Computer Science C73
Scarborough Campus

September 15, 2021
University of Toronto

Solutions for Homework Assignment #1

Answer to Question 1.

ALGORITHM. We sort the firefighters and fires in non-decreasing order. We look at the first firefighter and first fire; if they are close enough we assign that firefighter to that fire, and we look at the next firefigher and fire; otherwise, if the first fire is to the left of the first firefighter, we look at the next fire (because the first one cannot be assigned to any firefighter), and if the first fire is to the right of the first firefighter we look at the next firefighter (because the first one cannot be assigned to any fire). We continue like this until we run out of firefighters or fires. In pseudocode:

```
1  sort F[1..m] in non-decreasing order
2  sort R[1..n] in non-decreasing order
3  A := ∅;  f := 1;  r := 1
4  while f ≤ m and r ≤ n do
5      if |F[f] − R[r]| ≤ d then A := A ∪ {(f, r)};  f := f + 1;  r := r + 1      ▷ f close enough to r
6      elsif R[r] < F[f] then r := r + 1                                          ▷ r too far and to the left of f
7      else f := f + 1                                                            ▷ r too far and to the right of f
8  return A
```

CORRECTNESS. First we note that algorithm terminates, since in each iteration at least one of f and r is incremented, so after at most $m + n$ iterations, either $f > m$ or $r > n$.

Next we note that the algorithm produces an assignment (of firefighters to fires) that satisfies the constraints: Because each time f is assigned to r we increment both f and r, constraints (a) and (b) are satisfied; and because of the condition of line 5, when f is assigned to r constraint (c) is also satisfied.

It remains to prove that the algorithm produces an optimal assignment, i.e., one that assigns firefighters to as many fires as possible (subject to the constraints). We prove this using the "promising set"approach. Specifically we prove:

Promising Set Lemma. *At the end of each iteration, the assignment in A is contained in some optimal assignment.*

Proof. The proof is by induction on the iteration number. Let A_t be the value of variable A at the end of iteration t. (As usual "the end of iteration 0" refers to the point just before the first iteration of the loop.)

The basis is trivially true, since $A_0 = \varnothing$. Suppose the invariant holds after iteration t, and let A_t^* be an optimal set that contains A_t. We will now prove that, if iteration $t + 1$ exists, the invariant remains true after iteration $t + 1$; that is, A_{t+1} is contained in some optimal assignment A_{t+1}^*.

If no pair is added to A in iteration $t + 1$ (lines 6 or 7), then we take $A_{t+1}^* = A_t^*$ and we are done by the induction hypothesis. So, suppose that pair (f, r) is added to A in iteration $t + 1$ (line 5). There are four cases:

CASE 1. $(f, r) \in A_t^*$. Then we take $A_{t+1}^* = A_t^*$ and we are done by the induction hypothesis.

CASE 2. There is no pair (f, r') in A_t^*, for any r'. Then A_t^* contains a pair (f', r) (otherwise A_t^* would not be optimal, as we could add (f, r) to it without violating the constraints). In this case, define A_{t+1}^* to be A_t^* with (f', r) removed and (f, r) added; i.e., $A_{t+1}^* = (A_t^* - \{(f', r)\}) \cup \{(f, r)\}$. A_{t+1}^* satisfies the constraints and has the same number of pairs as A_t^*, so it is an optimal set that contains A_{t+1} and we are done.

CASE 3. Similar to Case 2.

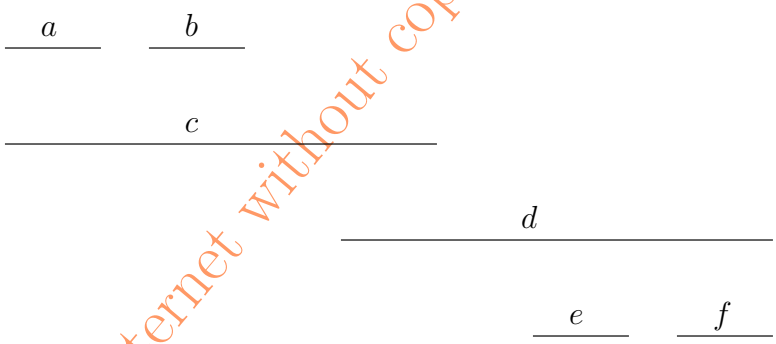
CASE 4. There are pairs (f, r') and (f', r) in A_t^*, for some $f' \neq f$ and $r' \neq r$. Note that because the greedy algorithm considers firefighter and fire locations in non-decreasing order, $f < f'$ and $r < r'$. (This is because, by the induction hypothesis, $A_t \subseteq A_t^*$.) Therefore among the four firefighter-fire pairs $(f, r), (f, r'), (f', r), (f', r')$, the pair in which the firefighter and fire are furthest apart are either (f, r') or (f', r). Since both of these are in A_t^* (by the hypothesis of Case 4), the distance between the locations in each of these pairs is at most d. In particular, the distance between the locations of firefighter f' and fire r' is also at most d. In this case define A_{t+1}^* to be A_t^* with (f, r') and (f', r) removed and replaced by (f, r) and (f', r'); i.e., $A_{t+1}^* = (A_t^* - \{(f', r), (f, r')\}) \cup \{(f, r), (f', r')\}$. A_{t+1}^* satisfies the constraints and has the same number of pairs as A_t^*, so it is an optimal set that contains A_{t+1} and we are done. □

It is easy to see that this lemma implies that the assignment A returned by the algorithm is optimal: By the Promising Set Lemma, A is contained in some optimal assignment. By the exit condition of the loop in line 4, there are either no more firefighters or no more fires to be assigned. Therefore, when we exit the loop A is equal to some optimal assignment.

RUNNING TIME. Sorting of the two arrays can be done in $O(m \log m + n \log n)$ time. As argued above, the loop terminates after at most $m + n$ iterations. Clearly each iteration takes $O(1)$ time, so the loop takes $O(m + n)$ time. So, the running time of the algorithm is dominated by the sorting steps, i.e., it is $O(m \log m + n \log n)$. (This can also be written as $O(\max(m, n) \cdot \log \max(m, n)$. Hopefully by now you have mastered the big-oh notation enough to see why this is the case; if not, time to review!)

Answer to Question 2.

a. The statement is false, as shown by the following counterexample:



The (unique) minimum cardinality cover of the above set of intervals is $\{c, d\}$, but these two intervals intersect.

b. ALGORITHM. The algorithm is as follows:

```
    COVER(I)
1   L := set of intervals in I sorted by non-decreasing finish time
2   C := ∅
3   while L ≠ ∅ do
4       I := first interval in L
5       if I does not intersect any interval in C then
6           I' := interval in L that intersects I and has maximum finish time (ties broken arbitrarily)
7           C := C ∪ {I'}
8       L := L − {I}
9   return C
```

The intuition why this is correct is that we "cover" each interval I of $\mathcal{I}$ using the interval that intersects I and extends as far to the right as possible in the hope that this will also cover the maximum number of other intervals, thereby minimizing the number of intervals needed to cover all intervals. This intuition is

more or less correct, but it is incomplete: for example, it does not explain why it is important to sort the intervals by non-decreasing *finish* time as opposed to, say, *start* time. To make sure you understand the proof thoroughly you should: (a) show that the algorithm would not be correct if we sorted the intervals by non-decreasing start time; and (b) pinpoint the place(s) in the proof below where the fact that intervals are sorted by finish time is relevant.

CORRECTNESS. First we note that algorithm terminates, since in each iteration one interval is removed from $\mathcal{I}$ (line 8), so after n iterations $\mathcal{I}$ is empty.

To prove that the set of intervals $\mathcal{C}$ returned by algorithm is optimal, i.e., is a minimum cardinality cover of $\mathcal{I}$, we use the "promising set"approach. Specifically we prove:

Promising Set Lemma. *At the end of each iteration, (a) $\mathcal{C}$ is a cover of the set of intervals removed from $\mathcal{L}$ (i.e., the set of intervals in $\mathcal{I} - \mathcal{L}$), and (b) $\mathcal{C}$ is contained in an optimal cover of $\mathcal{I}$.*

Proof. The proof is by induction on the iteration number. Let $\mathcal{C}_t$ and $\mathcal{L}_t$ be the set of intervals in variables $\mathcal{C}$ and $\mathcal{L}$ at the end of iteration t. (As usual "the end of iteration 0" refers to the point just before the first iteration of the loop.)

The basis is trivially true, since $C_0 = \varnothing$ and nothing has been removed from $\mathcal{L}$. Suppose the invariant holds after iteration t, and let $\mathcal{C}_t^*$ be an optimal set that contains $\mathcal{C}_t$. We will now prove that, if iteration $t+1$ exists, the invariant remains true after iteration $t+1$; that is, (a) $\mathcal{C}_{t+1}$ is a cover of the set of intervals in $\mathcal{I} - \mathcal{L}_{t+1}$, and (b) $\mathcal{C}_{t+1}$ is contained in some optimal cover $\mathcal{C}_{t+1}^*$ of $\mathcal{I}$.

Part (a) is true by the induction hypothesis for (a) and the fact that the interval I deleted from $\mathcal{L}$ in iteration $t+1$ either intersects an interval that is already in $\mathcal{C}_t$ (and therefore in $\mathcal{C}_{t+1}$), or intersects the interval I' added to $\mathcal{C}$ in iteration $t+1$.

For part (b), if no interval is added to $\mathcal{C}$ in iteration $t+1$ (i.e., the condition in line 5 is false) then we simply take $\mathcal{C}_{t+1}^* = \mathcal{C}_t^*$ and we are done by the induction hypothesis. So, suppose that interval $I' = [s', f']$ is added to $\mathcal{C}$ (line 7) and interval $I = [s, f]$ is the interval deleted from $\mathcal{L}$ (line 8) in iteration $t+1$. There are two cases:

CASE 1. $I' \in \mathcal{C}_t^*$. Then we take $\mathcal{C}_{t+1}^* = \mathcal{C}_t^*$ and we are done by the induction hypothesis.

CASE 2. $I' \notin \mathcal{C}_t^*$. Then there is some interval, say $I'' = [s'', f'']$, in $\mathcal{C}_t^*$ that intersects I (otherwise $\mathcal{C}_t^*$ would not be a cover of $\mathcal{I}$, let alone an optimal one). In this case, we take $\mathcal{C}_{t+1}^* = (\mathcal{C}_t^* - \{I''\}) \cup \{I'\}$; that is, we replace I'' by I' in $\mathcal{C}_t^*$. We claim that

$$\mathcal{C}_{t+1}^* \text{ is an optimal cover of } \mathcal{I} \text{ that contains } \mathcal{C}_{t+1}. \qquad (*)$$

To prove this, we first note that $I'' \notin \mathcal{C}_t$, for if it was then I would already be covered and I' would not be added to $\mathcal{C}$ in iteration $t+1$. By induction hypothesis, $\mathcal{C}_t \subseteq \mathcal{C}_t^*$, and since $I'' \notin \mathcal{C}_t$, $\mathcal{C}_t \subseteq \mathcal{C}_t^* - \{I''\}$. So, $\mathcal{C}_{t+1} = \mathcal{C}_t \cup \{I'\} \subseteq (\mathcal{C}_t^* - \{I''\}) \cup \{I'\} = \mathcal{C}_{t+1}^*$. So, $\mathcal{C}_{t+1}^*$ contains $\mathcal{C}_{t+1}$. Furthermore, $\mathcal{C}_{t+1}^*$ contains the same number of intervals as the optimal cover $\mathcal{C}_t^*$ of $\mathcal{I}$. So, to prove (*) it remains to show that $\mathcal{C}_{t+1}^*$ is a cover of $\mathcal{I}$. Since $\mathcal{C}_{t+1}^*$ differs from $\mathcal{C}_t^*$, which is a cover of $\mathcal{I}$, only in that it replaces I'' by I', it suffices to prove that

$$\text{If } J \in \mathcal{I} \text{ intersects } I'' \text{ then it also intersects some interval in } \mathcal{C}_{t+1}^*. \qquad (**)$$

To prove (**), let $J = [s_J, f_J]$ be any interval in $\mathcal{I}$ that intersects I''. If it also intersects I', we are done (since $I' \in \mathcal{C}_{t+1}^*$). So, suppose that J does not intersect I'. Therefore, either (i) $f' < s_J$ or (ii) $f_J < s'$.

- In case (i), we have $f'' \leq f' < s_J$ (the first inequality is because we consider intervals in $\mathcal{I}$ in order of finish time and because I' is chosen as an interval in $\mathcal{L}_t$ that intersects I and has maximum finish time). This contradicts that I'' intersects J, so this case is impossible.

- In case (ii), we have $f_J < s' \leq f$ (the second inequality is because I' intersects I). By part (a) of the induction hypothesis, $\mathcal{C}_t$ covers all intervals that are deleted from $\mathcal{L}_t$, and since $f_J < f$ and we consider intervals by non-decreasing finish time, J is deleted from $\mathcal{L}_t$, so J is covered by $\mathcal{C}_t$. We have shown above that $\mathcal{C}_t \subseteq \mathcal{C}^*_{t+1}$, so J is covered by $\mathcal{C}^*_{t+1}$. This completes the proof of (**), and therefore of the induction step for part (b). □

It is now easy to see that the Promising Set Lemma implies that the set $\mathcal{C}$ returned by the algorithm is an optimal cover of $\mathcal{I}$: By the exit condition of the loop in line 3, all intervals in $\mathcal{I}$ have been deleted from $\mathcal{L}$, so by part (a) of the lemma, $\mathcal{C}$ is a cover of $\mathcal{I}$. By part (b) of the lemma, $\mathcal{C}$ is contained in an optimal cover of $\mathcal{I}$, so it must itself be optimal.

RUNNING TIME. Sorting of the n intervals by finish time can be done in $O(n \log n)$ time. As argued above, the loop terminates after at most n iterations. Determining whether I intersects any interval in $\mathcal{C}$ (line 5) and finding the interval I' (line 6) can each be done in $O(n)$ time. So each iteration of the loop takes $O(n)$ time, and the entire algorithm takes $O(n^2)$ time.