Example: (continued from last time)

\[ \begin{align*}
\mathbf{b}^T &= \begin{bmatrix} 3 \ 4 \ 5 \ 6 \end{bmatrix}, \quad \mathbf{b}^T \mathbf{w} = \begin{bmatrix} 3 \ 4 \ 5 \ 6 \end{bmatrix} \begin{bmatrix} 0 \ 1 \ 0 \ 1 \end{bmatrix} \\
\mathbf{b}^T \mathbf{w} &= \begin{bmatrix} 3 \ 4 \ 5 \ 6 \end{bmatrix} \\
\mathbf{b}^T \mathbf{w} &= \begin{bmatrix} 2 \\
\end{bmatrix}
\end{align*} \]

The tableau we get is the same as the one we got as a result of several simplex iterations:

\[
\begin{array}{cc|ccc|c}
X_1 & X_2 & X_3 & X_4 & & 2 \\
1 & 0 & 3 & 1 & & 2 \\
0 & 1 & -1 & 1 & & 2
\end{array}
\]

- Can compute also the coefficients of the objective function:
  - Recall \( x_k - (\ell) \): \( X_0 + \mathbf{b}^T \mathbf{w} X_w = \mathbf{b}^T \mathbf{w} \)
  - Express \( X_0 \) in terms of \( X_w \):
    \[
    X_0 = \mathbf{b}^T \mathbf{w} - \mathbf{b}^T \mathbf{w} X_w
    \]
  - Plug these values of \( X_0 \) into the objective function:
    \[
    f = C_0 X_0 + C X_w = C_0 (\mathbf{b}^T \mathbf{w} - \mathbf{b}^T \mathbf{w} X_w) + C X_w
    \]

Revised Simplex Algorithm

Motivation: Suppose we get a problem with 10 constraints and 10,000,000 variables. To keep the whole simplex tableau in storage, need to keep >100 million real numbers in memory.

Question: Do we really need to keep the whole tableau in storage?

- Suppose at the beginning of an iteration we know just the current basis \( B \) (no simplex tableau is given).
  - 10 constraints \( \Rightarrow \) the basis matrix is \( 10 \times 10 \)
  - first compute \( B^T \)
- To start the iteration, need row 0 to find entering variable \( X_j \)
  - Compute it based on (x) of page 2:
    \[
    C_0 \mathbf{B}^T \mathbf{w} - C
    \]
  - Suppose \( X_j \) is chosen to enter the basis.
  - To determine the leaving variable, we need the column of \( X_j \) and current RHS:

\[
= \begin{bmatrix} C_0 \mathbf{B}^T \mathbf{w} \\
\end{bmatrix} + \begin{bmatrix} C \mathbf{B}^T \mathbf{w} \end{bmatrix} X_w
\]

Ex. (cont.):

\[
C_0 \mathbf{B}^T \mathbf{w} = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} 2 \\
\end{bmatrix} = 4
\]

\[
C \mathbf{B}^T \mathbf{w} = \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\
\end{bmatrix} = \begin{bmatrix} 2 & 2 \\
\end{bmatrix}
\]

These are the same coefficients as in the tableau of

- Summarizing, given a basis, the corresponding simplex tableau can be directly computed from given data:

<table>
<thead>
<tr>
<th>Basic var.</th>
<th>( X_0 )</th>
<th>( X_w )</th>
<th>RHS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_h )</td>
<td>( \mathbf{b}^T \mathbf{w} )</td>
<td>( \mathbf{b}^T \mathbf{w} )</td>
<td></td>
</tr>
</tbody>
</table>

- Based on this observation, a streamlined variation of Simplex (next page)

Again from (x), current column of \( X_j \) is \( B_j \) \( A_j \) \( i \) current RHS is \( B^T \mathbf{w} \).

After min-ratio test we get a new basis and can iterate.

Statement of the algorithm (Revised Simplex)

Start with a BF soln (do Phase 1 if necessary)

1. Compute \( B^T \), \( B^T \mathbf{w} \), \( C_0 B^T \mathbf{w} \)
2. Test optimality (or find entering basic variable)
   - Compute \( C_0 \mathbf{B}^T \mathbf{w} - C \mathbf{w} \)
   - If \( C_0 (\mathbf{B}^T \mathbf{w} - C \mathbf{w}) \geq 0 \) this has an optimal soln. Stop
   - Else determine an entering variable \( X_j \)
3. Compute the pivot column of tableau: \( B_j A_j \)
4. Perform min-ratio test to determine leaving basic variable (or to detect unboundedness)
5. Update the basis and go to step 1.