DEGENERACY IN THE SIMPLEX METHOD
DEGENERACY

\[ z = 2x_1 - x_2 + 8x_3 \]
\[ x_4 = 1 - 2x_3 \]
\[ x_5 = 3 - 2x_1 + 4x_2 - 6x_3 \]
\[ x_6 = 2 + x_1 - 3x_2 - 4x_3 \]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_0 \geq 0 \]

Clearly \( x_3 \) enters the basis, but who leaves? All variables \( x_4, x_5, x_6 \) give the same increase! Choose any!! Say \( x_4 \) pivot.

\[ z = 4 + 2x_1 - x_2 - 4x_4 \]
\[ x_3 = 0.5 - 0.5x_4 \]
\[ x_5 = -2x_1 + 4x_2 + 3x_4 \]
\[ x_6 = x_1 - 3x_2 - 2x_4 \]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_0 \geq 0 \]

NOTE: \( x_5, x_6 \) are basic, but they are also equal to ZERO! DEGENERATE PROBLEM!
This has annoying consequences. For example if we pivot again, \( x_1 \) enters the basis and \( x_5 \) leaves (limit of increment is zero!).

\[
\begin{align*}
    z &= 4 + 3x_2 - x_4 - x_5 \\
x_1 &= 2x_2 + 1.5x_4 - 0.5x_5 \\
x_3 &= 0.5 - 0.5x_4 \\
x_6 &= -x_2 + 3.5x_4 - 0.5x_5 \\
\end{align*}
\]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_0 \geq 0 \]

This does not change the solution at all!!! We did not move!!!

**Definition:** We say a dictionary is degenerate if one or more of the basic variables vanish.

**Definition:** A pivot is degenerate if the objective function value does not change.

Sometimes the simplex method goes through a few degenerate pivots one after the other, sometimes even **CYCLING CAN HAPPEN!!**

Namely, a sequence of pivots that returns to the dictionary from which the cycle began.
We use a concrete pivot rule:
(i) The entering variable will be the nonbasic variable that has the largest coefficient in the $z$-row.
(ii) If two or more basic variables compete for leaving the basis, then the candidate with the smallest subscript will be made to leave.

\[
\begin{align*}
  z &= 10x_1 - 57x_2 - 9x_3 - 24x_4 \\
  x_5 &= -0.5x_1 + 5.5x_2 + 2.5x_3 - 9x_4 \\
  x_6 &= -0.5x_1 + 1.5x_2 + 0.5x_3 - x_4 \\
  x_7 &= 1 - x_1 \\
  x_1, x_2, x_3, x_4, x_5, x_6, x_7 &\geq 0
\end{align*}
\]

What is the next dictionary? Who enters who leaves?

\[
\begin{align*}
  z &= 53x_2 + 41x_3 - 204x_4 - 20x_5 \\
  x_1 &= 11x_2 + 5x_3 - 18x_4 - 2x_3 \\
  x_6 &= -4x_2 - 2x_3 + 8x_4 + x_5 \\
  x_7 &= 1 - 11x_2 - 5x_3 + 18x_4 + 2x_5 \\
  x_1, x_2, x_3, x_4, x_5, x_6, x_7 &\geq 0
\end{align*}
\]
\[ z = 14.5x_3 - 98x_4 - 6.75x_5 - 13.25x_6 \]
\[ x_2 = -0.5x_3 + 2x_4 + 0.25x_5 - 0.25x_6 \]
\[ x_1 = -0.5x_3 + 4x_4 + 0.75x_5 - 2.75x_6 \]
\[ x_7 = 1 + 0.5x_3 - 4x_4 - 0.75x_5 - 13.25x_6 \]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \]

After the third pivot.

\[ z = 18x_4 + 15x_5 - 93x_6 - 29x_1 \]
\[ x_3 = 8x_4 + 1.5x_5 - 5.5x_6 - 2x_1 \]
\[ x_2 = -2x_4 - 0.5x_5 + 2.5x_6 + x_1 \]
\[ x_7 = 1 - x_1 \]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \]
After the fourth pivot:

\[ z = 10.5x_5 - 70.5x_6 - 20x_1 - 9x_2 \]
\[ x_3 = -0.5x_5 + 4.5x_6 + 2x_1 - 4x_2 \]
\[ x_4 = -0.25x_5 + 1.25x_6 + 0.5x_1 - 0.5x_2 \]
\[ x_7 = 1 \]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \]

After the fifth pivot:

\[ z = 24x_6 + 22x_1 - 93x_2 - 21x_3 \]
\[ x_5 = 9x_6 + 4x_1 - 8x_2 - 2x_3 \]
\[ x_4 = -x_6 - 0.5x_1 + 1.5x_2 + 0.5x_3 \]
\[ x_7 = 1 \]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \]

Finally, if we do it one more time we go back to where we started!!!

**QUESTION** How to deal with this potential problem?
Commercial Break:

MATRIX NOTATION
The standard linear program is

- We start with a problem in the form

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad i = 1 \ldots m \\
& \quad x_j \geq 0
\end{align*}
\]

- Turn it into a **DICTIONARY** by adding slack variables (one for each inequality).

\[
\begin{align*}
z = & \quad \sum_{j=1}^{n} c_j x_j \\
x_{n+i} = & \quad b_i - \sum_{j=1}^{n} a_{ij} x_j \quad \text{for} \quad i = 1 \ldots m \\
x_j, & \quad \geq 0
\end{align*}
\]

We can rewrite it using matrices: \( \max c^T x \) subject to \( Ax = b, \ x \geq 0 \) where
\[
A = \begin{bmatrix}
a_{11} & a_{12} & \cdots & a_{1n} & 1 \\
a_{21} & a_{22} & \cdots & a_{2n} & 1 \\
\vdots & \vdots & \ddots & \vdots & \ddots \\
a_{m1} & a_{m2} & \cdots & a_{mn} & 1 
\end{bmatrix},
\quad x = \begin{bmatrix}
x_1 \\
x_2 \\
\vdots \\
x_n \\
x_{n+1} \\
\vdots \\
x_{n+m}
\end{bmatrix},
\quad b = \begin{bmatrix}
b_1 \\
b_2 \\
\vdots \\
b_m
\end{bmatrix}
\]
\[
c = \begin{bmatrix}
c_1 & c_2 & \cdots & c_n & 0 & \cdots & 0
\end{bmatrix}.
\]
- Each iteration the simplex method selects some basic variables \( B \) and nonbasic \( N \).
- We separate the equation \( Ax \) as \( A_Bx_B + A_Nx_N \) where the indices \( B, N \) label the columns involved.
- Thus we can write \( A_Bx_B = b - A_Nx_N \).
- **Lemma** At each step \( A_B \) is invertible (WHY?)
- Multiply both sides by the inverse \( A_B^{-1} \):
  \[
  x_B = A_B^{-1} b - (A_B^{-1})A_Nx_N.
  \]
- To obtain the new objective function row: \( c_Bx_B + c_Nx_N \) substitute blue equation:
  \[
  z = c_B(A_B^{-1} b - (A_B^{-1})A_Nx_N) + c_Nx_N = c_BA_B^{-1} b + (c_N - c_BA_B^{-1}A_N)x_N
  \]
\[ z = c_B A_B^{-1} b + (c_N - c_B A_B^{-1} A_N) x_N = c_B B^{-1} b + (c_N - c_B B^{-1} N) x_N. \]

\[ x_B = A_B^{-1} b - (A_B^{-1}) A_N x_N = B^{-1} b - B^{-1} N x_N. \]

**An Example**

Consider

\[
\begin{align*}
\text{maximize} & \quad 3x_1 + 4x_2 - 2x_3 \\
\text{subject to} & \quad x_1 + 0.5x_2 - 5x_3 \leq 2 \\
& \quad 2x_1 - x_2 + 3x_3 \leq 3 \\
& \quad x_1, x_2, x_3 \geq 0.
\end{align*}
\]

Add slacks (using \( x \)'s for slack variables):

\[
\begin{align*}
x_1 + 0.5x_2 - 5x_3 + x_4 & = 2 \\
2x_1 - x_2 + 3x_3 + x_5 & = 3.
\end{align*}
\]

Cast constraints into matrix notation:

\[
\begin{bmatrix}
1 & 0.5 & -5 & 1 & 0 \\
2 & -1 & 3 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5
\end{bmatrix}
= 
\begin{bmatrix}
2 \\
3
\end{bmatrix}.
\]
Similarly cast objective function:

\[
\begin{bmatrix}
3 \\
4 \\
-2 \\
0 \\
0
\end{bmatrix}^T \begin{bmatrix}
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5
\end{bmatrix}.
\]

\[
Ax = \begin{bmatrix}
x_1 + 0.5x_2 - 5x_3 + x_4 \\
2x_1 - x_2 + 3x_3 + x_5
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0.5x_2 + x_1 - 5x_3 + x_4 \\
-x_2 + x_5 + 2x_1 + 3x_3
\end{bmatrix}
\]

\[
= \begin{bmatrix}
0.5 & 0 \\
-1 & 1
\end{bmatrix} \begin{bmatrix}
x_2 \\
x_5
\end{bmatrix} + \begin{bmatrix}
1 & -5 & 1 \\
2 & 3 & 0
\end{bmatrix} \begin{bmatrix}
x_1 \\
x_3 \\
x_4
\end{bmatrix}
\]

\[
= Bx_B + Nx_N.
\]
\[ B = \begin{bmatrix} 0.5 & 0 \\ -1 & 1 \end{bmatrix} \quad \Rightarrow \quad B^{-1} = \begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix} \]

\[ B^{-1}b = \begin{bmatrix} 4 \\ 7 \end{bmatrix} \]

\[ B^{-1}N = \begin{bmatrix} 2 & 0 \\ 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & -5 & 1 \\ 2 & 3 & 0 \end{bmatrix} = \begin{bmatrix} 2 & -10 & 2 \\ 4 & -7 & 2 \end{bmatrix} \]

\[ (B^{-1}N)^T c_B - c_N = \begin{bmatrix} 2 & 4 \\ -10 & -7 \\ 2 & 2 \end{bmatrix} \begin{bmatrix} 4 \\ 0 \end{bmatrix} - \begin{bmatrix} 3 \\ -2 \\ 0 \end{bmatrix} = \begin{bmatrix} 5 \\ -38 \\ 8 \end{bmatrix} \]

\[ c_B^T B^{-1}b = \begin{bmatrix} 4 & 0 \end{bmatrix} \begin{bmatrix} 4 \\ 7 \end{bmatrix} = 16 \]
Let us compare!! The matrix formulas for the dictionary are

\[ z = c_B B^{-1} b + (c_N - c_B B^{-1} N) x_N. \]

\[ x_B = B^{-1} b - B^{-1} N x_N. \]

Did we get the same answer in our example? (\(x_2\) enters, \(x_4\) leaves).

\[
\zeta = \frac{3x_1 + 4x_2 - 2x_3}{x_4} = 2 - x_1 - 0.5x_2 + 5x_3,
\]
\[
x_5 = 3 - 2x_1 + x_2 - 3x_3.
\]

\[
\zeta = \frac{16 - 5x_1 - 8x_4 + 38x_3}{x_2} = 4 - 2x_1 - 2x_4 + 10x_3,
\]
\[
x_5 = 7 - 4x_1 - 2x_4 + 7x_3.
\]
From matrix notation commercial: A dictionary is uniquely determined by the choice of BASIC variables.

**Proposition** If the simplex method fails to terminate, then it is because it cycled!

**Proof:** Each dictionary is uniquely determined by the choice of BASIC and NON-BASIC variables.

If the problem has $n$ variables, $m$ inequalities, then there are $\binom{n+m}{m}$ possible dictionaries. Thus if it fails to terminate, we must repeat a dictionary!

Every pivot on a cycle must be degenerate!! WHY?

There are two ways to fix the problem: Perturbation and Bland’s Rule

**Definition:** A **Pivot rule** is a method to choose which variable enters and which variable leaves.

**Largest coefficient rule:** Choose the variable with the largest coefficient in the objective function.

**Greatest increase rule:** Pick the entering/leaving pair so as to maximize the increase of the objective function over all possibilities.

**Random selection rule:** Select uniformly at random from the set of possibilities.
Bland’s pivot rule was the first ever pivot rule that did not cycle!!!

First ever pivot rule that guaranteed simplex method will terminate!!!

**Theorem** (Bland 1977) The simplex method terminates as long as the entering and leaving variables are selected by the

**Smallest-Subscript rule (Bland’s rule):** ALWAYS CHOOSE THE CANDIDATE $x_k$ THAT HAS THE SMALLEST SUBSCRIPT $k$
**Corollary** Every LP in the standard form \( \max c^T x \) subject to \( Ax = b, \ x \geq 0 \) has the following three properties:

- If it has no optimal solution, then it is either infeasible or unbounded.
- If it has a feasible solution, then it has a basic feasible solution.
- If it has an optimal solution, then it has a basic optimal solution

**Proof:**

- If the simplex method does not cycle, it must terminate. This can be achieved thanks to Bland’s rule!
- Phase I algorithm either proves that original problem is infeasible OR produces a basic feasible solution.
- Phase II either proves problem is unbounded OR reaches a basic optimal solution.
maximize $x_1 + 2x_2 + 3x_3$
subject to $x_1 + 2x_3 \leq 3$
$x_2 + 2x_3 \leq 2$
$x_1, x_2, x_3 \geq 0.$

maximize $x_1 + 2x_2 + 3x_3$
subject to $x_1 + 2x_3 \leq 2$
$x_2 + 2x_3 \leq 2$
$x_1, x_2, x_3 \geq 0.$
We start with the “cycling” example.

\[ z = 10x_1 - 57x_2 - 9x_3 - 24x_4 \]
\[ x_5 = -0.5x_1 + 5.5x_2 + 2.5x_3 - 9x_4 \]
\[ x_6 = -0.5x_1 + 1.5x_2 + 0.5x_3 - x_4 \]
\[ x_7 = 1 - x_1 \]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \]

We do a perturbation by adding \( \varepsilon \)'s. We think that \( 0 << \varepsilon_3 << \varepsilon_2 << \varepsilon_1 << 1 \).

\[ z = 10x_1 - 57x_2 - 9x_3 - 24x_4 \]
\[ x_5 = \varepsilon_1 - 0.5x_1 + 5.5x_2 + 2.5x_3 - 9x_4 \]
\[ x_6 = \varepsilon_2 - 0.5x_1 + 1.5x_2 + 0.5x_3 - x_4 \]
\[ x_7 = 1 + \varepsilon_3 - x_1 \]

\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \]
Sometimes the best way to think about the $\varepsilon_i$ is that for $0 < \varepsilon < 1$ we set $\varepsilon_i = \varepsilon^i$.

Who is entering now? $x_1$. The constraints that limit the increase of $x_1$ are: $x_5, x_6, x_7$, thus $2\varepsilon_1, 2\varepsilon_2, 1 + \varepsilon_3$.

HOW to tell which is bigger? Use the force... of the **LEXICOGRAPHIC RULE**!

These numbers are like “words” in a dictionary

$$r_0 + r_1\varepsilon_1 + r_2\varepsilon_2 + \cdots + r_m\varepsilon_m, \quad s_0 + s_1\varepsilon_1 + s_2\varepsilon_2 + \cdots + s_m\varepsilon_m$$

$$2 + 21\varepsilon_1 + 19\varepsilon_2 + \cdots + 20\varepsilon_3 < 2 + 21\varepsilon_1 + 20\varepsilon_2 + \cdots + 15\varepsilon_3$$

We can see

$$2\varepsilon_2 < 2\varepsilon_1 < 1 + \varepsilon_3$$

We have the variable $x_6$ leaves!!
\[ z = 20\varepsilon_2 \]
\[ x_1 = 2\varepsilon_2 \]
\[ x_6 = \varepsilon_1 - \varepsilon_2 \]
\[ x_7 = 1 - 2\varepsilon_2 + \varepsilon_3 \]
\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \]

WHO ENTERS? WHO LEAVES? \( x_3 \) enters and \( x_7 \) leaves!

\[ z = 1 + 18\varepsilon_2 + \varepsilon_3 \]
\[ x_3 = 1 - 2\varepsilon_2 + \varepsilon_3 \]
\[ x_1 = 1 + \varepsilon_3 \]
\[ x_5 = 2 + \varepsilon_1 - 5\varepsilon_2 + 2\varepsilon_3 \]
\[ x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0 \]

We have an optimum! This can be converted back to the original dictionary by simply disregarding the terms with \( \varepsilon_i \).
Efficiency

- **Question:** Given a problem of a certain size, how long will it take to solve it? How do you measuring duration? (e.g., Number of iterations, time per iteration, etc).

- Two Kinds of Answers:
  - **Average Case** How long for a random problem.
  - **Worst Case** How long for the hardest problem.

- Performance will depend on the size of the problem!! but ... what is the size of a problem?
  - Number of constraints $m$ and/or number of variables $n$.
  - Size in bytes for the formulation.

- We are looking for a function that describes the number of pivots.

- Some functions are faster than others! Some algorithms are better than others!
## Complexity

<table>
<thead>
<tr>
<th>$n$</th>
<th>$n^2$</th>
<th>$n^3$</th>
<th>$2^n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>8</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>27</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>64</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>125</td>
<td>32</td>
</tr>
<tr>
<td>6</td>
<td>36</td>
<td>216</td>
<td>64</td>
</tr>
<tr>
<td>7</td>
<td>49</td>
<td>343</td>
<td>128</td>
</tr>
<tr>
<td>8</td>
<td>64</td>
<td>512</td>
<td>256</td>
</tr>
<tr>
<td>9</td>
<td>81</td>
<td>729</td>
<td>512</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>1000</td>
<td>1024</td>
</tr>
<tr>
<td>12</td>
<td>144</td>
<td>1728</td>
<td>4096</td>
</tr>
<tr>
<td>14</td>
<td>196</td>
<td>2744</td>
<td>16384</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>4096</td>
<td>65536</td>
</tr>
<tr>
<td>18</td>
<td>324</td>
<td>5832</td>
<td>262144</td>
</tr>
<tr>
<td>20</td>
<td>400</td>
<td>8000</td>
<td>1048576</td>
</tr>
<tr>
<td>22</td>
<td>484</td>
<td>10648</td>
<td>4194304</td>
</tr>
<tr>
<td>24</td>
<td>576</td>
<td>13824</td>
<td>16777216</td>
</tr>
<tr>
<td>26</td>
<td>676</td>
<td>17576</td>
<td>67108864</td>
</tr>
</tbody>
</table>

- **Sorting**: fast algorithm $= n \log n$, slow algorithm $= n^2$
- **Matrix times vector**: $n^2$
- **Matrix times matrix**: $n^3$
- **Matrix inversion**: $n^3$

### Simplex Method:

- **Worst case**: $n^2 2^n$ operations.
- **Average case**: $n^3$ operations.
- **Open question**:
  - Does there exist a variant of the simplex method whose worst case performance is polynomial?

### Linear Programming:

- **Theorem**: There exists an algorithm whose worst case performance is $n^{3.5}$ operations.
Theorem V. Klee and G. Minty (1972) showed that there are explicit linear programs with $n$ constraints for which the largest-coefficient pivot rule can take $2^n - 1$ pivots to reach the optimal solution.

But we know today there are many alternative methods to the SIMPLEX METHOD:

- Fourier-Motzkin Elimination (Goes back to Fourier, rediscovered by T. Motzkin. Interesting in theory but much slower than Simplex)
- Relaxation Methods (invented by T. Motzkin and his students in the 1930’s, 1940’s. Interesting in theory but much slower than Simplex).
- Kachiyan’s Ellipsoid Method (invented in the late 1970’s. First ever polynomial time algorithm for solving linear programs. SLOW!)
- Karmarkar’s Interior Point Methods (invented in the late 1980’s. Good theoretical and practical performance!! Compete with Simplex!).

Many other algorithms exist, but they are mostly variations or improvements of those mentioned so far.