CSE 101- Winter ‘18
Discussion Section
Week 10
Topics for today

Greedy algorithms

➢ All points interval cover problem

Dynamic Programming

➢ Maximum interval cover of weighted points
Interval Problem

Given a set of points $x_1, x_2, \ldots$ on the real line, determine the smallest set of unit-length closed intervals (e.g. the interval $[1.25, 2.25]$ includes all $x_i$ such that $1.25 \leq x_i \leq 2.25$) that contains all of the points.

Give the most efficient algorithm you can to solve this problem, prove it is correct and analyze the time complexity.
How do you do it?

- Sort X
- Begin with no intervals.
- Pick an interval such that it starts at the first (lowest) point which is not already included in one of the picked intervals.
Proof - Exchange Method

Step 1: Represent the solutions.

Let \( \text{OPT} = \{ [o_1, o_1+1], [o_2, o_2+1], \ldots, [o_k, o_k+1] \} \) be the optimal solution in sorted order (based on \( o_i \)).

Let \( \text{G} = \{ [g_1, g_1+1], [g_2, g_2+1], \ldots, [g_m, g_m+1] \} \) be the greedy solution.

Let \( x_1, x_2, \ldots, x_n \) be the points to be covered.
Proof - Exchange Method

Step 2: Make arguments supporting why the elements are swappable.

Since our greedy algorithm starts at the lowest point $x_1$, $g_1 = x_1$. And all the remaining points are greater than $x_1$.

We know that at least one interval in $O$ has to include $x_1$. Let us assume that interval is $[o_j, o_j+1]$. And we know that the max value of $o_j = x_1$.

We know $o_1 <= o_j$, so max value of $o_1 = x_1$. So the max value of end point of the $o_1$’s interval is $x_1 + 1$.

Since all points lie after $x_1$, $[o_1, o_1 + 1]$ can atmost cover all the points in $[x_1, x_1 + 1]$. And points after $x_1 + 1$ are handled by other intervals in $O$. 
Proof - Exchange Method

Step 3: Swap

All points in \([x_1, x_1+1]\) are in \([g_1, g_1 + 1]\). So, all points which could have been included in \([o_1, o_1+1]\) are in \([g_1, g_1 + 1]\). Hence, if we swapped them, we would still cover all the remaining points. And solution will be optimal as the total number of intervals did not change.

Similarly for \(g_2\), and the first point after \(x_1+1\), We can apply the same argument to swap them. Do it until we get our new \(OPT = G\).

We can be sure that there will always exist an optimal interval to swap with as long as there are points after \(g_j+1\), as that point has to be covered by some interval which was not swapped with \([g_j, g_j+1]\)
Proof - Exchange Method

Since the quality of optimal value remained the same throughout the swapping, the new OPT (which is G) is as good as old OPT. Hence G is optimal.
Describe an efficient algorithm that, given a set \( \{x_1, x_2, \ldots, x_n\} \) of points on the real line, where the \( i \)-th point has weight \( w_i \geq 0 \) for \( 1 \leq i \leq n \), selects at most \( k \geq 0 \) unit length closed intervals so that the sum of weights of the points they contain is maximized. Argue that your algorithm is correct. State its complexity.

A unit length interval is an interval of the form \([y, y+1]\) where \( y \) is a real number. A point \( x_i \) is contained in the unit interval \([y, y+1]\) if \( y \leq x_i \leq y+1 \).

Note that one can select any unit interval \([y, y+1]\) that begins at real number \( y \). However, one cannot select more than \( k \) such intervals.
What are we given?
Describe an efficient algorithm that, given a set \( \{x_1, x_2, \ldots, x_n\} \) of points on the real line, where the i-th point has weight \( w_i \geq 0 \) for \( 1 \leq i \leq n \), selects at most \( k \geq 0 \) unit length closed intervals so that the sum of weights of the points they contain is maximized. Argue that your algorithm is correct. State its complexity.

A unit length interval is an interval of the form \([y, y+1]\) where \( y \) is a real number. A point \( x_i \) is contained in the unit interval \([y, y+1]\) if \( y \leq x_i \leq y+1 \).

Note that one can select any unit interval \([y, y+1]\) that begins at real number \( y \). However, one cannot select more than \( k \) such intervals.
Dynamic Programming

Greedy approach like the previous problem does not work? (Why not?)

- Sort by coordinates, and greedily cover as much as possible with k intervals.
- Sort by weights, and greedily cover as much as possible with k intervals.
  - Say you cover the first set of points greedily? With what new point do you restart?

STOP! Think about this before you proceed to the answer.
Dynamic Programming

We do not have to cover all points. We hit points with intervals only to maximize the sum of weights.

Therefore, at each point, we need to make a decision

1. Either hit the point with an interval starting/ending at this point, and solve the rest of the problem with k-1 intervals
2. Skip this point and solve the rest of the problem with k intervals.

Great! Now we have an idea (albeit vague) of what the optimal substructure looks like. Can you come up with a recursive formulation?
Dynamic Programming

We first sort the points based on their position on the real line. Let the new order of points be $x_1 \leq x_2 \leq \ldots \leq x_n$. Let the subproblem be $OPT(i,j)$ where we need to cover points $\{x_1, x_2, \ldots x_i\}$ with $j$ intervals.

$$OPT(n, k) = \max\{OPT(m, k - 1) + T[n], OPT(n - 1, k)\}$$

where $T[n] = \sum_{k=m+1}^{n} w_k$ for points covered by an unit closed interval whose endpoint is at $x_n$. 
Dynamic Programming

We first sort the points based on their position on the real line. Let the new order of points be \(x_1 \leq x_2 \leq \ldots \leq x_n\).

Let the subproblem be \(OPT(i,j)\) where we need to cover points \(\{x_1, x_2, \ldots, x_i\}\) with \(j\) intervals.

\[
OPT(n, k) = \max\{OPT(m, k - 1) + T[n], OPT(n - 1, k)\}
\]

where \(T[n] = \sum_{k=m+1}^{n} w_k\) for points covered by an unit closed interval whose endpoint is at \(x_n\).

Possibility of Precomputation! Use two pointer approach to calculate all values in \(O(n)\) time. (STOP! How?)
Dynamic Programming

\[ OPT(n, k) = \max \{ OPT(m, k - 1) + T[n], OPT(n - 1, k) \} \]

where \( T[n] = \sum_{k=m+1}^{n} w_k \) for points covered by an unit closed interval whose endpoint is at \( x_n \)

Are we done yet? What is wrong?

- We wrote the recursive formulation in terms of \( n,k \), But no base cases given!
Dynamic Programming

$$OPT(i, j) = \max \{OPT(m, j - 1) + T[i], OPT(i - 1, j)\}$$

where $T[i] = \sum_{k=m+1}^{i} w_k$ for points covered by an unit closed interval whose endpoint is at $x_i$

$$OPT(i, 0) = OPT(0, j) = 0$$

We return $OPT(n,k)$ which has the maximum weight of all points of given $n$ points covered by the $k$ intervals.

Note that we also expressed any subproblem $OPT(i,j)$ recursively! This is useful as this helps while writing the inductive proof!
Dynamic Programming

\[ OPT(i, j) = \max\{OPT(m, j - 1) + T[i], OPT(i - 1, j)\} \]

where \( T[i] = \sum_{k=m+1}^{i} w_k \) for points covered by an unit closed interval whose endpoint is at \( x_i \)

\[ OPT(i, 0) = OPT(0, j) = 0 \]

Time complexity:
\( O(nk) \) with the precomputation and memoization.
Dynamic Programming

\[ OPT(i, j) = \max\{OPT(m, j - 1) + T[i], OPT(i - 1, j)\} \]

where \( T[i] = \sum_{k=m+1}^{i} w_k \) for points covered by an unit closed interval whose endpoint is at \( x_i \)

\[ OPT(i, 0) = OPT(0, j) = 0 \]

Proof of correctness:

- Base case are computed correctly (Why?)
- Inductive hypothesis: (In how many variables?)
- Inductive proof

If you were to do this bottom-up, what kind of subproblems are you using? Solved ones!
Problem Variation

Here is a problem variation:
Describe an efficient algorithm that, given a set \{ [s_i, s_{i+1}] \} where the i-th interval has weight \( w_i \geq 0 \) for \( 1 \leq i \leq n \), selects at most \( k \geq 0 \) points on the real line, so that the sum of the weights of the intervals that are hit is maximized. Argue that your algorithm is correct. State its complexity.

A point \( x_j \) hits a unit length interval \([s_i, s_{i+1}]\) if \( x_j \) is contained in the unit interval \([s_i, s_{i+1}]\) if \( s_i \leq x_j \leq s_{i+1} \). However you can only select \( k \) such points. If an interval is hit by more than one point, you can add its weight to the sum only once. The goal is to select \( k \) points so that the sum of weights of the intervals hit by the selected points is maximized. You can assume all \( s_i \) are distinct.

Is this greedy or DP? (Hint: Remember \( s_i \) are real numbers, and therefore intervals could be overlapping)

Solution will be announced on Sunday
Handbooks on Proofs
Proof Methods - Exchange Method

Step 1: Label Your algorithm’s solution and optimal solution.

Eg G = \{a_1, a_2, a_3 ..\} OPT = \{o_1, o_2, o_3 ..\}

Step 2: Assume that OPT is not same as G. then,

There is an element of OPT in G and an element of G not in OPT.

There are 2 consecutive elements in OPT in a different order than they are in G. i.e. there is an inversion.
Proof Methods - Exchange Method

Step 3: Exchange

Swap or exchange the elements in O to make it more similar to A.

In case 1: Swap with the element in G \((o_1 \text{ with } a_1)\)

In case 2: Swap with the element which is in a different order in O. \((o_1 \text{ with } o_2)\)

Both times arguing that the swap does not worsen the quality of the optimal solution.

All the best!