1 Proof of Safety of Strategy for Interval Scheduling Problem

Lemma 1. *Let j be the job with the earliest finishing time. Then there is an optimum solution in which j is scheduled.*

Proof. Let S be any optimum solution for the interval selection instance. If $j \in S$, then we are done. So, we assume $j \notin S$. Let j' be the first job in S, i.e, the job with the earliest finishing time in S.

Then we show that $S' := S \setminus \{j'\} \cup \{j\}$ is also an optimum solution for the instance. First, all jobs in $S \setminus \{j'\}$ start at or after $f_{j'}$ since j' is the first job in S. By our choice of j we have $f_j \leq f_{j'}$. So, all jobs in $S \setminus \{j'\}$ start at or after f_j. Therefore, S' is a valid solution. $|S'| = |S|$ and thus S' is also valid.

In any case, there is an optimum solution that contains j. $\square$

2 Proof of Safety of Strategy for Offline Caching Problem

Lemma 2. *Assume at time 1 a page fault happens and there are no empty pages in the cache. Let p^* be the page in cache that is not requested until furthest in the future. Then, there is an optimum solution in which p^* is evicted at time 1.*

Proof. Let S be any optimum solution for the offline caching instance. If S evicts page p^* at time 1, then we are done. So assume S evicts some page $p' \neq p^*$ at time 1. Let t^* be the first time that p^* is requested, and t' be the first time that p' is requested; if a page is not requested in the future, then we let the correspondent time to be 0. So by our choice of p^*, we have $t' < t^*$. (An exception is that both t' and t^* are ∞. In this case both p' and p^* are not used in the future and thus evicting p^* is equivalent to evicting p'.)

Our goal is to construct a new optimum solution S' such that

- S' evicts p^* at time 1.
- the number of page misses in S' is at most that in S.

(Notice that by the assumption that S is optimum, the second property implies that the number of page misses in S' is the same as that in S. However, in our proof, it is more convenient for us to not take the optimality of S into the account, and only show one direction described in the second property.)

For a time t, let $\mathsf{cache}_t(S)$ and $\mathsf{cache}_t(S')$ be the sets of pages in the solution S and S' respectively, at the end of time t. Then $|\mathsf{cache}_t(S)| = |\mathsf{cache}_t(S')| = k$ for any time t.

At time 1, S evicts p' and we let S' evict p^*. They both have a page fault. Moreover, $\mathsf{cache}_1(S) \setminus \mathsf{cache}_1(S') = \{p^*\}$ and $\mathsf{cache}_1(S') \setminus \mathsf{cache}_1(S) = \{p'\}$. That is, at the end of time 1, the only difference between $\mathsf{cache}_1(S)$ and $\mathsf{cache}_1(S')$ is that the former contains p^* and the latter contains p'.

For every time t from 2 to $t' - 1$, we copy the solution S to the solution S'. Notice that both p' and p^* are not requested in this time period. If S has a page hit, so does S'. If S has a page miss and evicts a page other than p^*, then S' has a page miss and evicts the same page. If at some time t, S evicts p^*, then we let S' evict p', and we have $\mathsf{cache}_t(S') = \mathsf{cache}_t(S)$. In this case, we can copy S until the time T, to obtain the desired solution S'.

So, we assume in any time t from 2 to $t' - 1$, S does not evict the page p^*. Then, $\mathsf{cache}_{t'-1}(S) \setminus \mathsf{cache}_{t'-1}(S') = \{p^*\}$ and $\mathsf{cache}_{t'-1}(S') \setminus \mathsf{cache}_{t'-1}(S) = \{p'\}$. At time t', S has a page miss and S' has a page hit. S evicts some page and load page p'. Thus, by time t', S' has one less page miss than S. Moreover $|\mathsf{cache}_{t'}(S) \cap \mathsf{cache}_{t'}(S')| \geq k - 1$.

For every time t from $t' + 1$ to time T, we again copy S to construct our solution S'. We maintain that $|\mathsf{cache}_t(S) \cap \mathsf{cache}_t(S')| \geq k - 1$. Moreover, if at some time step t, S has a page hit and S' has a page miss, then we have $\mathsf{cache}_t(S') = \mathsf{cache}_t(S)$. Focus on time t:

- If both S and S' have a page hit, then nothing needs to be done.
- If both S and S' have a page miss, then we let S' evict the same page as S does; if S evicts the page in $\mathsf{cache}_t(S) \setminus \mathsf{cache}_{t'}(S')$, then let S' evict the page in $\mathsf{cache}_t(S') \setminus \mathsf{cache}_{t'}(S)$. We still have $|\mathsf{cache}_t(S) \cap \mathsf{cache}_t(S')| \geq k - 1$.

- If S has a page miss and S' has a page hit, then we do not need to do anything for S'. S will evict some page and load the page in $\mathsf{cache}_{t-1}(S') \setminus \mathsf{cache}_{t-1}(S)$. So, we shall have $|\mathsf{cache}_t(S') \cap \mathsf{cache}_t(S)| \geq k-1$. (Using the optimality of S, one can show that this case can not happen; but this is not needed.)
- If S has a page hit and S' has a page miss, then the page requested is the one in $\mathsf{cache}_{t-1}(S) \setminus \mathsf{cache}_{t-1}(S')$. Then, we let S' evict the page in $\mathsf{cache}_{t-1}(S') \setminus \mathsf{cache}_{t-1}(S)$, and load the page in $\mathsf{cache}_{t-1}(S) \setminus \mathsf{cache}_{t-1}(S')$. Then, we have $\mathsf{cache}_t(S') = \mathsf{cache}_t(S)$.

So, from time $t'+1$ to T, S' incurs at most one more page miss than S does. Overall the number of page misses in S' is at most that in S. □

3 Proof of Safety of Strategy for Best Prefix Code Problem

Lemma 3. *Given an instance for the best prefix code problem, and let x_1 and x_2 be the two least frequent letters. Then there is an optimum encoding tree in which the two nodes holding x_1 and x_2 are brothers.*

Proof. Let S be an optimum encoding tree for the instance. Then, there must be two deepest nodes in S that are brothers. Let v_1 and v_2 be the two nodes. If the two nodes hold x_1 and x_2 (not necessarily in this order), then we are done. Otherwise, we can make one or two swaps to make this happen.

In each swap, we take a node $u \notin \{v_1, v_2\}$ that holds some letter $x \in \{x_1, x_2\}$, and the node $v \in \{v_1, v_2\}$ that holds some letter $y \notin \{x_1, x_2\}$. Then we swap the letters held by the two nodes: we let u hold y and let v hold x. Notice that $f_x \leq f_y$ as x_1 and x_2 are the two least frequent letters, and $d_u \leq d_v$, where d_u and d_v are the depth of u and v in the solution S respectively. This holds since v_1 and v_2 are two deepest nodes in the tree. Then, $d_u f_y + d_v f_x \leq d_u f_x + d_v f_y$ since $(d_u f_x + d_v f_y) - (d_u f_y + d_v f_x) = (d_v - d_u)(f_y - f_x) \geq 0$. So, after the swap, the cost can only go down. Eventually, we obtain a new solution S' whose cost is at most that of S. Moreover, in S', v_1 and v_2 hold x_1 and x_2. □