Comp 7720 - Online Algorithms

Final Exam

Shahin Kamalli
University of Manitoba - Fall 2017

December 7, 2017

‘Difficult roads often lead to beautiful destinations’  unknown

Write your name here (no need for student id, etc.)  Homer Simpson

• Do not open this booklet until instructed.

• It is an open-book exam. Your can use any printed/written material from the course. You are not allowed to use laptops/cell-phones. Please turn off your cell phones and put them in your bags.

• Manage your time. We start the exam at 10:00 and end the exam at 10:15. You have 75 minutes.

• In order to save trees, exam is printed double-sided. There are 7 pages (including this cover page). Write your answers in the provided boxes.

• In the unlikely case that you find the exam too long/hard, do not panic. The marks will be scaled so that the highest mark gets the full mark.

• If you are not sure about the answer to a true/false question, work on other question and get back to it at the end. Do not leave true/false questions blank as there is no penalty for wrong answers.

• Do not waste your time writing lengthy answers. You can be succinct and yet precise. The provided space for each question is estimated to be 3 times more than the required space. Note that your time is limited.

• There are more important things in life than this exam. So, relax and smile. Also, there are more important components to this course than this exam. So, relax further and smile more (but still manage your time while relaxing).
Problem 1  True/False Questions [12 marks]

Indicate whether any of the following statements is true or false. There is no need to justify your answers.

- One bit of advice is sufficient to achieve an optimal solution for the cow-path problem  True  False
  **Answer:** It is true; with one bit you can indicate whether the target is on the left or on the right, and the cow just needs to go in that direction until it finds the target, moving an optimal distance.

- An online bidding algorithm in which bids are powers of 1.5 has a better competitive ratio than the doubling algorithm True  False
  **Answer:** It is false. As mentioned in the class (slide 7 of lecture 3), the competitive ratio 4 of the doubling algorithm is the best that a deterministic algorithm can achieve, i.e., no other deterministic algorithm, in particular the one that guesses powers of 1.5, can have a better competitive ratio. In Assignment 2-part(a), you indeed computed the competitive ratio with powers of \(\gamma\) as \(\frac{\gamma^2}{\gamma - 1}\), which becomes 4.5 for \(\gamma = 1.5\).

- No randomized list-update algorithm has a competitive ratio better than that of Move-To-Front True  False
  **Answer:** It is false. We saw a few randomized algorithms in the class which have competitive ratios strictly better than 2 of MTF. For example, in slide 10 of lecture 6 we saw BIT has a competitive ratio of 1.75.

- Consider two bin packing algorithms A and B. If competitive ratio of A is better than B then the average-case performance of B is better than A. True  False  **Answer:** It is false. In classes 17 and 18, it was repeatedly mentioned that there is not necessarily a trade-off between competitive ratio and average-case performance (e.g., slide 12 of lecture 18). In particular we saw algorithms like Harmonic-Match (algorithm A) which has a better competitive ratio than Best Fit (algorithm B) while its average-case performance is no worse than Best Fit.

- If there is an online clustering algorithm with competitive ratio \(c\) then there is a bidding algorithm with competitive ratio \(c/2\). True  False  **Answer:** It is false. Note that in the class, we proved that if there is a bidding algorithm with competitive ratio \(c/2\), then there is a clustering algorithm with competitive ratio \(c\). However, the other direction is not true: there might be clustering algorithms which do not use bidding as ‘black-boxes’. As mentioned in the last class (as an answer to a question in the sample exam), there are clustering algorithms with competitive ratio strictly better than 8. If the above statement was true, that meant there was a bidding algorithm with competitive ratio better than 4, which we know is not true.

- The following is a valid solution for fault-tolerant packing of an input sequence. True  False  **Answer:** It is false; failing of \(S_1\) causes an extra load of \(0.3(a) + 0.1(d)\) to \(S_2\) and the total load of items in \(S_2\) will be \((0.3 + 0.25 + 0.1) + (0.3 + 0.1) = 1.05\) which implies there is an overflow at \(S_2\).
Problem 2  Splay Trees [10 marks]

Apply the splay operation on the following splay tree when there is a request to node ‘15’. Show your steps.

Answer: Here it is:
Problem 3  Work-Function Algorithm [10 marks]

Consider the 2-server problem on the following metric of size \( m = 4 \).

Assume before serving the \( t \)th request, the servers are located at nodes \( a \) and \( b \). The values of work-function at time \( t \) are computed as shown in the following table. Indicate how the algorithm serves the request at time \( t \).

<table>
<thead>
<tr>
<th>Configuration</th>
<th>( w_t(X) + d(C, X) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a,b )</td>
<td>12</td>
</tr>
<tr>
<td>( a,c )</td>
<td>10</td>
</tr>
<tr>
<td>( a,d )</td>
<td>11</td>
</tr>
<tr>
<td>( b,c )</td>
<td>10</td>
</tr>
<tr>
<td>( b,d )</td>
<td>11</td>
</tr>
<tr>
<td>( c,d )</td>
<td>11</td>
</tr>
</tbody>
</table>

Answer:

Recall that work-function algorithm had three steps for serving request at time \( t \). First, we had to find the values of work-function for all configurations at time \( t \). This step is already done in this problem (hence, we do not really need to worry about what the request at time \( t \) was since it is reflected in the values of work-function given in the above table).

The second step is to find configuration \( X \) with minimum value of \( w_t(X) + d(C, X) \) where \( C \) is the current configuration \( (a, b) \) (see slide 6 of lecture 13). These values can be computed as follows:

\[
egin{align*}
  w_t(a, b) + d((a, b), (a, b)) &= 12 + 0 = 12 \\
  w_t(a, c) + d((a, b), (a, c)) &= 10 + 2 = 12 \\
  w_t(a, d) + d((a, b), (a, d)) &= 11 + 1 = 12 \\
  w_t(b, c) + d((a, b), (b, c)) &= 10 + 1 = 11 \\
  w_t(b, d) + d((a, b), (b, d)) &= 11 + 2 = 13 \\
  w_t(c, d) + d((a, b), (c, d)) &= 11 + 2 = 13 
\end{align*}
\]

The third and final step is to move servers to configuration that minimizes the above values. In this example, that configuration is \( (b, c) \). So, the algorithm moves server \( S_1 \) from \( a \) to \( c \).
Problem 4  Alternative Algorithm for Bin Packing [10 marks]

Consider a bin packing algorithm named FF-Har, which packs items of size larger than 1/3 using First Fit and separately from other items. For items of size smaller than or equal to 1/3, the algorithm uses Harmonic algorithm with parameter $K = 20$.

a) Show the final packing of the algorithm for sequence $\sigma = \langle 0.21, 0.35, 0.3, 0.7, 0.22, 0.25, 0.65 \rangle$ (the intermediate packings are not required).

**Answer:**

Here it is:

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.22</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.21</td>
<td>0.3</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.35</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.65</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0.7</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

b) Verify whether the following statement is correct or incorrect. Briefly justify your answer in one or two sentences.

Statement: Competitive ratio of FF-Har is no more than competitive ratio of Harmonic.

**Answer:**

It is true because the only difference in FF-Har and Harmonic is in the packing of items larger than 1/3: FF-Har might place an item of class 1 (larger than 1/2) together with an item of size larger than 1/3. Regardless, if we use the same weighting function argument that we used for Harmonic (see slides 8-11 of lecture 16), we can prove the competitive ratio of FF-Har is no more than Harmonic. Steps 1 and 3 of the weighting argument will be the same. In step 2, where we prove the weight of any bin of FF-Har is at least 1, we note that weight of bins which include an item of size larger than 1/2 might be more than 1 (when they include another item of size larger than 1/3). Regardless, the weights will be at least one and the same competitive ratio as Harmonic can be derived for FF-Har.
Problem 5  Advice [10 marks]

Consider a variant model for list-update in which paid-exchanges are forbidden, i.e., after accessing an item, the list can be modified only by using a free exchange (i.e., moving the accessed item closer to the front of the list). Show that in this model, for a sequence of \( n \) requests and a list of length \( m \), advice of size \( O(n \log m) \) is sufficient to achieve an optimal solution. Follow the four steps to describe an algorithm with advice.

**Answer:**

We follow the four steps to define an algorithm with advice with the desired property:

- **What does advice encode?** Assume \( \text{Opt} \) accesses an item \( x \) at index \( j \) of the list. Since \( \text{Opt} \) can only use a free exchange to move the requested item \( x \) to index \( i \leq j \) of the list, the advice can encode the index \( i \). In other words, advice indicates the index of the accessed item after \( \text{Opt} \) has accessed and potentially moved it closer to the front.

- **What is the size of advice?** There are \( m \) possible indices for each accessed item after \( \text{Opt} \) reorganizes the list. Hence, to encode the index as discussed above we need \( O(\log m) \) bits of advice per request. For a total of \( n \) requests, it will be \( O(n \log m) \) bits.

- **How the algorithm works with this advice:** The algorithm simply mimics \( \text{Opt} \). If \( \text{Opt} \) accesses an item at index \( j \) and moves it to index \( i \), the algorithm also accesses the item at index \( j \) and uses the advice to move it to index \( i \).

- **Why this is the optimal algorithm?** The lists maintained by the algorithm and \( \text{Opt} \) will be the same at any given time. Hence, the access costs will be the same and consequently both algorithms will have same costs.
Problem 6 Adversarial Arguments [10 marks]
Consider the Most-Recently-Used (MRU) algorithm for paging which upon a fault evicts the most recently used page. Assume the cache is initially empty. Provide an adversarial sequence to show MRU is not competitive.

Answer:
Assume the size of the cache is $k$. Consider the following sequence:

$$\sigma = (a_1, a_2, \ldots, a_{k-1}, (a_k, a_{k+1})^m)$$

On the first $k$ requests, the algorithm brings $a_1, \ldots, a_k$ to the cache. On the request to $a_{k+1}$, it evicts the most recently used page which is $a_k$. On the next access to $a_k$, it evicts the most recently used page which is $a_{k+1}$ and this continues. The cost of the algorithm for $\sigma$ will be $k - 1 + 2m$. On the other hand, Opt brings $a_1, \ldots, a_k$ to the cache. On the first fault at $a_{k+1}$, it evicts any page except $a_k$. After that, $a_k$ and $a_{k+1}$ will be in the cache and no further fault happens. So the cost of Opt for $\sigma$ is $k + 1$. We conclude that the competitive ratio of MRU is at least $\frac{k - 1 + 2m}{k + 1}$ which grows with the input length. Hence, the algorithm is not competitive.