or, at latest, after the for loop. All of this is guaranteed by the counter `ChosenElements`. This means that we are respecting the constraint "when an element arrives, we must immediately decide if we include it in one of the two elements".

Competitiveness of the algorithm. Now we want to analyze the performance. To do so, let $\hat{i}, \hat{j}, \hat{i} \leq \hat{j}$ be an optimal solution that maximize, $x_{\hat{j}} - x_{\hat{i}} + 1$.

Theorem 1 (Algorithm 1 is $\sqrt{n}$-competitive)**.** *The algorithm 1 is $\sqrt{n}$-competitive against the optimal solution. Or, formally, let $i, j,$ `profit` be the output of the algorithm 1. Let OPT be the maximal profit, then, for any input*

$$\texttt{profit}\sqrt{n} \geq OPT.$$

Proof. We divide the proof in two parts depending on the behavior of the algorithm.

Case 1: (there is a $k \in 1, \ldots, n-1$ for which we enter the **if** at line 9) Let for now k be the iteration in which we enter the **if** at line 9. One can notice that we will end up with a final cost of $\texttt{optFutu} - X[k] + 1$. This is because we chose $x_i = X[k]$, and then we chose x_j to be the maximal value among the remaining ones. We can safely find such value since we are keeping track of the missing ones (`optFutu`). Once $\max \texttt{Missing}$ will arrive, we pick it for x_j. Notice that we cannot miss it since we are only looking at permutations as inputs.

Using this, we get that

$$\texttt{profit}\sqrt{n} = (\texttt{optFutu} - X[k] + 1)\sqrt{n} \overset{(i)}{\geq} \sqrt{n}\sqrt{n} = n \overset{(ii)}{\geq} OPT.$$

Where (i) follows from the fact that we enter the **if** at line 9, and (ii) follows from the fact that maximum possible profit is n. In this scenario we have shown that the algorithm is $\sqrt{n}$ competitive .

Case 2: (we never enter the **if** at line 9) In this scenario, our algorithm picks $x_i = x_j = n$, and our `profit` will be 1. We now want to show that $OPT \leq \sqrt{n}$. To do so, we continue by contradiction. Assume that there are $\hat{i} \leq \hat{j}$ such that $x_{\hat{j}} - x_{\hat{i}} + 1 > \sqrt{n}$. Our algorithm gets to step $k = \hat{i}$. Since $\hat{j} > \hat{i}$ (if $\hat{j} = \hat{i}$ then the optimal cost would be $1 < \sqrt{n}$). In this step $\texttt{optFutu} = \max \texttt{Missing} \geq x_j$, but this yields that $\texttt{optFutu} - X[\hat{i}] + 1 \geq x_j - x_i + 1 \geq \sqrt{n}$, which itself it yields that we enter the **if** at line 9. This is however a contradiction.

From this we have that

$$\texttt{profit}\sqrt{n} = \sqrt{n} \geq OPT.$$

Conclusion: We have shown that independently of which part of the algorithm triggers, we can guarantee that it outputs a $\sqrt{n}$-competitive solution □

2. Show that any randomized algorithm has competitive ratio at least $\Omega(\sqrt{n})$

Solutions

1. To make the solution easier to read we define some notation:
    - Let $G = (V, E, w)$ be our weighted graph, $w : E \to \mathbb{R}^+$