Assignment 2: Compression, Splay Trees, Caching, and \( k \)-server

Shahin Kamalli
University of Manitoba - Fall 2019

Due: October 14th at 11:59 pm

“Some people talk to animals. Not many listen though. That’s the problem ...”
Winnie-the-Pooh (A. A. MILNE)

Happy October 4th, World Animal Day

Please pay attention to the followings when preparing/submitting your assignment:

- All problems are written problems. There are six problems with a total of 55 marks. The last problem is a bonus problem (your assignment will be marked out of 45). The bonus problem is harder than the rest and will be marked in a different way. Approach it only after you finished other questions.

- If you have any question related to the assignment, you are encouraged to post it on Piazza. Note that you can submit anonymously. Also, instead of emailing me, you can always write a private note in Piazza. It is likely that I drop hints when a question is posted publicly on Piazza (because all students can benefit from it). It is not the case when you ask questions in emails or during office hours.

- You are welcome to discuss the problems with your friends (or enemies). But you should write your answers individually. You might be interviewed about your answers. Be careful not to accidentally copy.

- Submit your answers electronically using Crowdmark. You should answers for different questions separately.
Problem 1  Compression [10 marks]
a) Apply the Burrows-Wheeler transform on the following string; show your work and the output.

HONEYPONY$

Assume $ precedes all characters when you sort rotations.

Answer:

HONEYPONY$  $HONEYPONY
ONEYPONY$H  EYPONY$HONY
YEYPONY$HO  HONEYPONY$
YPONY$HONE  NY$HONEYPO
PONY$HONEY  ONEYPONY$H
ONY$HONEYP  ONY$HONEYP
Y$HONEYPON  Y$HONEYPON
$HONEYPONY  YPONY$HONE

The result will be the last column, i.e., YN$OOHPYNE

b) Assume an initial list $ → A → B → E → H → N → O → P → Y, i.e., initially $ is at index 0, A is at index 1, etc. Assume we use Move-To-Front on the above list to encode the outcome of the BWT transform from part (a). Show what numbers will be encoded (you need to show how the list is updated).

Answer:

$ → A → B → E → H → N → O → P → Y \implies Y : 8 \text{ is encoded}

Y → $ → A → B → E → H → N → O → P \implies N : 6 \text{ is encoded}

N → Y → $ → A → B → E → H → O → P \implies $ : 2 \text{ is encoded}

$ → N → Y → A → B → E → H → O → P \implies O : 7 \text{ is encoded}

O → $ → N → Y → A → B → E → H → P \implies O : 0 \text{ is encoded}

O → $ → N → Y → A → B → E → H → P \implies H : 7 \text{ is encoded}

H → O → $ → N → Y → A → B → E → P \implies P : 8 \text{ is encoded}

P → H → O → $ → N → Y → A → B → E \implies Y : 5 \text{ is encoded}

Y → P → H → O → $ → N → A → B → E \implies N : 5 \text{ is encoded}

N → Y → P → H → O → $ → A → B → E \implies E : 8 \text{ is encoded}

So, the text is encoded as 8627078558.

c) Assume an initial list $ → A → B → C → D, i.e., initially $ is at index 0, A is at index 1, etc. Assume we use Timestamp on the above list to encode DB$AABCD. Show what numbers will be encoded (you need to show how the list is updated and in particular its last state).

Answer:  The list is updated as follows:

$ → A → B → C → D \implies DB$A : the first accesses to these items does not change the list, 4, 2, 0, 1 are encoded

$ → A → B → C → D \implies A : 1 \text{ is encoded}

A → $ → B → C → D \implies B : 2 \text{ is encoded}

A → B → $ → C → D \implies C : 3 \text{ is encoded}

A → B → $ → C → D \implies D : 4 \text{ is encoded}

A → B → D → $ → C

So, the text is encoded as 42011234.
d) Assume an initial list $\rightarrow A \rightarrow B \rightarrow C \rightarrow D$. A compressing scheme that uses Move-To-Front has encoded the following numbers for a text $T$. Show what the actual text is. The numbers are 2 2 0 3 3.

Answer:

\[
\begin{align*}
$ \rightarrow A \rightarrow B \rightarrow C \rightarrow D & \implies 2: B \text{ is decoded} \\
B \rightarrow $ \rightarrow A \rightarrow C \rightarrow D & \implies 2: A \text{ is decoded} \\
A \rightarrow B \rightarrow $ \rightarrow C \rightarrow D & \implies 0: A \text{ is decoded} \\
A \rightarrow B \rightarrow $ \rightarrow C \rightarrow D & \implies 3: C \text{ is decoded} \\
C \rightarrow A \rightarrow B \rightarrow $ \rightarrow D & \implies 3: $ \text{ is decoded}
\end{align*}
\]

So, the text is decoded as $BAAC$.

Problem 2 Splay Trees [10 marks]

a) Apply the splay operation on the following splay tree when there is a request to node ‘8’. Show your steps.

b) Consider a splay tree on $m$ nodes. Assume the next access is to a node that is selected uniformly at random. What is the expected number of the children of the root after the access? (you should provide a number as a function of $m$). Answer: The root after the access will be the newly accessed item. It will have one child if and only if it is either the largest or the smallest item in the tree. The chance of that happening is $2/m$. So, with a chance of $2/m$, the root has 1 child and with a chance of $(m-2)/m$, it has two children. On expectation, it will have $2/m + 2(m-2)/m = 2m - 2/m$ children.

(c) Prove or disprove the following statement: “after a splay operation, the old root is always at depth 1 or 2 (i.e., it is at distance 1 or 2 of the new root)”.

Answer: It is true; the old root’s position is changed as a result of the very last rotation; the old root will be at depth 1 in case of the last operation being zig or zig-zag, and at depth 2 in case of the last operation being zig-zig.
Problem 3  Paging & Resource Augmentation [5 marks]

Sometimes when we analyse online algorithms, we reduce the power of OPT to be more fair to the online algorithm; it is called resource augmentation. For example, for the paging algorithm, instead of comparing an online algorithm with an optimal algorithm which has the same cache-size, we assume the size of the cache of OPT is smaller than that of algorithm. Clearly, when comparing with a weaker optimal algorithm, the competitive ratio of algorithms is expected to improve.

Assume the cache of a marking algorithm A has size k and the cache of OPT has size k/5. Provide an upper bound for the competitive ratio of LRU under this setting.

Answer: Use a phase partitioning technique as we did for LRU (so that there are k distinct pages in each phase). As discussed in the class, the number of faults by any marking algorithm for each phase is at most k. Opt, however, has a cache of size k/5; each phase contains k distinct pages, and only k/5 of them can be hits by OPT in their first access. So, OPT has to incur a cost of k − k/5 = 4k/5 per phase. This gives a ratio of at most k/(4k/5) = 5/4 for each phase, which can be extended to the whole sequence for an upper bound.

Problem 4  k-Server Lower Bound [10 marks]

Consider the k-server under resource augmentation setting (see Question 3) where the offline algorithm has k − 1 servers while the online algorithm has k servers. Adapt the lower bound that we saw in the class to show a lower bound for the competitive ratio of any deterministic algorithm for this setting.

Answer: As before consider a subgraph of size k and let C denote the total cost of the online algorithm. We devise \( \binom{k}{2} \) offline algorithms that have a total cost of roughly \((k−1)C\) (there would be an additive constant for initial moves of the server of offline algorithms); this means the best among these offline algorithms has a cost no more than \(\frac{(k−1)C}{\binom{k}{2}} = \frac{2C}{k} \). Consequently the competitive ratio is at least \(\frac{C}{\frac{2C}{k}} = \frac{k}{2}\).

At each given time, the server has at least 1 exposed point, that is, a vertex without a server. We maintain an invariant that ensures that all offline algorithms have a server in the exposed point of the online algorithm. The other \(k−2\) servers of the offline algorithms are located in the \(k \) remaining vertices in a way that no two offline algorithms have the same configuration; as such, the number of offline algorithms is \(\binom{k−2}{2}\), which is \(\binom{k}{2}\).

We always ask for a request to a vertex \(x\) that is exposed by the online algorithm. The algorithm makes a move from some other vertex \(y\) to \(x\). Now, \(y\) is exposed in the online algorithm. Among the offline algorithms, \(k−1\) of them do not have a server at \(y\); these algorithms make a reverse move from \(x\) to \(y\), each paying a cost equal to that of the online algorithm. So, at each step, the online algorithm pays a cost \(d\) and all offline algorithms pay a total cost of \((k−1)d\). After these moves, the invariant about the exposed vertices is maintained, and we can repeat this process indefinitely.

Problem 5  Double-Coverage-Algorithm [10 marks]

(a) Consider a different analysis of the double coverage algorithm in which the potential is defined as \(3(P+Q)\), where \(P\) and \(Q\) are defined as before. Follow the steps of the algorithm and indicate whether we can achieve an upper bound for the competitive ratio of the algorithm with this way of defining the potential (if the answer is yes, you should show what that competitive ratio is).

Answer: All steps will be similar to the previous potential function argument (in particular, the changes in \(P\) and \(Q\) remain the same). In Case 1, however, the difference in potential will be at most \(3(kj−d)\) instead of \(kj−d\), and the amortized cost will be at most \(d+3(kj−d) < 3kj = 3cost(\text{opt})\). In Case 2, the difference in potential will be at most \(3(kj−2d)\), and the amortized cost will be at most \(2d+3(kj−2d) < 3kj\). So, the best competitive ratio that we get in this case is \(3kj\).

(b) Consider a different analysis of the double coverage algorithm in which the potential is defined as \(P+3Q\), where \(P\) and \(Q\) are defined as before. Follow the steps of the algorithm and indicate whether we can achieve an upper bound for the competitive ratio of the algorithm with this way of defining the potential (if the answer is yes, you should show what that competitive ratio is).

Answer: This potential does not work for Case 1. Like before we have \(\Delta_{\text{opt}}\Phi = kj\). For the actions of the algorithm, the value of \(P\) is increased by \(-kd\) while the value of \(Q\) is increased by \(3(k−1)d\). The potential will increase by \(2kd−3d\) (for actions of Alg), and the amortized cost is at most \(d\) plus the difference in potential, i.e., \(d+(2kd−3d)+kj = 2kd−2d+j\), which is not possible to bound with the cost \(j\) of Opt.
Problem 6  [Bonus] Double-Coverage-Algorithm [10 marks]

Consider a variant of the double coverage algorithm for paths that works as follows. Upon a request to a vertex \( x \), if there is only one server on its two sides, that server is moved to serve the request (as before). If there are two servers on the two sides of the request, both move towards the request (as before). The left server, however, moves at twice the speed of the right server. In the below picture for example, if there is a request to 5 at time \( t \), the left server \( s_1 \) gets there before \( s_2 \), despite being further initially.

Follow the same steps as we took in the analysis of the double coverage algorithm to provide an upper bound for this variant of the algorithm.

Answer: Assume the vertices are indexed from 1 to \( m \). As before we use the potential function argument. Define the potential as \( 3(P + Q') \). Here \( q(s, s') \) is defined differently; assume \( s \) at vertex with label \( i \) and \( s' \) is at vertex of label \( j \). \( q(s, s') \) is defined as \( j - i/2 \).

- For the actions of \( \text{Opt} \), we have \( \text{cost}_{\text{opt}} = j \), while \( \Delta_{\text{Opt}} \Phi \leq 3kj \); this is because \( P \) increases by at most \( k \cdot j \) while \( Q' \) remains unchanged.

- In case 1, the algorithm moves a distance of \( d \), the value of \( P \) decreases by \( kd \) and the value of \( Q' \) increases by at most \( (k - 1)d \) (it will increase by \( (k - 1)d/2 \) if the leftmost server is moving and \( (k - 1)d \) if the rightmost server moves). Regardless, the value of \( 3(P + Q') \) decreases by at least \( 3d \). So, we have \( \Delta_{\text{Alg}} \Phi \leq -3d \). The actual cost of the algorithm is \( d \) and \( \Delta \Phi \) is the sum of \( \Delta_{\text{Opt}} \Phi + \Delta_{\text{Alg}} \Phi \) which is at most \( -3d + 2kj \). The amortized cost is consequently at most \( d - 3d + 3kj < 3kj = 3\text{cost}(\text{Opt}) \).

- In case 2, the algorithm moves a distance of \( 3d \) (left server \( L \) moves \( 2d \) and right server \( R \) moves \( d \)). For any other server \( Z \), \( q(Z, R) \) decreases by 1 and \( q(Z, L) \) increases by 1. So, the value of \( Q \) is decreased by \( 1/2(2d) + d = 2d \). The value of \( P \) increases by at most \( d \) (which happens when \( p(L) \) is increased by \( 2d \) and \( p(R) \) decreased by \( d \)). So, \( 3(P + Q') \) is decreased by at least \( 3d \), i.e., \( \Delta_{\text{Alg}} \Phi \leq -3d \). The amortized cost is then \( 3d - 3d + 3kj = 3kj \).