Intermediate Data Structures & Algorithms – CS 141
(Discussion)

Amirali Darvishzadeh
Fall 2015
Consider the following problem:

- **Input**: A set $S = \{(x_i, y_i) | 1 \leq i \leq n\}$ of intervals over the real line.
- **Output**: A maximum cardinality subset $S'$ of $S$ such that no pair of intervals in $S'$ overlap.

Consider the following algorithm:

Repeat until $S$ is empty
1. Select the interval $I$ that overlaps the least number of intervals.
2. Add $I$ to final solution set $S'$.
3. Remove all intervals from $S$ that overlaps with $I$.

Prove or disprove that this algorithm solves the problem.
Question 2

Consider the following problem:

The input is a collection $A = \{a_1, ..., a_n\}$ of $n$ points on the real line. The problem is to find a minimum cardinality collection $S$ of unit intervals that cover every point in $A$. Another way to think about this same problem is the following.

a) Prove or disprove that the following algorithm correctly solves this problem. Let $I$ be the interval that covers the most number of points in $A$. Add $I$ to the solution set $S$. Then recursively continue on the points in $A$ not covered by $I$.

b) Prove or disprove that the following algorithm correctly solves this problem. Let $a_j$ be the smallest (leftmost) point in $A$. Add the interval $I = a_j, a_j + 1$ to the solution set $S$. Then recursively continue on the points in $A$ not covered by $I$. 

b) This algorithm is optimal for the problem of covering points with unit intervals. Assume there is a set of points $A = \{a_1, ..., a_n\}$ such that the greedy algorithm is not optimal

$G = \{g_1, ..., g_n\}$ (greedy solution)

$T = \{t_1, ..., t_n\}$ (optimal solution)

Assume the intervals are numbered in increasing order of left endpoint.
Starting at the left most interval in $T$ compare $g_i$ and $t_i$ until you find $k$ for which $g_k \neq t_k$.

$g_k > t_k$ ($g_k$ begins further to the right than $t_k$)

Create solution $T'$ by replacing interval $t_k$ with $g_k$. Since for $i = 1, ..., k - 1, g_i = t_i$, Solution $T$ will continue to cover all the points in $A$. If $g_k$ overlaps any other interval $t_j$ in $T$, shift $t_j$ to the right until it no longer overlaps $g_k$. Continue shifting intervals in $T'$ to the right until there are no more overlaps. $T'$ continues to cover all points in $A$. By repeating above process we can make $T = G$. Contradicting our assumption that $G$ in not optimal solution.
b) This algorithm is optimal for the problem of covering points with unit intervals. Assume there is a set of points \( A = \{a_1, ..., a_n\} \) such that the greedy algorithm is not optimal.

\[
G = \{g_1, ..., g_n\} \text{ (greedy solution)}
\]
\[
T = \{t_1, ..., t_n\} \text{ (optimal solution)}
\]

Assume the intervals are numbered in increasing order of left endpoint. Starting at the left most interval in \( T \) compare \( g_i \) and \( t_i \) until you find \( k \) for which

\[
g_k > t_k \quad (g_k \text{ begins further to the right than } t_k)
\]

Create solution \( T' \) by replacing interval \( t_k \) with \( g_k \). Since for \( i = 1, ..., k - 1, g_i = t_i \), Solution \( T \) will continue to cover all the points in \( A \). If \( g_k \) overlaps any other interval \( t_j \) in \( T \), shift \( t_j \) to the right until it no longer overlaps \( g_k \). Continue shifting intervals in \( T' \) to the right until there are no more overlaps. \( T' \) continues to cover all points in \( A \). By repeating above process we can make \( T = G \). Contradicting our assumption that \( G \) in not optimal solution.
Question 3

How to compress Image using Huffman Coding technique.
• Each pixel is represented by 8 bits (intensity of pixel)
• \( p_i, 1 \leq p_i \leq 255 \)
• Pixel values that appear a lot we want to assign short code
• Size = \( 500 \times 500 \times 8 = 250 \text{ KB} \)

<table>
<thead>
<tr>
<th>( p_i )</th>
<th>( P_i(p_i) )</th>
<th>Code 1</th>
<th>Size (bits)</th>
<th>Code 2</th>
<th>Size (bits)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( p_{87} = 87 )</td>
<td>0.25</td>
<td>01010111</td>
<td>8</td>
<td>01</td>
<td>2</td>
</tr>
<tr>
<td>( p_{128} = 128 )</td>
<td>0.47</td>
<td>10000000</td>
<td>8</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>( p_{186} = 186 )</td>
<td>0.25</td>
<td>11000100</td>
<td>8</td>
<td>000</td>
<td>3</td>
</tr>
<tr>
<td>( p_{255} = 255 )</td>
<td>0.03</td>
<td>11111111</td>
<td>8</td>
<td>001</td>
<td>3</td>
</tr>
<tr>
<td>( p_k = 0 )</td>
<td>0</td>
<td>-</td>
<td>8</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

K is not 87, 128, 186, 255

Width = 500, Height = 500
Question 3

- At each source reduction iteration, sort probabilities first and then merge two lowest probabilities.
- Keep merging until you end up with only two probabilities at the end.

<table>
<thead>
<tr>
<th>Original source</th>
<th>Source reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Symbol</td>
<td>Probability</td>
</tr>
<tr>
<td>$a_2$</td>
<td>0.4</td>
</tr>
<tr>
<td>$a_6$</td>
<td>0.3</td>
</tr>
<tr>
<td>$a_1$</td>
<td>0.1</td>
</tr>
<tr>
<td>$a_4$</td>
<td>0.1</td>
</tr>
<tr>
<td>$a_3$</td>
<td>0.06</td>
</tr>
<tr>
<td>$a_5$</td>
<td>0.04</td>
</tr>
</tbody>
</table>
Question 3

- Go backwards and assign code
- Every time you see a split, we add 0 and 1 to the codes.
Question 3

- Go backwards and assign code
- Every time you see a split, we add 0 and 1 to the codes.
Question 4

A subsequence is *palindromic* if it is the same whether read left to right or right to left. For the sequence


Has many palindromic subsequence, including \( A, C, G, C, A \) and \( A, A, A, A \).

Devise an algorithm that takes a sequence \( x[1 \ldots n] \) and returns the longest palindromic subsequence. Its running time should be \( O(n^2) \).