5. We have shown that for every input sequence $(x_1, \ldots, x_n)$ we have $FirstFit(x_1, \ldots, x_n) \leq \frac{17}{10} OPT(x_1, \ldots, x_n) + 3$ using the modified weighting technique. The proof was a bit involved. One can show a much simpler proof if one aims for a slightly weaker result, namely: $FirstFit(x_1, \ldots, x_n) \leq \frac{7}{4} OPT(x_1, \ldots, x_n) + 2$. In this case one doesn't even have to use the modified weighting technique and can apply the original weighting technique instead. This is what you are asked to do in this question.

(a) Consider the following weight function w

$$w(x) = \begin{cases} \frac{3}{2}x & \text{if } 0 \leq x \leq 1/2 \\ 1 & \text{if } x > 1/2 \end{cases}$$

Prove that for all $k \in \mathbb{N}$ and for all $y_1, \ldots, y_k \in \mathbb{R}_{\geq 0}$ if $\sum_{i=1}^{k} y_i \leq 1$ then $\sum_{i=1}^{k} w(y_i) \leq 7/4$. This shows that applying the weight function to a bin can never stretch the total weight over $7/4$.

Pick an arbitrary k.

Case 1. $\forall y_j \in \{y_1, \ldots, y_k\}$, $y_i \leq \frac{1}{2}$

Then $w(y_j) = \frac{3}{2} y_j$, $\forall j$

So $\sum_{i=1}^{k} w(y_i) = \frac{3}{2} \sum_{i=1}^{k} y_i \leq \frac{3}{2}$.

Case 2. $\exists y_j \in \{y_1, \ldots, y_k\}$ s.t. $y_j > \frac{1}{2}$. Obviously y_i is unique.

Then $w(y_j) = 1$.

Since $y_j > \frac{1}{2}$. $\sum_{i=1}^{k} y_i \leq 1$, we have $\sum_{\substack{i=0 \\ i \neq j}}^{k} y_i < \frac{1}{2}$

Then $\sum_{i=1}^{k} w(y_i) = w(y_j) + \sum_{\substack{i=0 \\ i \neq j}}^{k} w(y_i) = 1 + \frac{3}{2} \sum_{\substack{i=0 \\ i \neq j}}^{k} y_i < 1 + \frac{3}{2} \cdot \frac{1}{2} = \frac{7}{4}$

In both cases $\sum_{i=1}^{k} w(y_i) < \frac{7}{4}$. $\square$

(b) Consider an arbitrary input sequence $(x_1, \ldots, x_n)$ and the bins produced by the $FirstFit$ algorithm. Define the following bin types:

Type I. Each bin of this type contains a single item of weight $> 1/2$.

Type II. Each bin of this type contains more than one item and the total weight of the bin is $> 2/3$.

Type III. Any bin that is not of Type I or Type II.

Prove that the weight function applied to each bin of Type I or Type II is ≥ 1.

$$w(x) = \begin{cases} \frac{3}{2}x & \text{if } 0 \leq x \leq 1/2 \\ 1 & \text{if } x > 1/2 \end{cases}$$

Type I:

Each bin of Type I contains exactly one item x and its size is $> \frac{1}{2}$.

And its weight is $w(x) = 1$. So the total weight of the bin is $1 \geq 1$.

Type II:

Each bin of Type II contains at least two items and the total size $> \frac{2}{3}$.

So every item in the bin has size $\leq \frac{1}{2}$.

So the total weight of the bin is

$$\sum_i w(y_i) = \frac{3}{2} \cdot \sum_i y_i \geq \frac{3}{2} \cdot \frac{2}{3} = 1.$$

Thus, both Type I and Type II have total weight ≥ 1. $\square$

(c) Prove that there can be at most 2 bins of Type III.

For a bin to be Type III, it either:

1. Contain one item with size $\leq \frac{1}{2}$
2. contain more than one item but total size $\leq \frac{2}{3}$.

Suppose, for contradiction, there are three bins of Type III in the final packing. Call them A, B, C. WLOG, let A be the bin that was opened first, then B, then C.

Case 1. A contains one item. then A has size $\leq \frac{1}{2}$.

Then B must have more than one item, with total size $\leq \frac{2}{3}$.

This is because if B has only one item of size $\leq \frac{1}{2}$, it would have been placed into A.

Similarly, C cannot have only one item. Then it becomes Type III after taking a second item. Before that, it must be Type I with size $> \frac{1}{2}$. So it must have taken an item of size $< \frac{2}{3} - \frac{1}{2} = \frac{1}{6}$

This impossible because the item would fit A.

Case 2. A has more than one item. Then size of A $\leq \frac{2}{3}$.

Then B must have only one item x. with $\frac{1}{3} < x \leq \frac{1}{2}$.

This is because if B consists of two or more items with total size $\leq \frac{2}{3}$, at least one of them has size $\leq \frac{1}{3}$.

Similarly C cannot have two or more items with total size $\leq \frac{2}{3}$.

However, C also cannot have only one item with size $\leq \frac{1}{2}$, since that would fit in B.

In both cases, C is impossible. So there are at most 2 bins of type III $\square$

(d) Combine all of the above to show that $FirstFit(x_1, \ldots, x_n) \leq \frac{7}{4} OPT(x_1, \ldots, x_n) + 2$.

$OPT(x_1, \cdots, x_n)$ is the total number of bins used by OPT.

By part (a), each bin provides total weight at most $\frac{7}{4}$.

So in an optimal packing using OPT bins, the weight W satisfies

$$W \leq \frac{7}{4} OPT(x_1, \cdots, x_n)$$

$FirstFit(x_1, \cdots, x_n)$ is the total number of bins used by FirstFit.

By part b, Type I and Type II bins have weight at least 1.

By part c, there are at most 2 Type III bins.

So there are at least $FirstFit(x_1, \cdots, x_n) - 2$ bins of Type I or Type II.

So in the packing by FirstFit, the weight w satisfies

$$FirstFit(x_1, \cdots, x_n) - 2 \leq w.$$

Then we have

$$FirstFit(x_1, \cdots, x_n) - 2 \leq w \leq W \leq \frac{7}{4} OPT(x_1, \cdots, x_n)$$

Rearrange, we get desired

$$FirstFit(x_1, \cdots, x_n) \leq \frac{7}{4} OPT(x_1, \cdots, x_n) + 2. \qquad \square$$

7. Suppose that in the Time Series Search problem the algorithm only knows L beforehand. Does there exist a deterministic algorithm with competitive ratio better than ϕ?

Claim. $\rho(ALG) \geq \phi$.

Proof.

Consider arbitrary deterministic algorithm ALG.

Case 1.

ALG always trades on the last day.

Then for some $U > L$, an adversary presents input sequence:

$p_1 = U$, $p_2 = L$.

So ALG trades on the last day. $ALG = L$. But $OPT = U$

So $\rho(ALG) = U/L = \phi$.

Case 2.

ALG would trade before the last day.

Then for some arbitrarily small $\varepsilon > 0$, an adversary presents

Input series: $p_1 = L$, $p_2 = L + \varepsilon$, $p_3 = L + 2\varepsilon$, $\cdots$, $p_i = L + (i-1)\varepsilon$.

Keep giving new prices until ALG trades, say on day k,

then presents $p_{k+1} = U$ significantly larger than p_k.

Then $ALG = L + (k-1)\varepsilon$, $OPT = U$.

$\rho(ALG) = U / L + (k-1)\varepsilon$. as $\varepsilon \to 0$, $\rho(ALG) \to U/L = \phi$.

So no deterministic algorithm with competitive ratio better than ϕ. $\square$

3. Considr any online problem. Prove that if there is no optimal deterministic algorithm then there is no randomized algorithm (using only finite probability distributions) that always returns an optimal solution.

Consider arbitrary online problem. Assume no optimal deterministic algorithm for the problem.

Let RAN be an arbitrary randomized algorithm using only finite probability distributions. Then RAN has a finite support P_{RAN}.

For each $A_i \in P_{RAN}$, pick an instance I_i such that A_i is not optimal on this instance. I_i exists for all i due to assumption.

Let U be the uniform distribution over the picked instances.

Let $O(A, I) = 1$ represent "A spends more cost than optimal on I" and $O(A, I) = 0$ represent "A spends optimal cost on I".

Then for every deterministic algorithm $A_i \in P_{RAN}$.

$$\mathbb{E}_{I \sim U}[O(A_i, I)] \geq \frac{1}{k} > 0$$

This is because A_i does not spend optimal cost on I_i with probability $\frac{1}{k}$.

So we have

$$\min_{A \text{ deterministic}} \mathbb{E}_{I \sim U}[O(A, I)] \geq \frac{1}{k} > 0$$

By Yao's Minimax Principle,

$$\max_{I} \mathbb{E}_{r \sim R}[O(ALG_r, I)] \geq \min_{A \text{ deterministic}} \mathbb{E}_{I \sim U}[O(A, I)] \geq \frac{1}{k} > 0.$$

Therefore, no randomized algorithm can achieve $O(ALG_r, I) = 0$ on every instance I. $\square$

4. This question relates to Section 4.4 in the text. [20 points]
Consider the (unweighted) disjoint paths problem on an N node line graph. But now consider online algorithms with revoking.

- Show that for input instances where all calls have the same path length, the optimal deterministic competitive ratio is exactly 2 (expressing ratios as being greater than or equal to 1). This requires both a positive and matching negative result. [5 points]
  Hint: Use a charging argument for the positive result which does not need revoking.

In a line graph with vertices labeled $1, 2, \dots, N$.
Each path (s, t) defines a unique simple path covering all vertices between them, WLOG, let $1 \le s < t \le N$, which can be represented by the interval $[s, t)$.
Thus, the unweighted disjoint paths problem on a line graph is equivalent to the unweighted interval selection problem, where each path corresponds to an interval and disjointness corresponds to non-overlap of intervals.
We define each path (s, t) as interval $[s, t)$, closed on the left, but open on the right, so that intervals sharing endpoints are still considered non-overlap, just as simple paths sharing one terminal vertex are still disjoint.

1. Positive Result

For input instances where all calls have the same path length, it is equivalent to intervals all have the same length.
Consider the following algorithm.

```
Algorithm GT
  S = ϕ          # set of intervals taken.
  while there are more intervals:
      On arrival of interval I:
      if I does not conflict with any interval in S:
          S = S ∪ {I}   # take I.
```

Claim: $\rho(GT) \leq 2$.

Proof.

Define a mapping $f: OPT \to GT$, where OPT is the offline optimal schedule.

We consider a charging argument.

Any interval I selected by OPT. There are several cases.

Case 1.

I is also selected by GT. This happens when I is not conflicted with any other intervals already taken by GT. In this case, I is charged to itself.

Case 2.

I conflicts with one interval that is already taken by GT. In this case, I is charged to the interval it conflicts with.

Case 3.

I conflicts with two interval that is already taken by GT. In this case, I can be charged to either of these

intervals. WLOG, we charge I to the interval it conflicts on the right.

Since all intervals have equal length, an taken by GT can only be conflicted with at most two intervals taken by OPT.
So we know that for any interval taken by GT, it is charged by at most two intervals in OPT.
Therefore, algorithm GreedyTake has a competitive ratio at most 2.
Proven $\rho(GT) \leq 2$. $\square$

2. Negative Result.

Claim.

No deterministic algorithm can achieve a competitive ratio better than two.

Proof.

Fix any deterministic algorithm ALG, we adversarially release intervals as follows.

Step 1. Release interval I_1.

Case 1. If ALG does not take the interval I_1, then ALG schedules 0 interval, but OPT schedules 1.

We stop releasing intervals right away.

The ALG is unboundedly bad. We focus on case 2.

Case 2. ALG takes I_1.

WLOG, let I_1 has length 1.

Step 2. Release I_2 and I_3 so that I_2 conflicts with the left half of I_1, and I_3 conflicts with right half of I_1. That is, suppose $I_1 = [k, k+1)$, let $I_2 = [k-\frac{1}{2}, k+\frac{1}{2})$, $I_3 = [k+\frac{1}{2}, k+\frac{3}{2})$.

Then ALG schedules 1 interval, but OPT schedules 2 intervals (I_2 and I_3).

We stop releasing intervals right away.

The competitive ratio is 2.

Combining the two cases, ALG has competitive ratio ≥ 2. $\square$

By the positive result and the negative result, we conclude that the optimal deterministic competitive ratio is exactly 2.

- Show that for input instances where calls can have one or two possible path lengths, the deterministic competitive ratio is exactly 4. [10 points]
  Hint: For the positive result, we now need revoking. There is a charging argument for the positive result but it is not as direct as for single length instances. The interesting case is how to charge calls that caused a previous call to be revoked.

Consider two possible path lengths. We denote the two possible lengths by A and B. WLOG, let $A < B$.

There are several ways for two intervals to be conflicted to each other.

1. An interval is partially overlapped with another.
2. An interval of length A is entirely contained in an interval of length B.

1. Positive Result.

Consider the algorithm:

```
Algorithm GTR
  S = ϕ                    # set of intervals taken.
  while there are more intervals:
      On arrival of interval I:
          for J in S:
              if J contains I:
                  S = S ∪ {I} \ {J}.   # revoke J, take I.
              else if J partially overlaps with I:
                  S = S.               # discard I.
          S = S ∪ {I}                  # Take I.
```

Claim. : $\rho(GTR) \leq 4$.

Proof.

Define a mapping $f: OPT \to GTR$, where OPT is the offline optimal schedule. We consider a charging argument.

Any interval I selected by OPT, there are several cases.

Case 1.

I is also taken by GTR. The happens either because I did not conflict with any other interval previously taken by GTR. Or because I was entirely contained by another interval taken by GTR.

In this case, I is charged to itself.

Case 2.

I partially overlaps with one interval taken by GTR. In this case I is charged to the interval it overlaps with.

Case 3.

I partially overlaps with two intervals taken by GTR. In this case I can be charged to either one of the intervals.

WLOG. charge I to the interval that overlaps with I on the right.

Case 4.

I entirely contains an interval taken by GTR. We can assume this never happens, because we can have another optimal schedule OPT' with I replaced with the interval contained by I.

Consider any interval J taken by GTR. is later replaced by an interval K because K is entirely contained in J, then for all intervals in OPT originally charged to J. redirect the charges to K.

So there are two ways for an interval in OPT to charge to intervals taken by GTR.

1. indirect charge. As described above
2. direct charge. As in Case 1-4.

We can see that any interval taken by GTR can only be charged directed by at most two intervals in OPT. Note that if an interval K of length B in GTR overlaps with more than three other non-overlapping intervals, K must contain one of those intervals entirely. so K would have been replaced.

Further, we argue that an interval taken by GTR can only be charged indirectly by at most 2 intervals in OPT.

We consider a sequence of intervals $I_1, I_2, \dots, I_n$ such that I_i was originally taken by GTR, but later replaced by I_{i+1} for all i. And I_n is the interval kept by GTR in the end. So by how we defined indirect charge, the indirect charge on I_n is the accumulated from all I_i with $i < n$.

We see that n is at most 2. This is because when intervals can have only two possible lengths, A or B, the only way to have an interval containing another is that an interval of length B contains an interval of length A. Any interval of length A cannot contain any other interval, and any interval of length B cannot be contained by any other interval.

Since each interval can have direct charge at most 2, and each interval can only have indirect charge from at most one other interval. We can see that each interval can have indirect charge at most 2.

Therefore, each interval in GTR can be charged by at most $2+2=4$ intervals in OPT.

Thus, GTR has a competitive ratio at most 4.

proves $f(GTR) \leq 4$. $\square$

2. Negative Result.

Claim.

No deterministic algorithm can achieve a competitive ratio better than 4.

Proof.

Fix any deterministic algorithm ALG. We adversarially release intervals as follows. Pick B to be one of the possible lengths. Fix $\varepsilon > 0$ such that $B > 2\varepsilon$.

Step 1.

Release intervals $I_1 = [0, B)$ first and then $I_2 = [B-\varepsilon, 2B-\varepsilon)$

Case 1.1 If ALG takes I_1 and discards I_2, release another interval $I_3 = [-B+\varepsilon, \varepsilon)$. This extends the interval chain to the left.

Case 1.2 If ALG takes I_1 but replaces it with I_2, release another interval $I_4 = [2B-2\varepsilon, 3B-2\varepsilon)$. This extends the interval chain to the right.

In either case, keep extending the chain to one direction if ALG keeps replace its taken interval with new one on that direction, until it chains.

Step 2.

Consider three intervals:

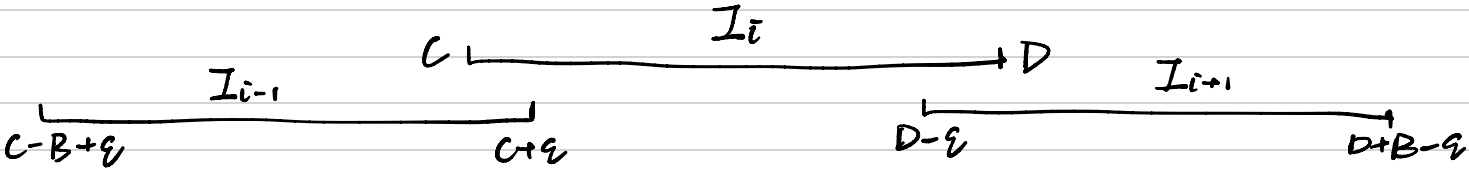
Let I_{i-1} be the one ALG replaced most recently.

Let I_i be the one ALG refused to be replaced.

Let I_{i+1} be the one ALG refused to replace I_i.

Then I_{i-1} overlaps with I_i by exactly ε. and I_{i+1} also overlaps with I_i by exactly ε.

WLOG, let I_{i-1} be on the right side of I_i, and I_{i+1} be on the left side of I_{i+1}. Let $I = [C, D)$ for some C, D.



For now, ALG = 1 as it only takes I_i.

But ODT ≥ 2. as it takes I_{i-1} and I_{i+1} possibly other ones.

Step 3.

Consider the gap between I_{i-1} and I_{i+1},

namely, $I_g = [C+\varepsilon, D-\varepsilon)$. the length of I_g is

$|I_g| = D-\varepsilon-C-\varepsilon = (D-C)-2\varepsilon = B-2\varepsilon$.

To make the notations simple, WLOG, we introduce

$N := C+\varepsilon$, so $D-\varepsilon = C+\varepsilon+B-2\varepsilon = C+B+\varepsilon = N+B-2\varepsilon$.

So $I_g = [N, N+B-2\varepsilon)$.

Pick the second interval length as $A = \frac{B-2\varepsilon}{4}$.

Fix $\delta > 0$ such that $A > 2\delta$.

Now release intervals $I_a = [N+A-\delta, N+2A-\delta)$ first, and then $I_b = [N+2A-2\delta, N+3A-2\delta)$.

Case 3.1. if ALG keeps I_i and discards both I_a and I_b.

We have ALG $= 1$ but OPT ≥ 4 since OPT takes $I_{i-1}, I_{i+1}, I_a, I_b$ and possibly other ones.

Case 3.2 if ALG discards I_i, takes I_a and discards I_b, we release another interval $I_c = [N, N+A)$.

Obviously I_c partially overlaps with I_c.

now ALG either keeps I_a and discards I_c, or discards I_a and takes I_c. Either way, ALG $= 1$.

but OPT ≥ 4 since it takes $I_{i-1}, I_{i+1}, I_c, I_b$ and possibly other ones.

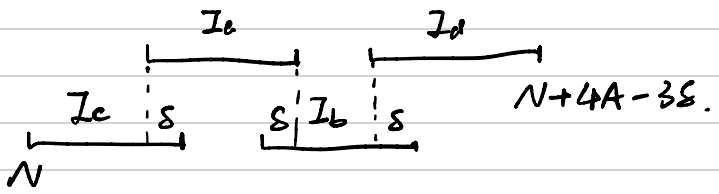
Case 3.3 if ALG discards I_i, takes I_b and discards I_a, we release another interval

$I_d = [N+3A-3\delta, N+4A-3\delta)$.

Obviously I_b partially overlaps with I_d

now ALG either keeps I_b and discards I_d, or discards I_b and takes I_d. Either way, ALG $= 1$.

but OPT ≥ 4 since it takes $I_{i-1}, I_{i+1}, I_a, I_d$ and possibly other ones.



We verify that the four intervals fit entirely in I_g.

$N+4A-3\delta-N = 4A-3\delta = B-2\varepsilon-3\delta < B-2\varepsilon = |I_g|$.

In every cases, ALG = 1 and OPT ≥ 4.

We stop releasing intervals.

Then $\rho(ALG) = \frac{OPT}{ALG} \geq 4$ $\square$

By the positive result and the negative result, we conclude that the optimal deterministic competitive ratio is exactly 4.

- Show that for input instances where calls can have arbitrary path lengths, the deterministic competitive ratio is not constant. [10 points]

Let ALG be any deterministic online algorithm for the problem.

Suppose, for contradiction, that ALG is C-competitive for some constant $C \geq 1$ on all instances with arbitrary lengths.

From previous part, that for any integer $k \geq 1$ there exists an instance whose intervals use exactly k distinct lengths and for which every deterministic algorithm has competitive ratio at least $2k$. In other words,

$$\forall k, \exists I_k \text{ s.t. } \frac{OPT(I_k)}{ALG(I_k)} \geq 2k.$$

Pick k such that $2k > C$, which is possible since k can be arbitrarily large. Let I_k be the corresponding adversarial instance.

By the lower bound for k lengths we get $\frac{OPT(I_k)}{ALG(I_k)} \geq 2k > C$.

Contradicting the assumption that ALG is C-competitive on all instances. Therefore, no constant C can bound the competitive ratio for all instances when interval lengths are arbitrary.

Thus, the competitive ratio is not constant. $\square$

5. Prove that there is no fixed or adaptive priority algorithm for the general knapsack problem that can achieve a constant ratio. [10 points]

Fix any fixed or adaptive priority algorithm ALG.

We show that for any constant C, there exists an adversarial instance such that ALG achieves competitive ratio $\geq C$.

Consider the following construction.

We provide two classes of input items.

Class H consists of items with size 1 and weight $W \geq C$.

Class L consists of items with size $\varepsilon > 0$ s.t. $\varepsilon \leq \frac{1}{C \cdot W}$ and weight 1.

Let $h \in H$, $l_i \in L$ for some i.

Consider the input set $S = \{h, l_1, l_2, l_3, \cdots, l_n\}$, which contains one item from class H and n items from class L. Pick n such that $n \geq CW$

Case 1. ALG prioritizes h.

Then present the input set S, and the first item is h, which is the order decided by ALG.

ALG would accept h, and it is full, and must reject all other items. Then $ALG(S) = W$.

OPT takes in all the items from class L, with $OPT(S) = n$.

the competitive ratio is $\frac{OPT(S)}{ALG(S)} = \frac{n}{W} \geq \frac{CW}{W} = C$

Case 2. ALG prioritizes items from L.

Then present the input set $S = \{l_1, h\}$ and the first item is l_1.

ALG would take l_1, with remaining capacity $1 - \varepsilon < 1$.

So h does not fit anymore. So $ALG(S') = 1$.

OPT would take h only, then $OPT(S') = W$

then the competitive ratio is $\frac{OPT(S')}{ALG(S')} = W \geq C$.

In both cases, the competitive ratio is at least C.

Therefore, no fixed or adaptive priority algorithm can achieve a constant ratio. $\square$