Introduction to Lexicographic Reverse Search: lrs

June 29, 2012

Jayant Apte
ASPITRG

Outline

- Introduction
- Lexicographic Simplex Algorithm
- Lex-positive and Lex min bases
- The pitfalls in reverse search
- Lexicographic Reverse search

Introduction

- Representations of a Polyhedron
- Reverse Search: Some Trivia
- Reverse Search: The high level idea

H-Representation of a Polyhedron

- The halfspace or inequality representation
- Polyhedron $\mathcal{P}$ is the set $x \in \mathbb{R}^n$ obeying a system of linear inequalities i.e.
- $\mathcal{P} := \{x \in \mathbb{R}^n | Hx \leq h\}$
- $H \in \mathbb{R}^{m \times n}, h \in \mathbb{R}^m$ and the inequality is understood to hold elementwise
- Assumption: Polyhedron admits no lines.

V-Representation of a Polyhedron

- $\mathcal{P} = conv\mathcal{S} + cone\mathcal{T}$
- $\mathcal{S}$ is the finite set of extreme points, $\mathcal{T}$ is the finite set of extreme directions(scaled to unit length).
- In other words, any point $x \in \mathcal{P}$ can be represented as,

  $x = \sum_{j=1}^{J} \beta_j s_j + \sum_{k=1}^{K} \gamma_k t_k$
- J is the number of extreme points, K is the number of extreme directions, $\alpha_i \in \mathbb{R}, \beta_j \geq 0, \forall j, \sum_{j=1}^{J} \beta_j = 1, \gamma_k \geq 0.$

Switching between the two representations

- H-representation $\longrightarrow$ V-representation: The Vertex Enumeration problem
- Methods:
  - Reverse Search, Lexicographic Reverse Search
  - Double-description method
- Reverse search is the most robust method

Brief history of Reverse Search

- Presented by David Avis and Komei Fukuda in a 1992 article

  “A Pivoting Algorithm for Convex Hulls and Vertex Enumeration of Arrangements and Polyhedra"

- A C implementation: *rs*
- Modified by Avis in 1998 to use lexicographic pivoting and implemented in rational arithmetic
- A C implementation: *lrs*

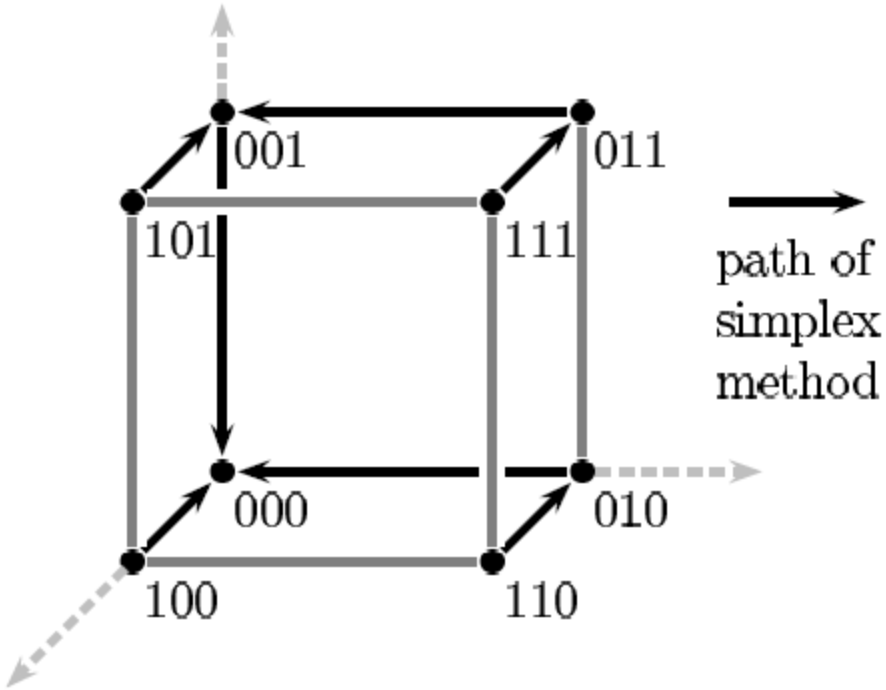
Applications

- Finding the vertices and extreme rays given the H-representation of polyhedron
- Facet enumeration: V-representation to H-representation
- Enumerating all veritces of Voronoi diagram
- Computing volume of convex hull of set of points
- Estimation of number of vertices of polyhedron
- Nash equilibrium in non-cooperative bi-matrix games

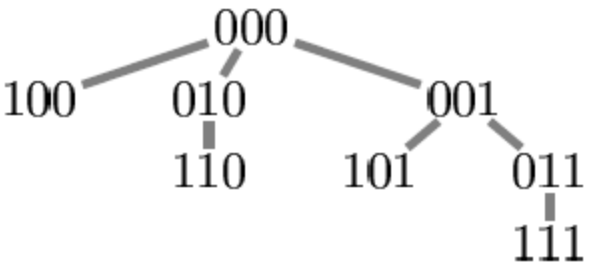
Relationship between Simplex Algorithm and Reverse Search

- Simplex Algorithm is used for solving linear programs
- Simplex Algorithm starts at any vertex of polyhedron defined by the constraint set
- Travels along the edges of polyhedron to find the optimum
- Reverse search runs simplex in reverse
- Traces all possible paths taken by Simplex algorithm in reverse

Relationship between Simplex Algorithm and Reverse Search



(a) The "simplex tree" induced by the objective $(-\sum x_i)$.



(b) The corresponding reverse search tree.

Ref.David Avis, *lrs: A Revised Implementation of the Reverse Search Vertex Enumeration Algorithm*

Background: Lexicographic Order

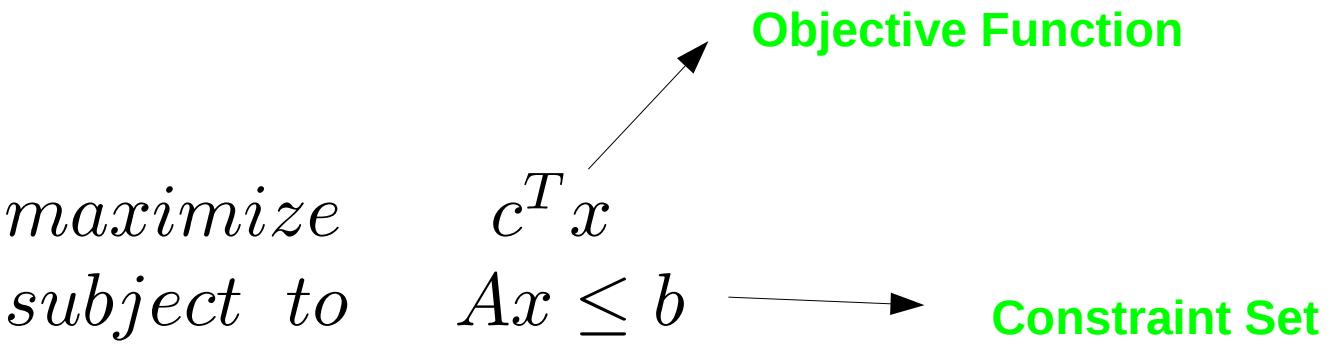
Vectors

- Comparing two vectors in lexicographic sense
  - e.g. [5, 4, 1, 7, 4] $>^{L}$ [5, 4, 0, 9, 8]
- Lex-positive vector
  - First non-zero element is positive
  - e.g. [0, 4, -1, -7, -4], [5, -4, 1, -7, 4] are lexicographically positive vectors
  - e.g. [0 -4 -1 -7 -4], [-5 - 4 1 -7 4] are lexicographically negative vectors
- Lex-min vector
  - Out of [1 2 3 -5 9],[1 2 0 9 9], [2 3 6 -7 1];

    [1 2 0 9 9] is lex-min vector

Lexicographic Simplex Method

The Linear Programming Problem



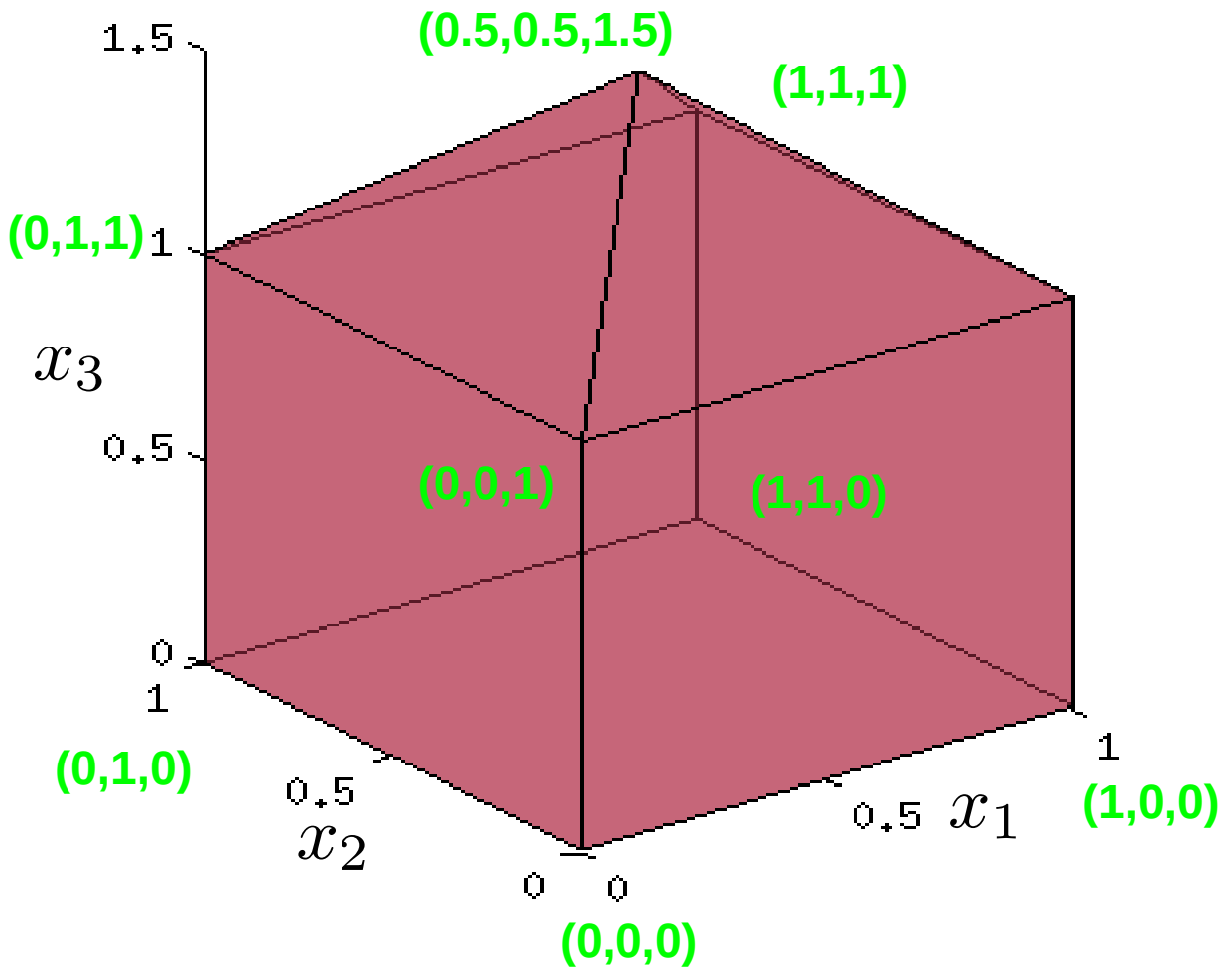
- **d-decision variables**
- **m-constraints**

Example

$$
\begin{aligned}
maximize \quad & -(x_1 + x_2 + x_3) \\
subject\ to \quad & x_1 \leq 1 \\
& x_2 \leq 1 \\
& -x_1 \leq 0 \\
& -x_2 \leq 0 \\
& -x_3 \leq 0 \\
& x_2 + x_3 \leq 2 \\
& -x_2 + x_3 \leq 1 \\
& x_1 + x_3 \leq 2 \\
& -x_1 + x_3 \leq 1
\end{aligned}
\tag{1}
$$

Nature of Constrained Region

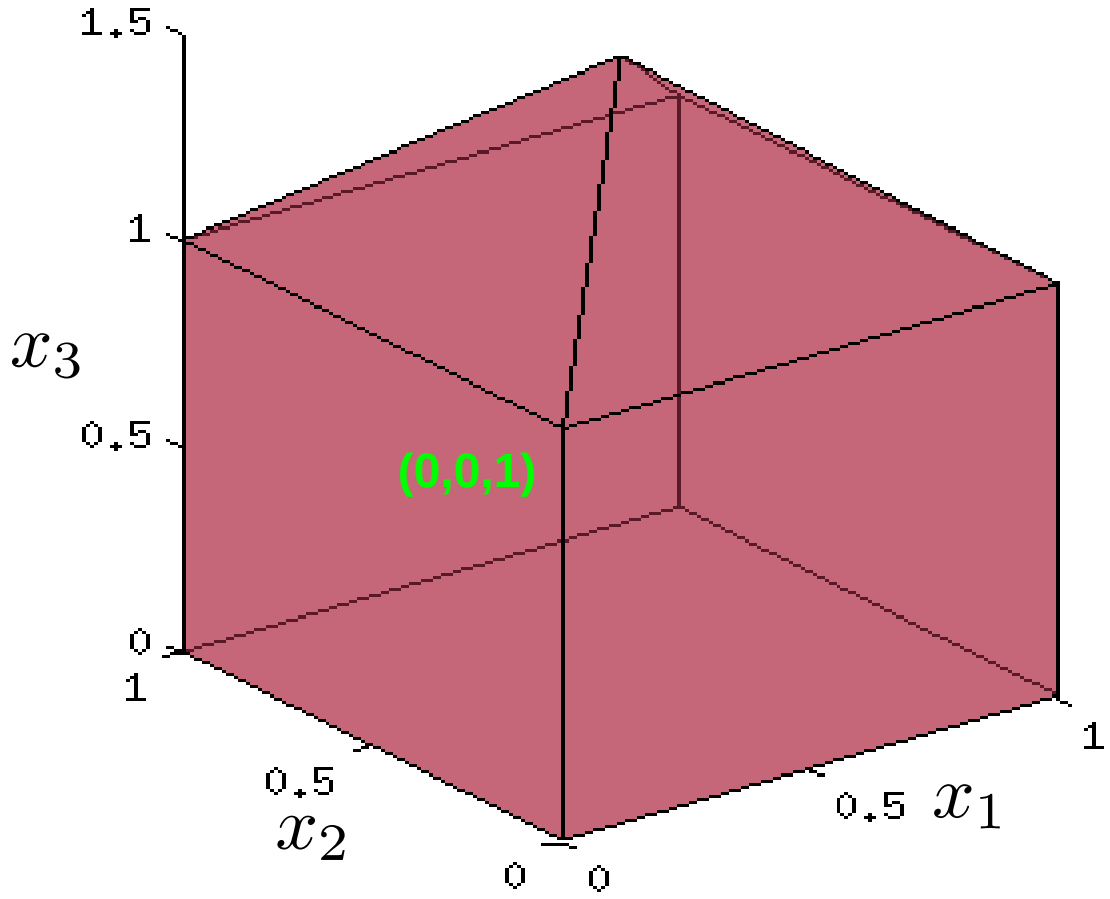
$$
\begin{aligned}
x_1 &\leq 1 \\
x_2 &\leq 1 \\
-x_1 &\leq 0 \\
-x_2 &\leq 0 \\
-x_3 &\leq 0 \\
x_2 + x_3 &\leq 2 \\
-x_2 + x_3 &\leq 1 \\
x_1 + x_3 &\leq 2 \\
-x_1 + x_3 &\leq 1
\end{aligned}
$$



Relationship between inequalities and Vertex: Geometric intuition of Degeneracy

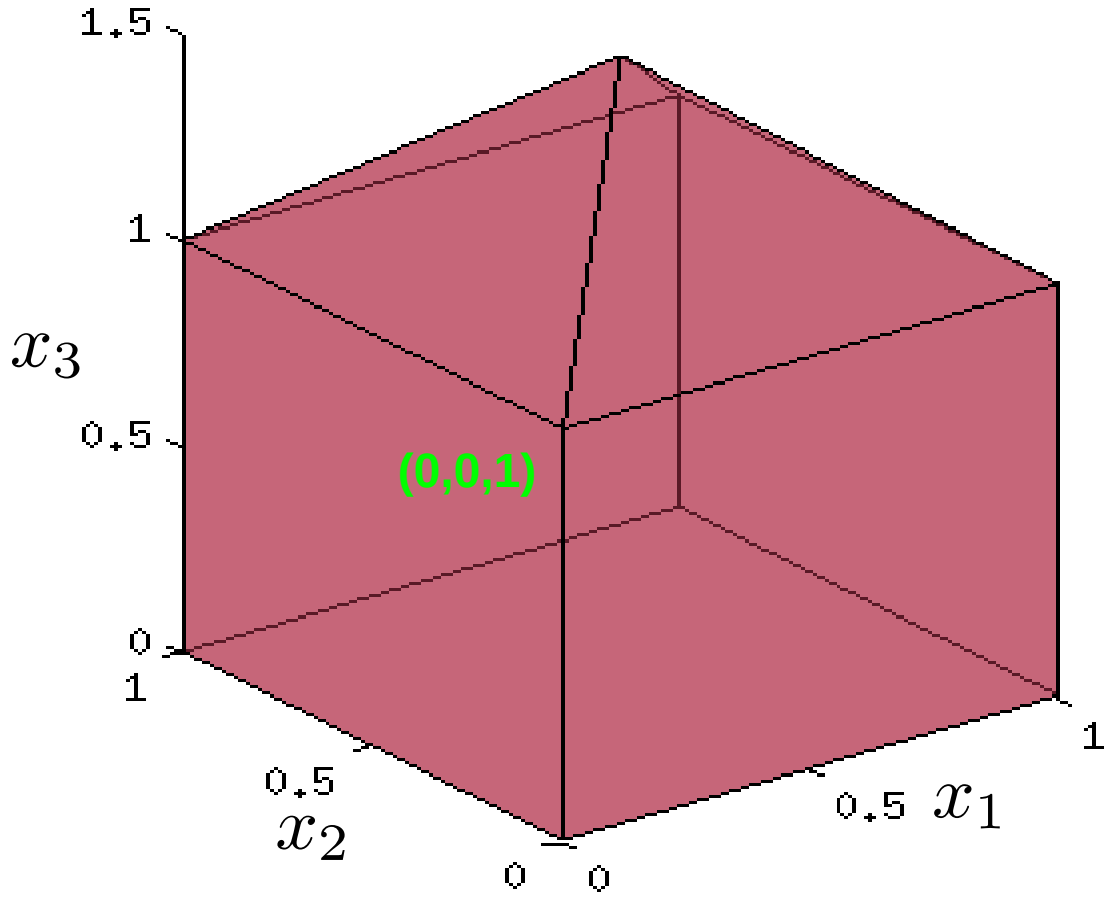
Highlighted inequalities are tight at (0,0,1)

$$x_1 \leq 1$$
$$x_2 \leq 1$$
$$-x_1 \leq 0$$
$$-x_2 \leq 0$$
$$-x_3 \leq 0$$
$$x_2 + x_3 \leq 2$$
$$-x_2 + x_3 \leq 1$$
$$x_1 + x_3 \leq 2$$
$$-x_1 + x_3 \leq 1$$



Relationship between inequalities and Vertex: Geometric intuition of Degeneracy

$$x_1 \leq 1$$
$$x_2 \leq 1$$
$$-x_1 \leq 0$$
$$-x_2 \leq 0$$
$$-x_3 \leq 0$$
$$x_2 + x_3 \leq 2$$
$$-x_2 + x_3 \leq 1$$
$$x_1 + x_3 \leq 2$$
$$-x_1 + x_3 \leq 1$$



Relationship between Co-Basis and Vertex: Geometric intuition of Degeneracy

- (0,0,1) is a degenerate vertex
- There are ${}^4C_3$ ways of specifying the vertex (0,0,1)
- In general this number are ${}^nC_d$ ways of representing a degenerate vertex, where

  n=total number of inequalities that are tight at that vertex

  d=number of dimensions
- This number can get very large very quickly.
- One of the places where simplex and by implication reverse search runs into problems.

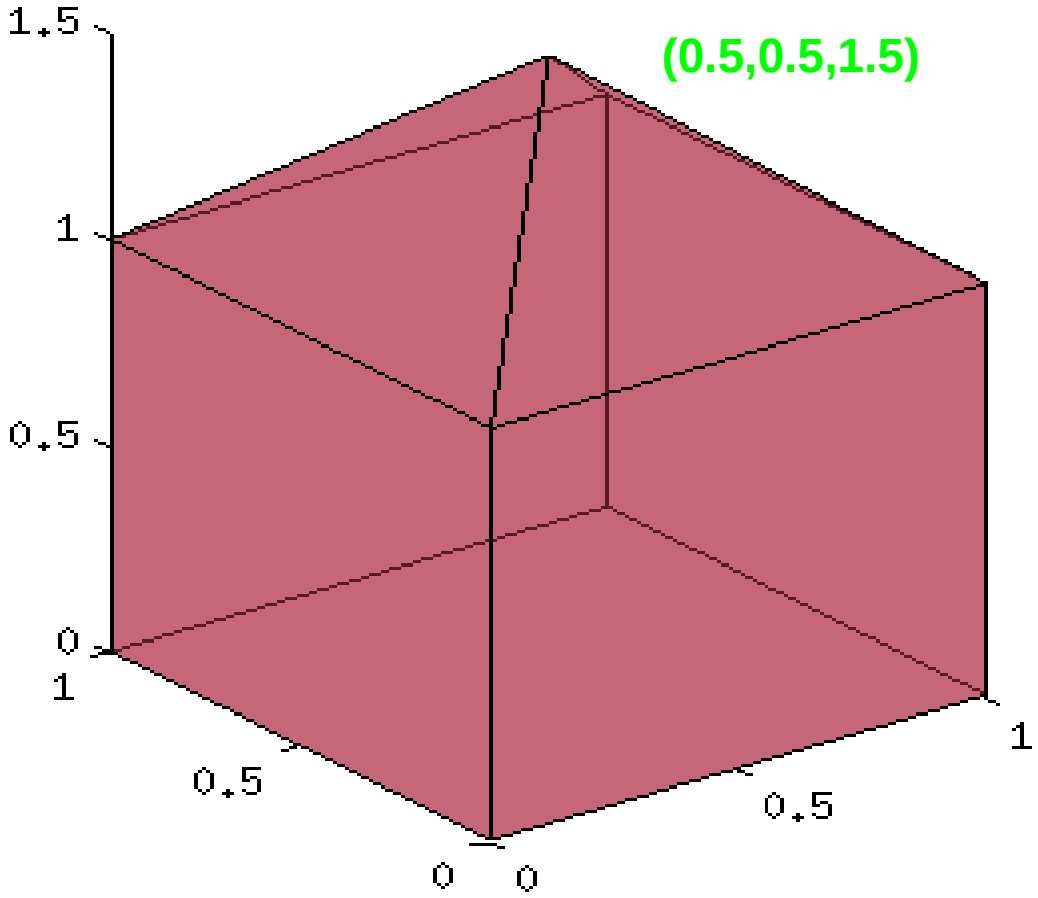
How to solve this LP?

- Rearrange inequalities to decide a starting vertex
- Introduce slack variables
- Convert to matrix form
- Convert to dictionary form

Starting Simplex at a vertex: Rearrangement of the inequalities

- Rearrange the inequalities so that last 'd' inequalities contain initial vertex

$$x_1 \leq 1$$
$$x_2 \leq 1$$
$$-x_1 \leq 0$$
$$-x_2 \leq 0$$
$$-x_3 \leq 0$$
$$x_2 + x_3 \leq 2$$
$$-x_2 + x_3 \leq 1$$
$$x_1 + x_3 \leq 2$$
$$-x_1 + x_3 \leq 1$$



Define Slack Variables

- Convert inequalities to equalities by introducing slack variables
- Original variables are called decision variables

$$\sum_{j=1}^{d} a_{i,j} x_j + x_{d+i} = b_i \qquad i = 1, \ldots m.$$
$$x_{d+i} \geq 0$$

$$A = [\ A \quad I\ ]$$

$$Ax = b$$

Our Example

$$x_1 + x_4 = 1$$
$$x_2 + x_5 = 1$$
$$-x_1 + x_6 = 0$$
$$-x_2 + x_7 = 0$$
$$-x_3 + x_8 = 0$$
$$x_2 + x_3 + x_9 = 2$$
$$-x_2 + x_3 + x_{10} = 1$$
$$x_1 + x_3 + x_{11} = 2$$
$$-x_1 + x_3 + x_{12} = 1$$

Our Example

$$\begin{pmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

Terminology

- *d* decision variables
- *m* constraints
- Hence *m* slack variables
- Basis: *B:* Ordered *m-tuple* indexing set of *m* affinely independent columns of *A*
- $\{1,...\mathrm{d}\} \subset B$
- Co-basis: *N:* Ordered *d-tuple* indexing remaining *d* columns
- A_B : Sub-matrix of A indexed by B

Our example

Basic Solution: $x_B = A_B^{-1} b$ and $x_N = 0$

B is a *feasible basis* and x_B is a *basic feasible solution*(BFS) if

(i) $Ax = b$

(ii) $x_i \geq 0, i \in B, i = d+1, ...d+m.$

We define initial basis as $\mathrm{B}=\{1,2,3,4,5,6,7,8,9\}$

Hence initial co-basis is $\mathrm{N}=\{10,11,12\}$

A_B

$$\begin{pmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A_N

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

b

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

Our example

A_B is invertible

$x_B = A_B^{-1} b, x_N = 0$

$x_B = \{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9\}$

$= \{0.5, 0.5, 1.5, 0.5, 0.5, 1.5, 0.5, 0.5, 0\}$

$x_N = \{x_{10}, x_{11}, x_{12}\} = \{0, 0, 0\}$

x_B Is a Basic Feasible Solution(BFS)

A_B

$$\begin{pmatrix} -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

A_N

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

b

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

Add a cost row

- Add a cost row at the top

$$x_0 = \sum_{j=1}^{m+d} a_{0,j} x_j$$

$$x_0 - \sum_{j=1}^{m+d} a_{0,j} x_j = 0 \longrightarrow \mathrm{Ax{=}b}$$

Our example

$$x_0 + (x_1 + x_2 + x_3) = 0$$

0	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	1	0	0	0	0	0	0	0	0	0
0	-1	0	0	1	0	0	0	0	0	0	0	0
0	0	-1	0	0	1	0	0	0	0	0	0	0
0	0	0	-1	0	0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0	1	0	0	0	0
0	0	1	1	0	0	0	0	0	1	0	0	0
0	0	-1	1	0	0	0	0	0	0	1	0	0
0	1	0	1	0	0	0	0	0	0	0	1	0
0	-1	0	1	0	0	0	0	0	0	0	0	1

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 2 \\ 1 \\ 2 \\ 1 \end{bmatrix}$$

Converting Linear program to the dictionary format

Derive the dictionary from matrix form

$Ax = b$ can be written as:

$$A_B x_B + A_N x_N = b$$
$$x_B + A_B^{-1} A_N x_N = A_B^{-1} b$$

Similarly, $x_0 = cx$ can be written as,

$$x_0 = c_B x_B + c_N x_N$$
$$= c_B(A_B^{-1} b - A_B^{-1} A_N x_N) + c_N x_N$$
$$x_0 - (c_N - c_B A_B^{-1} A_N) x_N = c_B A_B^{-1} b$$

Our example

$$x_0 - (c_N - c_B A_B^{-1} A_N) x_N = c_B A_B^{-1} b$$

$$x_0 + 1x_{10} - 1.5x_{11} - 0.5x_{12} = -2.5$$

Cost function value
At current vertex

$$\mathrm{x}_B \qquad + \mathrm{A}_B^{-1} A_N x_N \qquad = \mathrm{A}_B^{-1} b$$

$$\begin{aligned}
x_1 + \quad & 0x_{10} + 0.5x_{11} - 0.5x_{12} = 0.5 \\
x_2 - \quad & 1x_{10} + 0.5x_{11} + 0.5x_{12} = 0.5 \\
x_3 + \quad & 0x_{10} + 0.5x_{11} + 0.5x_{12} = 1.5 \\
x_4 + \quad & 0x_{10} + 0.5x_{11} - 0.5x_{12} = 0.5 \\
x_5 - \quad & 1x_{10} + 0.5x_{11} + 0.5x_{12} = 0.5 \\
x_6 + \quad & 0x_{10} + 0.5x_{11} + 0.5x_{12} = 1.5 \\
x_7 + \quad & 0x_{10} - 0.5x_{11} + 0.5x_{12} = 0.5 \\
x_8 + \quad & 1x_{10} - 0.5x_{11} - 0.5x_{12} = 0.5 \\
x_9 + \quad & 1x_{10} - \quad 1x_{11} - \quad 1x_{12} = 0
\end{aligned}$$

The vertex

Dictionary Format

Change in Cost Corresponding to each Co-basic Variable

Cost Corresponding To Dictionary

Basic Indices

Co-Basic Indices

$$\begin{bmatrix} 1 & 0_{1\times m} & -\left(c_N - c_B A_B^{-1} A_N\right) & c_B A_B^{-1} b \\ 0_{m\times 1} & I_m & A_B^{-1} A_N & A_B^{-1} b \end{bmatrix}$$

Solution Corresponding To Dictionary

Redefine a bit

Basis: *B:* Ordered *m+1-tuple* indexing set of *m+1* affinely independent columns of dictionary & $\{0,1,...\mathrm{d}\} \subset B$

Dictionary is to serve as starting point

$$\begin{bmatrix} 1 & 0_{1\times m} & -(c_N - c_B A_B^{-1} A_N) & c_B A_B^{-1} b \\ 0_{m\times 1} & I_m & A_B^{-1} A_N & A_B^{-1} b \end{bmatrix}$$

Call this the solution of the initial dictionary or the initial $\mathrm{A}_B^{-1} b$

Call this the initial A_B^{-1} implying that initial basis is identity matrix

Our example

Basis	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	0	0	0	0	0	0	0	0	0	1	-1.5	-0.5	-2.5
1	0	1	0	0	0	0	0	0	0	0	0	0.5	-0.5	0.5
2	0	0	1	0	0	0	0	0	0	0	-1	0.5	0.5	0.5
3	0	0	0	1	0	0	0	0	0	0	0	0.5	0.5	1.5
4	0	0	0	0	1	0	0	0	0	0	0	0.5	-0.5	0.5
5	0	0	0	0	0	1	0	0	0	0	-1	0.5	0.5	0.5
6	0	0	0	0	0	0	1	0	0	0	0	0.5	0.5	1.5
7	0	0	0	0	0	0	0	1	0	0	0	-0.5	0.5	0.5
8	0	0	0	0	0	0	0	0	1	0	1	-0.5	-0.5	0.5
9	0	0	0	0	0	0	0	0	0	1	1	-1	-1	0

Cost

The vertex

Simplex Algorithm: The Ingredients

- It travels from one vertex to another, or more specifically one BFS to another BFS
- For that it uses pivoting operation to obtain the dictionary corresponding to new BFS.
- A co-basic variable assumes a nonzero value and is said to 'enter' the basis
- A basic variable 'leaves' the basis
- We get a new dictionary which is 'equivalent' to current dictionary in algebraic terms, i.e. It carries the same solution
- This new dictionary is obtained by means of 'pivot operation'.

Choosing the entering variable

- Choose the entering variable that brings about an increase in cost

$$x_0 = (c_N - c_B A_B^{-1} A_N) x_N + c_B A_B^{-1} b$$

One of these is supposed to assume a nonzero value

- Only a negative coefficient at $-(c_N - c_B A_B^{-1} A_N)$ will do as it will mean that $(c_N - c_B A_B^{-1} A_N)$ is positive and results in an increase in cost
- If there are many such co-basic variables, choose the one with least subscript
- If there are no such co-basic variables, current dictionary is optimal and current BFS is the optimum

Choosing entering variable

Basis	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	0	0	0	0	0	0	0	0	0	1	-1.5	-0.5	-2.5
1	0	1	0	0	0	0	0	0	0	0	0	0.5	-0.5	0.5
2	0	0	1	0	0	0	0	0	0	0	-1	0.5	0.5	0.5
3	0	0	0	1	0	0	0	0	0	0	0	0.5	0.5	1.5
4	0	0	0	0	1	0	0	0	0	0	0	0.5	-0.5	0.5
5	0	0	0	0	0	1	0	0	0	0	-1	0.5	0.5	0.5
6	0	0	0	0	0	0	1	0	0	0	0	0.5	0.5	1.5
7	0	0	0	0	0	0	0	1	0	0	0	-0.5	0.5	0.5
8	0	0	0	0	0	0	0	0	1	0	1	-0.5	-0.5	0.5
9	0	0	0	0	0	0	0	0	0	1	1	-1	-1	0

Choosing the leaving variable

- Recall lexicographic order
- Construct following matrix:

$$\mathrm{D} = \begin{bmatrix} A_B^{-1}b & A_B^{-1} \end{bmatrix} = \begin{bmatrix} \beta_0 & \alpha_{0,1} & \cdots & \alpha_{0,m} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \beta_m & \alpha_{m,1} & \cdots & \alpha_{m,m} \end{bmatrix}$$

Choosing the leaving variable

- Let $s \in N$ and consider the column $A_B^{-1} A_s$. If

$a_i \leq 0 \quad \forall d+1 \leq i \leq m$ this column defines an extreme ray.

- Otherwise, let t be the index such that D^t / a_t is lexicographically minimum vector of:

$$\{ \frac{D^i}{a_i} : \quad a_i \geq 0, d+1 \leq i \leq m \}$$

- Such a minimum is unique as D has full row rank
- This notation can be abbreviated as,

$$r = lexminratio(B, s)$$

- In case there is no such r we set $r = 0$

Choosing the Leaving Variable

$A_B^{-1} A_{11}$

Basis	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	0	0	0	0	0	0	0	0	0	1	-1.5	-0.5	-2.5
1	0	1	0	0	0	0	0	0	0	0	0	0.5	-0.5	0.5
2	0	0	1	0	0	0	0	0	0	0	-1	0.5	0.5	0.5
3	0	0	0	1	0	0	0	0	0	0	0	0.5	0.5	1.5
4	0	0	0	0	1	0	0	0	0	0	0	0.5	-0.5	0.5
5	0	0	0	0	0	1	0	0	0	0	-1	0.5	0.5	0.5
6	0	0	0	0	0	0	1	0	0	0	0	0.5	0.5	1.5
7	0	0	0	0	0	0	0	1	0	0	0	-0.5	0.5	0.5
8	0	0	0	0	0	0	0	0	1	0	1	-0.5	-0.5	0.5
9	0	0	0	0	0	0	0	0	0	1	1	-1	-1	0

Lexicographic choice of leaving variable

$$D = \begin{bmatrix} A_B^{-1}b & A_B^{-1} \end{bmatrix}$$

$$D^4 = [0.5\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]$$

$$D^5 = [0.5\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0]$$

$$D^6 = [1.5\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0]$$

$$D^4/a_4 = [1\ 0\ 0\ 0\ 0\ 2\ 0\ 0\ 0\ 0\ 0]$$

$$D^5/a_5 = [1\ 0\ 0\ 0\ 0\ 0\ 2\ 0\ 0\ 0\ 0]$$

$$D^6/a_6 = [3\ 0\ 0\ 0\ 0\ 0\ 0\ 2\ 0\ 0\ 0]$$

$$t = 5$$

$$x_5 \text{ Leaves the basis}$$

Where to get A_B^{-1} after each successive iteration?

- Let $\mathrm{A}_{B^{l-1}}^{-1}$ be current basis inverse and $\mathrm{A}_{B^l}^{-1}$ be the next basis inverse
- B^l and B^{l-1} differ only in 1 column
- We can take advantage of this fact and write

$$[A_{B^l}]^{-1} = E_l[A_{B^{l-1}}]^{-1}$$

- Pre-multiplication by elementary matrix E_l corresponds to row operations
- Hence, we get new basis inverse by some row operations

The pivot Operation

A pivot between two bases B and $\bar{B}$ is defined
by indices $r \in B$ and $s \in N$ by setting $\bar{B} = B - r + s$
Procedure: For $a = A_B^{-1} A_s$,
(a) Divide row t of dictionary by
a_t (a_t is t^{th} element in a))
(b) From rest of the rows i subtract a_i times the
row obtained from (a)
(c) Pivot is feasible if both x_B and $x_{\bar{B}}$ are feasible.
For convenience we call it $pivot(r, s), r \in B, s \in N$

This is how to get A_B^{-1} after each successive iteration

- E_l is given as:

$$E_l = [U_0, \ldots, U_{t-1}, \eta_l, U_{t+1}, \ldots, U_m]$$
$$\eta_{il} = -a_i/a_t, \quad i \neq t$$
$$\eta_{rl} = 1/a_t, \quad i = t$$
$$where \quad a_t \neq 0,$$
$$and \;\; a \;\; is \;\; given \;\; as$$
$$a = [A_{B^{l-1}}]^{-1} A_s$$

This is how to get A_B^{-1} after each successive iteration

$$\mathrm{E}_1 = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & 1/a_t & & \vdots \\ \vdots & & & \ddots & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

Row multiplication Operation
•Row corresponding to pivot element is divided by pivot element

To perform $R_i - a_i \times R_t$ multiply by:

$$\mathrm{E}_i = \begin{bmatrix} 1 & 0 & \dots & \dots & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & 1 & & \vdots \\ \vdots & & -a_i & \ddots & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

Index (i, t)

$$\mathrm{E}_l = E_m \times E_{m-1} \times \ldots \times E_1$$

Hence pivot operation automatically yields new basis inverse

Result of pivot (5,11)

Basis:0 1 2 3 4 11 6 7 8 9 Co-Basis:10 5 12

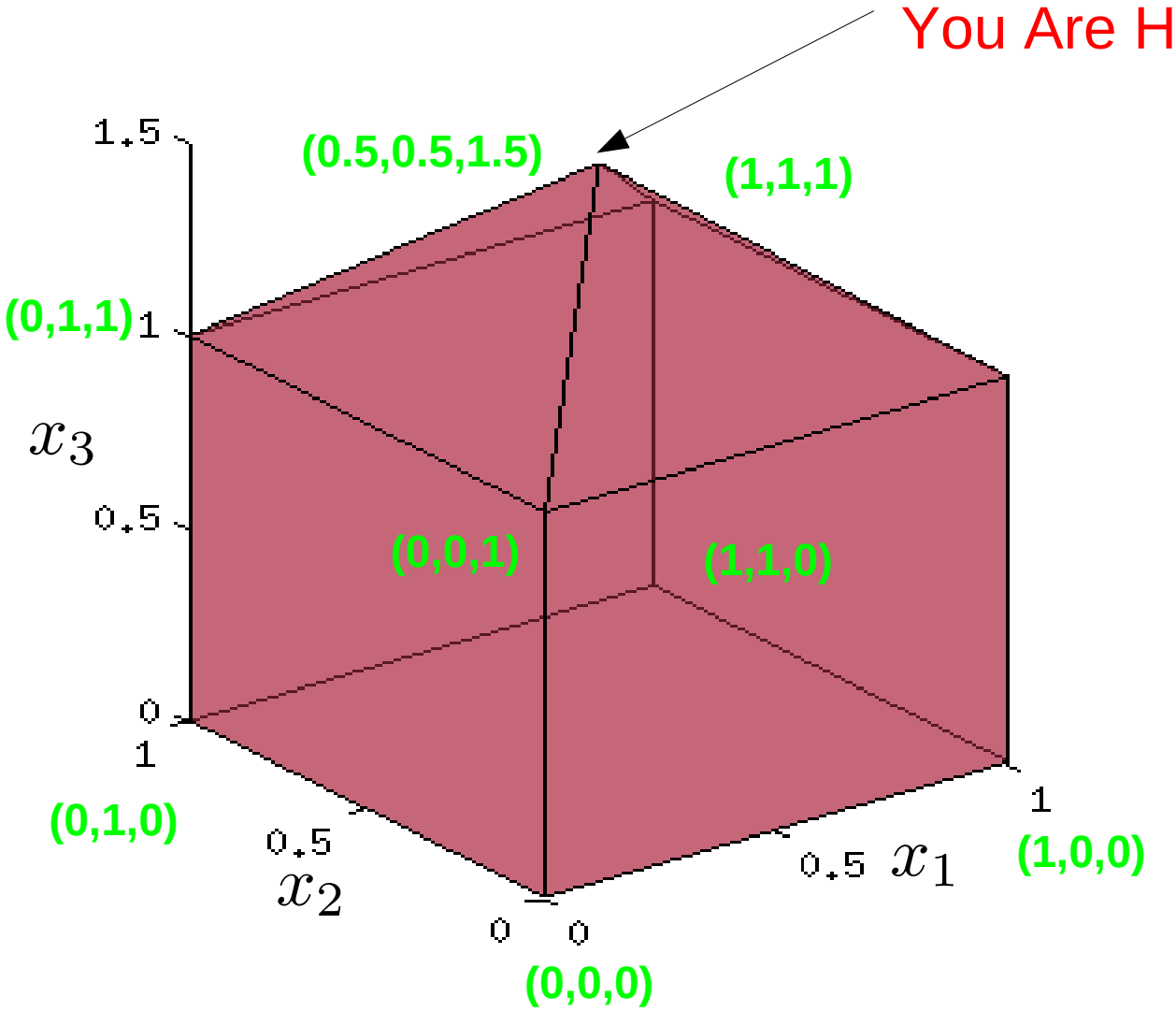
Basis	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	0	0	0	0	3	0	0	0	0	-2	0	1	-1
1	0	1	0	0	0	-1	0	0	0	0	1	0	-1	0
2	0	0	1	0	0	-1	0	0	0	0	0	0	0	0
3	0	0	0	1	0	-1	0	0	0	0	1	0	0	1
4	0	0	0	0	1	-1	0	0	0	0	1	0	-1	0
11	0	0	0	0	0	2	0	0	0	0	-2	1	1	1
6	0	0	0	0	0	-1	1	0	0	0	1	0	0	1
7	0	0	0	0	0	1	0	1	0	0	-1	0	1	1
8	0	0	0	0	0	1	0	0	1	0	0	0	0	1
9	0	0	0	0	0	2	0	0	0	1	-1	0	0	1

Identity column

Solving Linear Programs:
Simplex Algorithm

1. From $-(c_N - c_B A_B^{-1} A_N)$, choose the **entering variable** If none is found, current dictionary is optimal and first column has optimal BFS and optimal cost.
2. Choose the **leaving variable**. If none is found, problem is unbounded.
3. Update dictionary by the **pivot operation**.

Iteration 1



Iteration 1

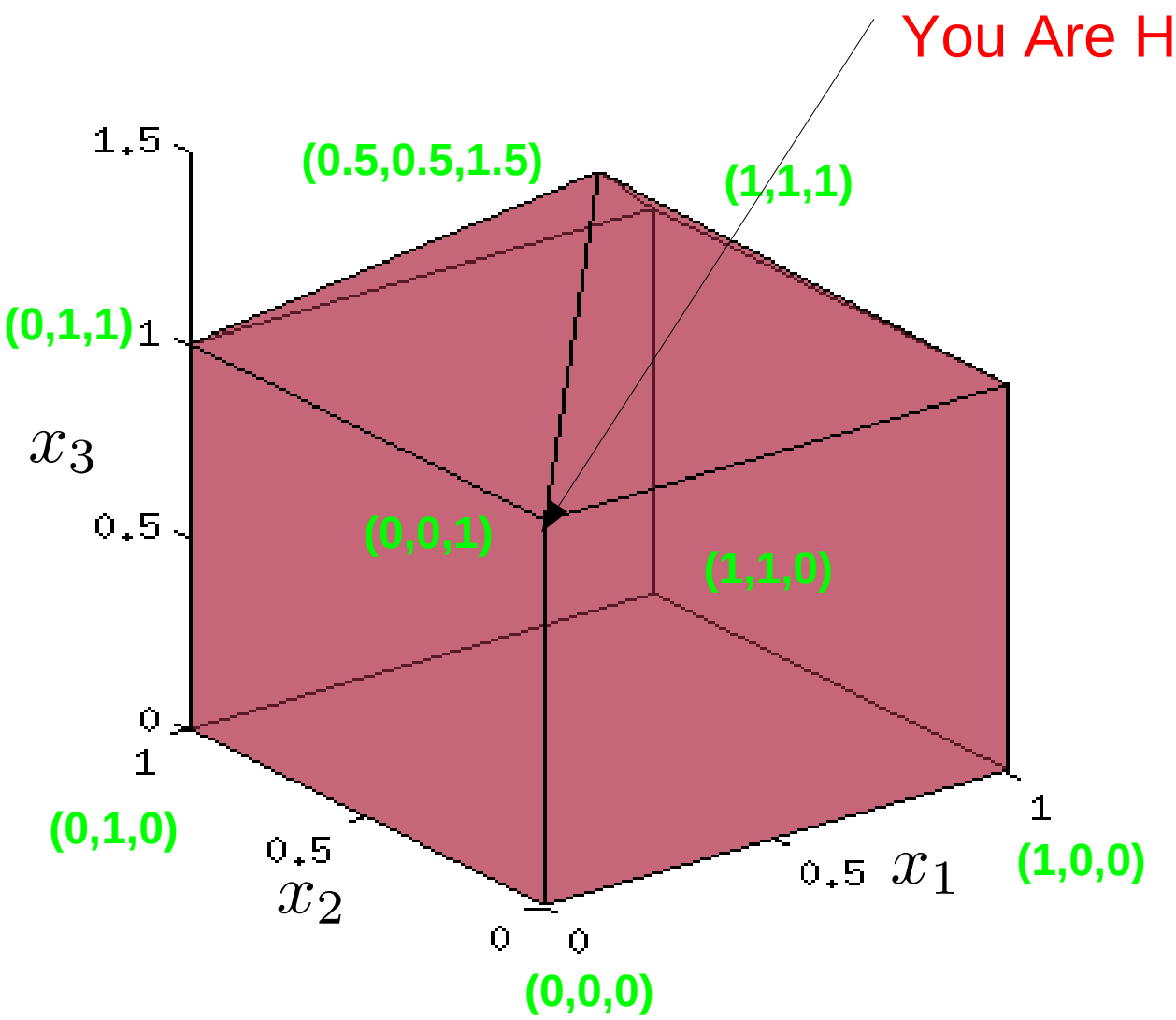
Basis	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	0	0	0	0	0	0	0	0	0	1	-1.5	-0.5	-2.5
1	0	1	0	0	0	0	0	0	0	0	0	0.5	-0.5	0.5
2	0	0	1	0	0	0	0	0	0	0	-1	0.5	0.5	0.5
3	0	0	0	1	0	0	0	0	0	0	0	0.5	0.5	1.5
4	0	0	0	0	1	0	0	0	0	0	0	0.5	-0.5	0.5
5	0	0	0	0	0	1	0	0	0	0	-1	0.5	0.5	0.5
6	0	0	0	0	0	0	1	0	0	0	0	0.5	0.5	1.5
7	0	0	0	0	0	0	0	1	0	0	0	-0.5	0.5	0.5
8	0	0	0	0	0	0	0	0	1	0	1	-0.5	-0.5	0.5
9	0	0	0	0	0	0	0	0	0	1	1	-1	-1	0

Cost

The vertex

Pivot(5,11)

Iteration 2



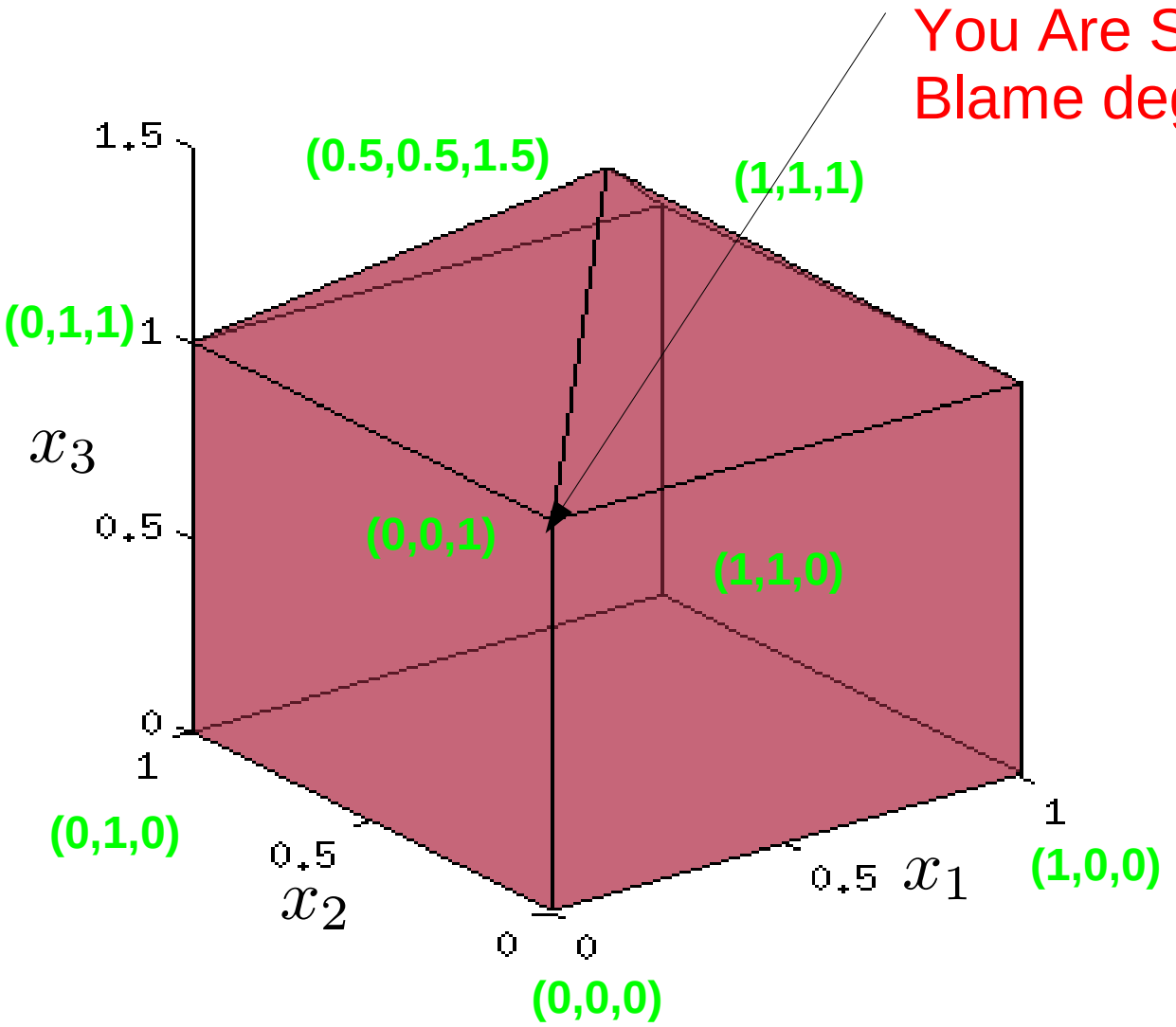
Iteration 2

Basis:0 1 2 3 4 11 6 7 8 9 Co-Basis:10 5 12

Basis	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	0	0	0	0	3	0	0	0	0	-2	0	1	-1
1	0	1	0	0	0	-1	0	0	0	0	1	0	-1	0
2	0	0	1	0	0	-1	0	0	0	0	0	0	0	0
3	0	0	0	1	0	-1	0	0	0	0	1	0	0	1
4	0	0	0	0	1	-1	0	0	0	0	1	0	-1	0
11	0	0	0	0	0	2	0	0	0	0	-2	1	1	1
6	0	0	0	0	0	-1	1	0	0	0	1	0	0	1
7	0	0	0	0	0	1	0	1	0	0	-1	0	1	1
8	0	0	0	0	0	1	0	0	1	0	0	0	0	1
9	0	0	0	0	0	2	0	0	0	1	-1	0	0	1

pivot(4,10)

Iteration 3


You Are Still Here.
Blame degeneracy!
(0.5,0.5,1.5)
(1,1,1)
(0,1,1)
(0,0,1)
(1,1,0)
(0,1,0)
(1,0,0)
(0,0,0)
1.5
1
0.5
0
x_3
x_2
x_1

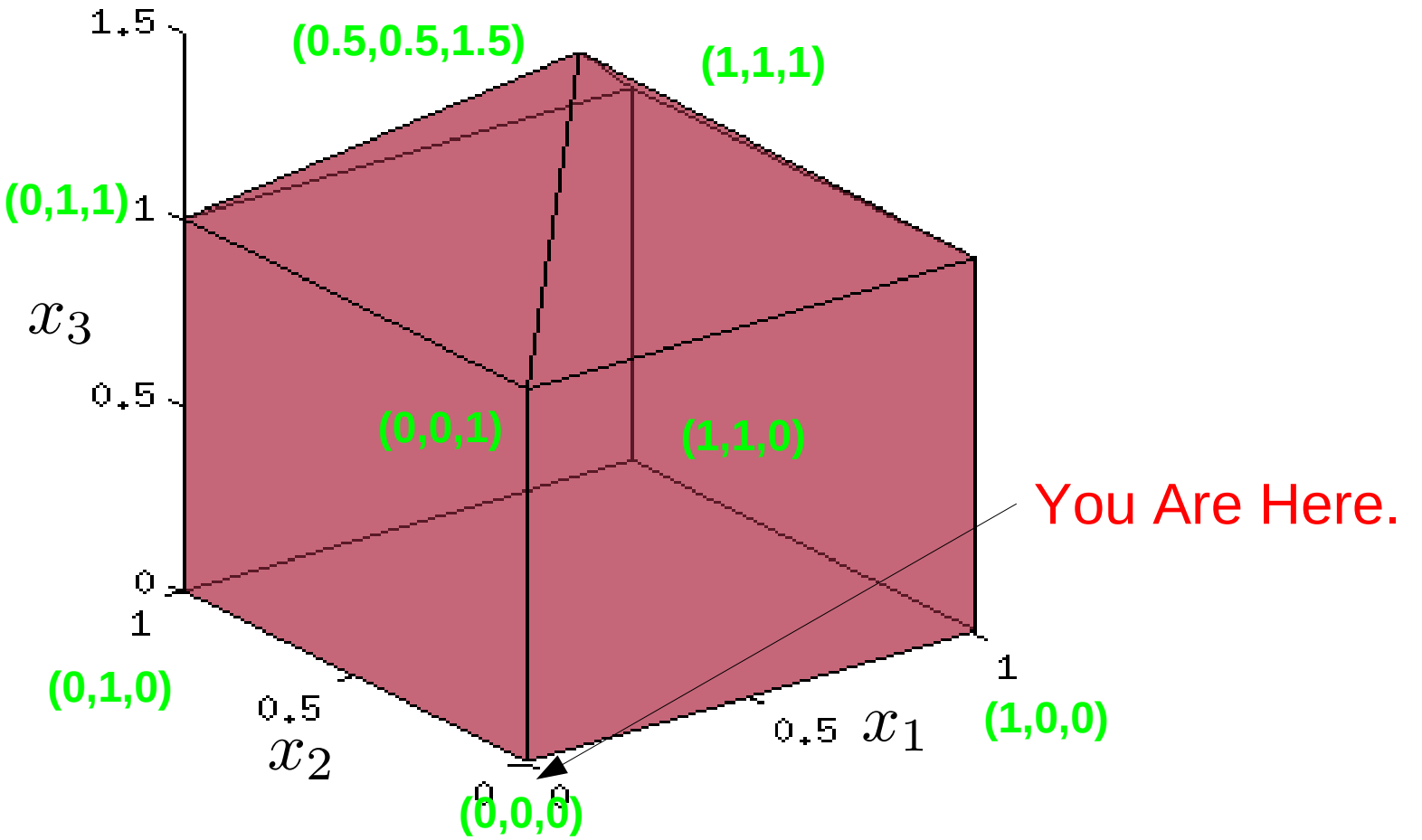
Iteration 3

Basis:0 1 2 3 10 11 6 7 8 9 Co-Basis:4 5 12

Basis	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	0	0	0	2	1	0	0	0	0	0	0	-1	-1
1	0	1	0	0	-1	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	-1	0	0	0	0	0	0	0	0
3	0	0	0	1	-1	0	0	0	0	0	0	0	1	1
10	0	0	0	0	1	-1	0	0	0	0	1	0	-1	0
11	0	0	0	0	2	0	0	0	0	0	0	1	-1	1
6	0	0	0	0	-1	0	1	0	0	0	0	0	1	1
7	0	0	0	0	1	0	0	1	0	0	0	0	0	1
8	0	0	0	0	0	1	0	0	1	0	0	0	0	1
9	0	0	0	0	1	1	0	0	0	1	0	0	-1	1

pivot(6,12)

Iteration 4


1.5
(0.5,0.5,1.5)
(1,1,1)
(0,1,1)
1
x_3
0.5
(0,0,1)
(1,1,0)
You Are Here.
0
1
(0,1,0)
0.5
x_2
0
0
(0,0,0)
0.5
x_1
1
(1,0,0)

Iteration 4

Basis:0 1 2 3 10 11 12 7 8 9 Co-Basis:4 5 6

Basis	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	1	0	0	0	1	1	1	0	0	0	0	0	0	0
1	0	1	0	0	-1	0	0	0	0	0	0	0	0	0
2	0	0	1	0	0	-1	0	0	0	0	0	0	0	0
3	0	0	0	1	0	0	-1	0	0	0	0	0	0	0
10	0	0	0	0	0	-1	1	0	0	0	1	0	0	1
11	0	0	0	0	1	0	1	0	0	0	0	1	0	2
12	0	0	0	0	-1	0	1	0	0	0	0	0	1	1
7	0	0	0	0	1	0	0	1	0	0	0	0	0	1
8	0	0	0	0	0	1	0	0	1	0	0	0	0	1
9	0	0	0	0	0	1	1	0	0	1	0	0	0	2

Observations

- Lexicographic simplex method doesn't travel through all the bases corresponding to a vertex
  - It only visits bases that are lex-positive
  - Recall $D = \begin{bmatrix} A_B^{-1}b & A_B^{-1} \end{bmatrix}$
  - A basis is lex-positive if the rows indexed d+1,...,m of D are lex-positive
- $\text{If } B \text{ is a lex positive Basis, } \bar{B} = B - r + s \text{ such that } r = lexminratio(B, s) \text{ then } s = lexminratio(\bar{B}, r)$
  - Lexicographic pivot rule is reversible!
- $pivot(r, s)$ is reversible

Reverse Search:
The Recipe

- Start with dictionary corresponding to the optimal vertex
- Ask yourself 'What pivot would have landed me at this dictionary if i was running simplex?'
- Go to that dictionary by applying reverse pivot to current dictionary
- Ask the same question again
- Generate the so-called 'reverse search tree'

What pivot would have landed me at this dictionary if i was running simplex?

Given a basis B, index $v \in N$, let $a = A_B^{-1} A_v$,
and for any $t = 1, \ldots, m$ let w^t be the vector of
coefficients of row t of the current dictionary.
The function $reverse(B, u, v)$ is true and returns $u = B_i$
if and only if
(i) $\mathrm{w}_v^0 > 0$
(ii) $u = B_i = leximinratio(B, v) \neq 0$, and
(iii) setting $\bar{w} = w^0 - a_0 w^i / a_i$, we have $w_j \geq 0, \forall j \in N, j < u$

Pitfalls

- Initial basis not being the unique optimum basis
- Degeneracy of other vertices: One vertex corresponding to many (actually too many) bases
  - Can result in program outputting a vertex more than once
- Earlier version *rs* suffered from all these

Initial Basis not the unique optimum basis

- Reverse search must begin at the optimal vertex corresponding to any cost.
- In *rs*, one would have to find all possible bases corresponding to initial vertex and perform reverse search on them.
- In *lrs*, we define initial cost in such a way that initial vertex has a unique Basis
- We define cost as: $x_0 = \sum_{j=1}^{d} a_{0,j} x_{m+j}$

$$\text{where } b_0 = 0 \text{ and } a_{0,j} = -1, j = 1, \ldots, d$$

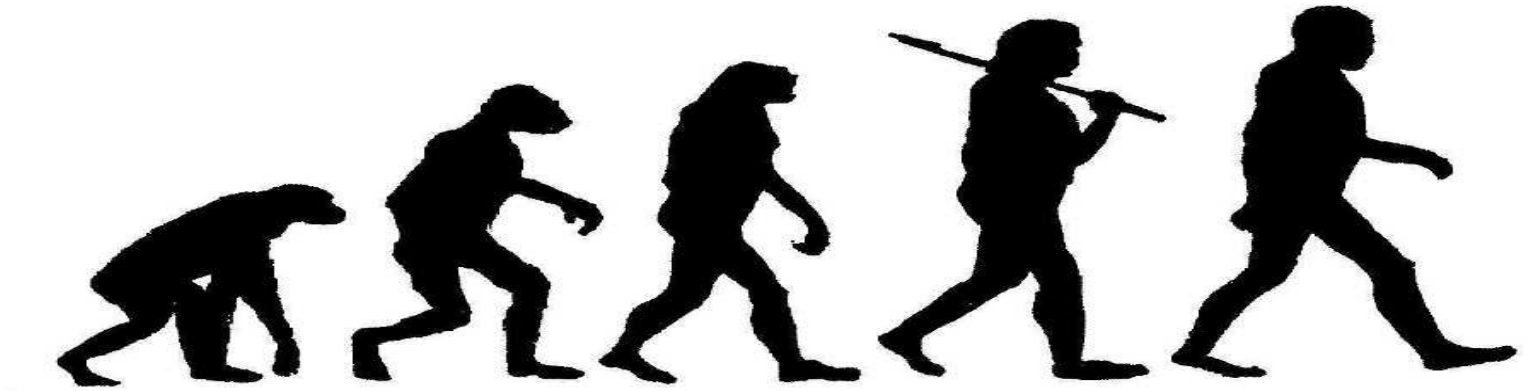
$$x_0 - \sum_{j=1}^{d} a_{0,j} x_{m+j} = 0$$

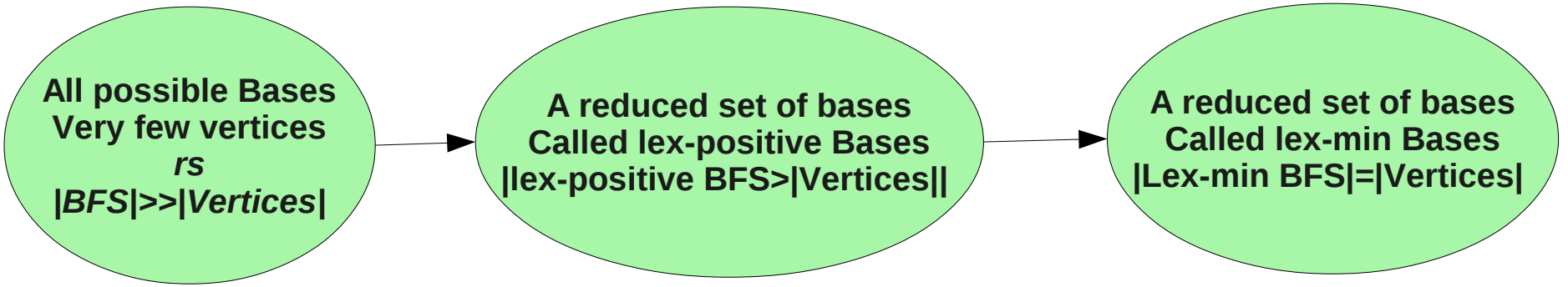
- This cost makes the top row of the dictionary

Degeneracy of vertices

- Define lex-min bases
- B is lex-min for some Basic Feasible Solution x if and only if there does not exist $r \in B$ and $s \in N$ such that,
  (i) $r > s$ (ii) $x_r = 0$ and (iii) $(A_B^{-1} A_s)_r \neq 0$
- Clearly if (i)-(iii) was to be true, we could do $pivot(r, s)$ and get $\bar{B} = B - r + s$ that is lexicographically smaller than B
- $lexmin(B, 0)$ summarizes the operation

Evolution of *lrs*





Recall these functions

$pivot(r, s)$: translate current dictionary into dictionary corresponding to $\bar{B} = B - r + s$ by performing pivot operation

$reverse(B, u, v)$: Given basis B and cobasic index v, returns a basic index u if the said reverse pivot is feasible, otherwise return 0

$lexmin(B, 0)$=True if current basis B is lexmin

selectpivot(B,r,j)= Given a basis B, returns indices $r \in B$ and j such that $N_j = s$ selected by lexicographic pivot rule for simplex algorithm

Pseudo-code

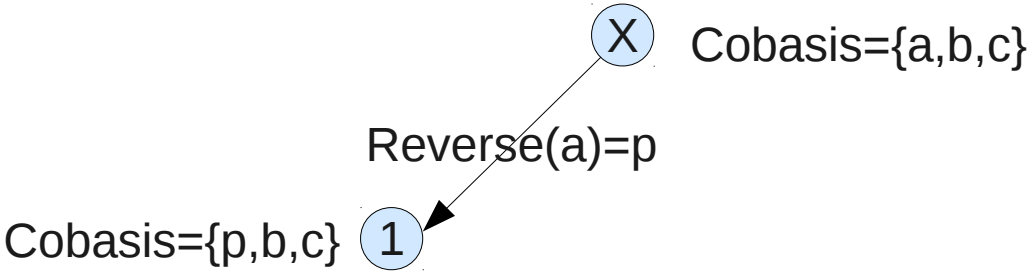
```
B = B*; j = 1;
repeat
    while j ≤ d
        { v = N_j ;
        if reverse(B, u, v)
        then { pivot(B, u, v); // new basis found //
               if lexmin(B, 0) then output current vertex;
               j = 1;
              }
         else j = j + 1;
        }
    selectpivot(B, r, j); //backtrack //
    pivot(B, r, N_j );
    j = j + 1;
until j > d and B = B*
.
```

Pseudo-code

```
while j ≤ d
    { v = N_j ;
    if reverse(B, u, v)
    then { pivot(B, u, v); // new basis found //
           if lexmin(B, 0) then output current vertex;
           j = 1;
          }
     else j = j + 1;
    }
```

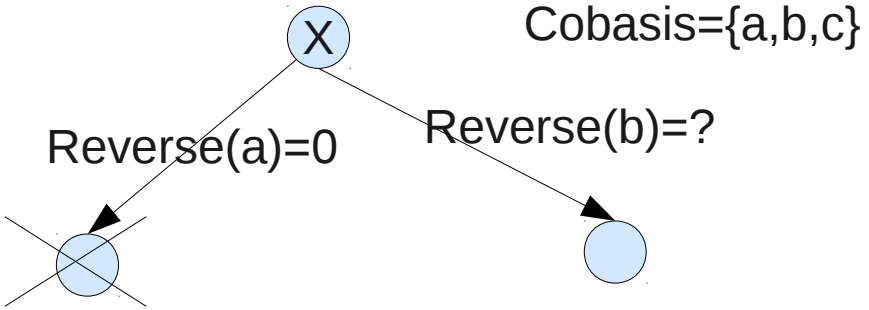
•Examine each cobasic column of dictionary by **reverse** to see if there exists a lex positive pivot using this column

•Reverse will be executed d times for each dictionary



Pseudo-code

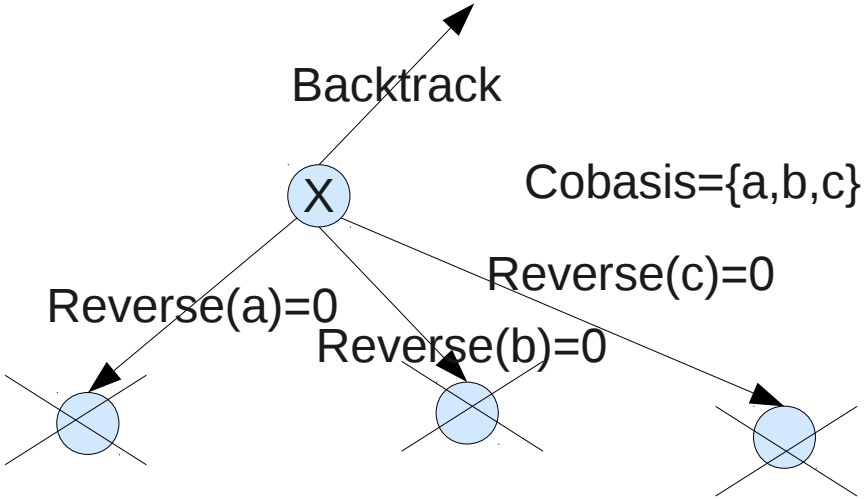
```
while j ≤ d
      { v = N_j ;
      if reverse(B, u, v)
      then { pivot(B, u, v); // new basis found //
             if lexmin(B, 0) then output current vertex;
             j = 1;
             }
       else j = j + 1;
      }
```



Pseudo-code

$$\text{selectpivot(B, r, j); //backtrack //}$$
$$\text{pivot(B, r, } N_j \text{);}$$
$$\text{j} = \text{j} + 1;$$

- When whie loop terminates, return to the parent of current node: backtrack
- Correct j is restored by selectpivot
- Subsequent j=j+1 means next cobasic element of the parent comes under consideration



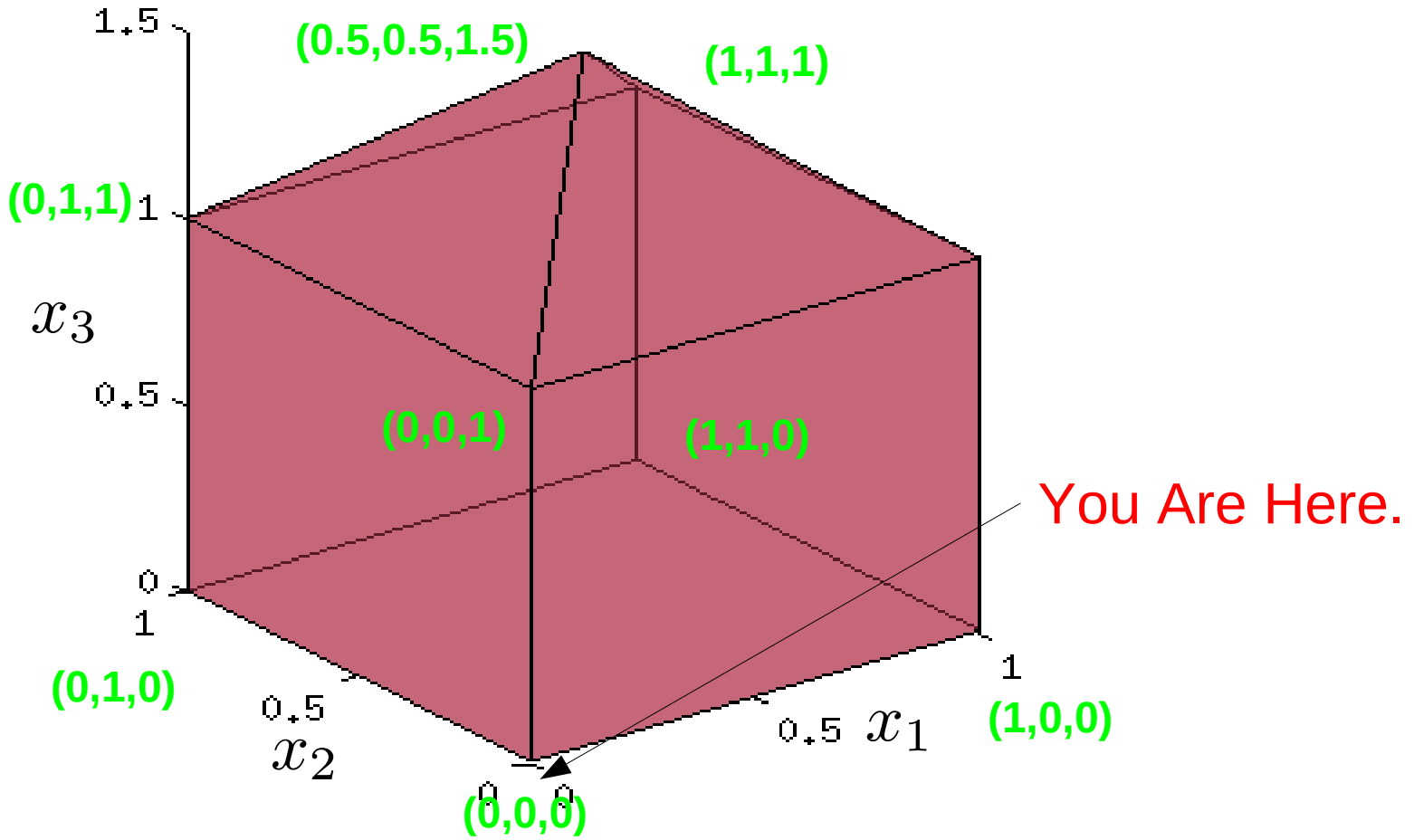
Pseudo-code

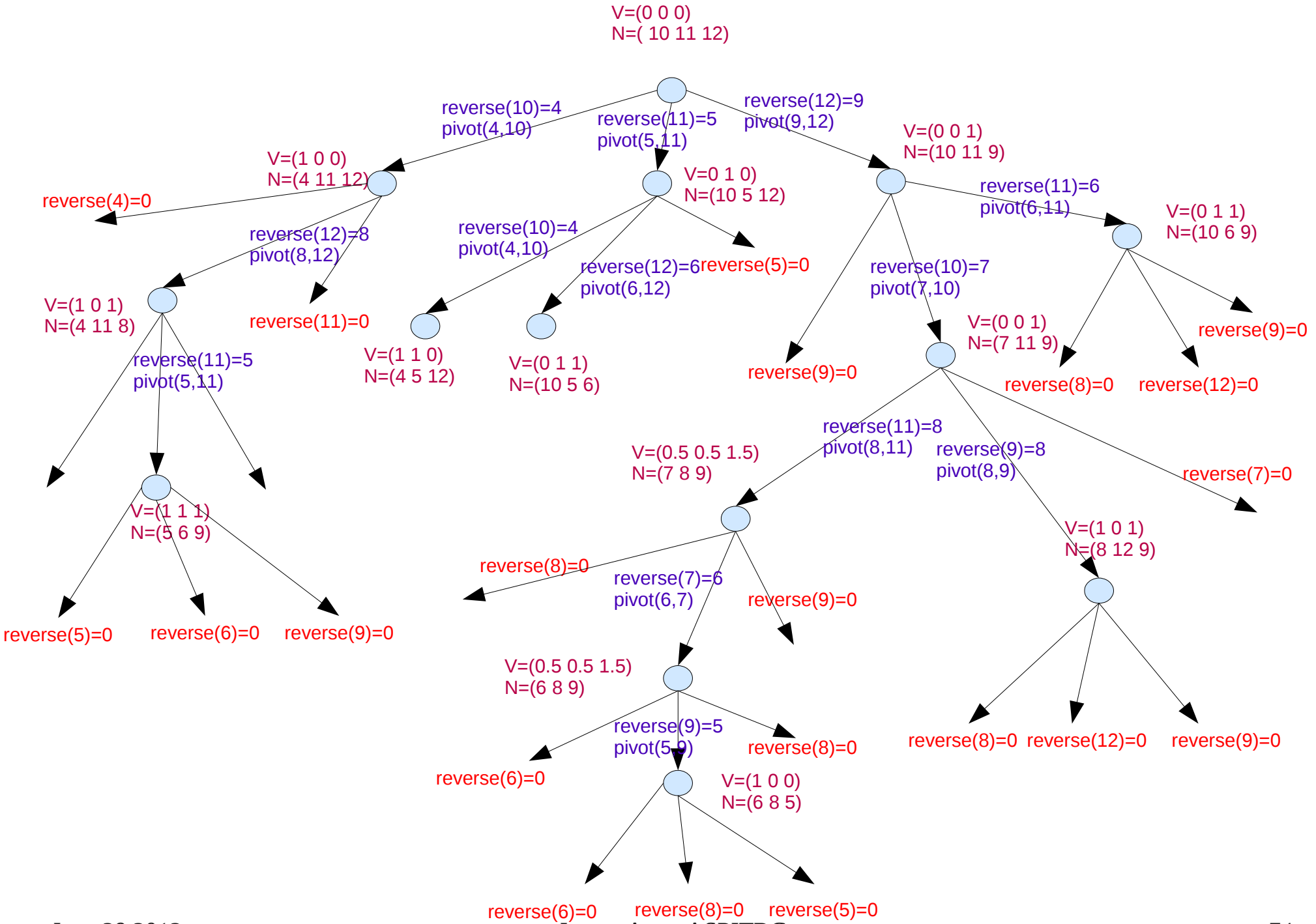
$$\textbf{until } j > d \text{ and } \mathrm{B} = \mathrm{B}^*$$

.

- Execution continues until all the columns of the starting dictionary are examined

Lexicographic Reverse Search on our example




V=(0 0 0)
N=(10 11 12)
reverse(10)=4
pivot(4,10)
reverse(11)=5
pivot(5,11)
reverse(12)=9
pivot(9,12)
V=(1 0 0)
N=(4 11 12)
V=0 1 0)
N=(10 5 12)
V=(0 0 1)
N=(10 11 9)
reverse(4)=0
reverse(12)=8
pivot(8,12)
reverse(10)=4
pivot(4,10)
reverse(12)=6
pivot(6,12)
reverse(5)=0
reverse(11)=6
pivot(6,11)
V=(0 1 1)
N=(10 6 9)
V=(1 0 1)
N=(4 11 8)
reverse(11)=0
V=(1 1 0)
N=(4 5 12)
V=(0 1 1)
N=(10 5 6)
reverse(10)=7
pivot(7,10)
reverse(9)=0
V=(0 0 1)
N=(7 11 9)
reverse(9)=0
reverse(8)=0
reverse(12)=0
reverse(11)=5
pivot(5,11)
V=(1 1 1)
N=(5 6 9)
reverse(5)=0
reverse(6)=0
reverse(9)=0
V=(0.5 0.5 1.5)
N=(7 8 9)
reverse(11)=8
pivot(8,11)
reverse(9)=8
pivot(8,9)
reverse(7)=0
V=(1 0 1)
N=(8 12 9)
reverse(8)=0
reverse(7)=6
pivot(6,7)
reverse(9)=0
V=(0.5 0.5 1.5)
N=(6 8 9)
reverse(9)=5
pivot(5,9)
reverse(8)=0
reverse(6)=0
V=(1 0 0)
N=(6 8 5)
reverse(8)=0
reverse(12)=0
reverse(9)=0
reverse(6)=0
reverse(8)=0
reverse(5)=0

Conclusion

- Lexicographic reverse search is an improved reverse search for vertex enumeration
- It is robust- not affected by degeneracy
- Yet, it travels more bases than there are vertices, hence requiring lot of computational power

Starting dictionary

1	0	0	0	0	0	0	0	0	0	1	1	1	0
0	1	0	0	0	0	0	0	0	0	-1	0	0	0
0	0	1	0	0	0	0	0	0	0	0	-1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	-1	0
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	2
0	0	0	0	0	0	0	1	0	0	0	-1	1	1
0	0	0	0	0	0	0	0	1	0	1	0	1	2
0	0	0	0	0	0	0	0	0	1	-1	0	1	1

Result of pivot(4,10) (real notation)

1	0	0	0	-1	0	0	0	0	0	0	1	1	-1
0	1	0	0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	-1	0	0
0	0	0	1	0	0	0	0	0	0	0	0	-1	0
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	0	0	1	1	2
0	0	0	0	0	0	0	1	0	0	0	-1	1	1
0	0	0	0	-1	0	0	0	1	0	0	0	1	1
0	0	0	0	1	0	0	0	0	1	0	0	1	2

Result of pivot(8,12)

1	0	0	0	0	0	0	0	-1	0	0	1	0	-2
0	1	0	0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	0	0	-1	0	0
0	0	0	1	-1	0	0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	1	0	1	0	-1	0	0	1	0	1
0	0	0	0	1	0	0	1	-1	0	0	-1	0	0
0	0	0	0	-1	0	0	0	1	0	0	0	1	1
0	0	0	0	2	0	0	0	-1	1	0	0	0	1

Result of pivot(5,11)

1	0	0	0	0	-1	0	0	-1	0	0	0	0	-3
0	1	0	0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	-1	0	0	0	1	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	1	-1	1	0	-1	0	0	0	0	0
0	0	0	0	1	1	0	1	-1	0	0	0	0	1
0	0	0	0	-1	0	0	0	1	0	0	0	1	1
0	0	0	0	2	0	0	0	-1	1	0	0	0	1

Now backtrack all the way to root

Result of pivot(5,11)

1	0	0	0	0	-1	0	0	0	0	1	0	1	-1
0	1	0	0	0	0	0	0	0	0	-1	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	-1	0
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	-1	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	1	0	0	0	0	1	2
0	0	0	0	0	0	0	0	1	0	1	0	1	2
0	0	0	0	0	0	0	0	0	1	-1	0	1	1

Result of pivot(4,10)

1	0	0	0	-1	-1	0	0	0	0	0	0	1	-2
0	1	0	0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	-1	0
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	-1	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	1	0	0	0	0	1	2
0	0	0	0	-1	0	0	0	1	0	0	0	1	1
0	0	0	0	1	0	0	0	0	1	0	0	1	2

Backtrack 1 step: pivot(10,4)

1	0	0	0	0	-1	0	0	0	0	1	0	1	-1
0	1	0	0	0	0	0	0	0	0	-1	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	0	0	0	0	0	0	0	-1	0
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	-1	1	0	0	0	0	0	1	1
0	0	0	0	0	1	0	1	0	0	0	0	1	2
0	0	0	0	0	0	0	0	1	0	1	0	1	2
0	0	0	0	0	0	0	0	0	1	-1	0	1	1

Result of pivot(6,12)

1	0	0	0	0	0	-1	0	0	0	1	0	0	-2
0	1	0	0	0	0	0	0	0	0	-1	0	0	0
0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	-1	1	0	0	0	0	0	0	1
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	-1	1	0	0	0	0	0	1	1
0	0	0	0	0	2	-1	1	0	0	0	0	0	1
0	0	0	0	0	1	-1	0	1	0	1	0	0	1
0	0	0	0	0	1	-1	0	0	1	-1	0	0	0

Backtrack all the way to root

Result of pivot(9,12)

1	0	0	0	0	0	0	0	0	-1	2	1	0	-1
0	1	0	0	0	0	0	0	0	0	-1	0	0	0
0	0	1	0	0	0	0	0	0	0	0	-1	0	0
0	0	0	1	0	0	0	0	0	1	-1	0	0	1
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	-1	1	1	0	1
0	0	0	0	0	0	0	1	0	-1	1	-1	0	0
0	0	0	0	0	0	0	0	1	-1	2	0	0	1
0	0	0	0	0	0	0	0	0	1	-1	0	1	1

Result of pivot (7,10)

1	0	0	0	0	0	0	-2	0	1	0	3	0	-1
0	1	0	0	0	0	0	1	0	-1	0	-1	0	0
0	0	1	0	0	0	0	0	0	0	0	-1	0	0
0	0	0	1	0	0	0	1	0	0	0	-1	0	1
0	0	0	0	1	0	0	-1	0	1	0	1	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	-1	0	0	0	2	0	1
0	0	0	0	0	0	0	1	0	-1	1	-1	0	0
0	0	0	0	0	0	0	-2	1	1	0	2	0	1
0	0	0	0	0	0	0	1	0	0	0	-1	1	1

Result of pivot(8,11)

1	0	0	0	0	0	0	1	-1.5	-0.5	0	0	0	-2.5
0	1	0	0	0	0	0	0	0.5	-0.5	0	0	0	0.5
0	0	1	0	0	0	0	-1	0.5	0.5	0	0	0	0.5
0	0	0	1	0	0	0	0	0.5	0.5	0	0	0	1.5
0	0	0	0	1	0	0	0	-0.5	0.5	0	0	0	0.5
0	0	0	0	0	1	0	1	-0.5	-0.5	0	0	0	0.5
0	0	0	0	0	0	1	1	-1	-1	0	0	0	0
0	0	0	0	0	0	0	0	0.5	-0.5	1	0	0	0.5
0	0	0	0	0	0	0	-1	0.5	0.5	0	1	0	0.5
0	0	0	0	0	0	0	0	0.5	0.5	0	0	1	1.5

Result of pivot(6,7)

1	0	0	0	0	0	-1	0	-0.5	0.5	0	0	0	-2.5
0	1	0	0	0	0	0	0	0.5	-0.5	0	0	0	0.5
0	0	1	0	0	0	1	0	-0.5	-0.5	0	0	0	0.5
0	0	0	1	0	0	0	0	0.5	0.5	0	0	0	1.5
0	0	0	0	1	0	0	0	-0.5	0.5	0	0	0	0.5
0	0	0	0	0	1	-1	0	0.5	0.5	0	0	0	0.5
0	0	0	0	0	0	1	1	-1	-1	0	0	0	0
0	0	0	0	0	0	0	0	0.5	-0.5	1	0	0	0.5
0	0	0	0	0	0	1	0	-0.5	-0.5	0	1	0	0.5
0	0	0	0	0	0	0	0	0.5	0.5	0	0	1	1.5

Result of pivot(6,10)

1	0	0	0	0	-1	0	0	-1	0	0	0	0	-3
0	1	0	0	0	1	-1	0	1	0	0	0	0	1
0	0	1	0	0	1	0	0	0	0	0	0	0	1
0	0	0	1	0	-1	1	0	0	0	0	0	0	1
0	0	0	0	1	-1	1	0	-1	0	0	0	0	0
0	0	0	0	0	2	-2	0	1	1	0	0	0	1
0	0	0	0	0	2	-1	1	0	0	0	0	0	1
0	0	0	0	0	1	-1	0	1	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	-1	1	0	0	0	0	0	1	1

Backtrack: pivot(9,5),pivot(7,6),pivot(11,8)

1	0	0	0	0	0	0	-2	0	1	0	3	0	-1
0	1	0	0	0	0	0	1	0	-1	0	-1	0	0
0	0	1	0	0	0	0	0	0	0	0	-1	0	0
0	0	0	1	0	0	0	1	0	0	0	-1	0	1
0	0	0	0	1	0	0	-1	0	1	0	1	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	-1	0	0	0	2	0	1
0	0	0	0	0	0	0	1	0	-1	1	-1	0	0
0	0	0	0	0	0	0	-2	1	1	0	2	0	1
0	0	0	0	0	0	0	1	0	0	0	-1	1	1

Result of pivot(8,9)

1	0	0	0	0	0	0	0	-1	0	0	1	0	-2
0	1	0	0	0	0	0	-1	1	0	0	1	0	1
0	0	1	0	0	0	0	0	0	0	0	-1	0	0
0	0	0	1	0	0	0	1	0	0	0	-1	0	1
0	0	0	0	1	0	0	1	-1	0	0	-1	0	0
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	-1	0	0	0	2	0	1
0	0	0	0	0	0	0	-1	1	0	1	1	0	1
0	0	0	0	0	0	0	-2	1	1	0	2	0	1
0	0	0	0	0	0	0	1	0	0	0	-1	1	1

Backtrack pivot(9,8),pivot(10,7)

1	0	0	0	0	0	0	0	0	-1	2	1	0	-1
0	1	0	0	0	0	0	0	0	0	-1	0	0	0
0	0	1	0	0	0	0	0	0	0	0	-1	0	0
0	0	0	1	0	0	0	0	0	1	-1	0	0	1
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	1	0	0	-1	1	1	0	1
0	0	0	0	0	0	0	1	0	-1	1	-1	0	0
0	0	0	0	0	0	0	0	1	-1	2	0	0	1
0	0	0	0	0	0	0	0	0	1	-1	0	1	1

Result of pivot(6,11)

1	0	0	0	0	0	-1	0	0	0	1	0	0	-2
0	1	0	0	0	0	0	0	0	0	-1	0	0	0
0	0	1	0	0	0	1	0	0	-1	1	0	0	1
0	0	0	1	0	0	0	0	0	1	-1	0	0	1
0	0	0	0	1	0	0	0	0	0	1	0	0	1
0	0	0	0	0	1	-1	0	0	1	-1	0	0	0
0	0	0	0	0	0	1	0	0	-1	1	1	0	1
0	0	0	0	0	0	1	1	0	-2	2	0	0	1
0	0	0	0	0	0	0	0	1	-1	2	0	0	1
0	0	0	0	0	0	0	0	0	1	-1	0	1	1

Backtrack to root

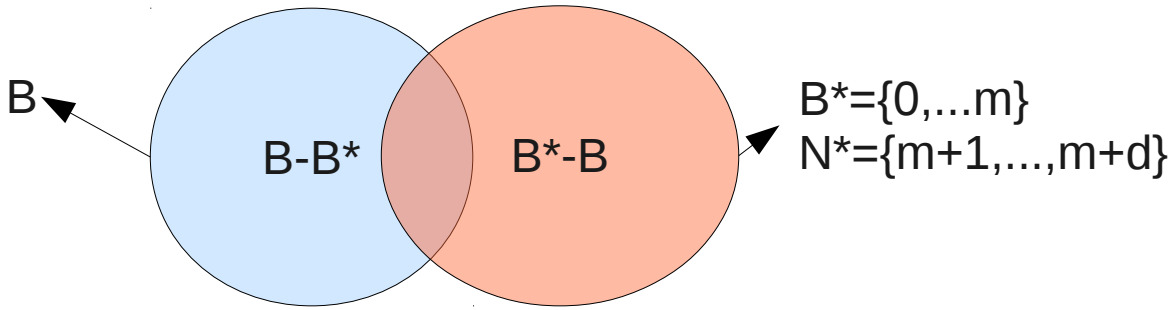
Proof of uniqueness of initial basis

- Suppose there exists another lex-positive optimum basis B
- Recall D matrix.

$$\mathrm{D} = \begin{bmatrix} A_B^{-1}b & A_B^{-1} \end{bmatrix} = \begin{bmatrix} \beta_0 & \alpha_{0,1} & \dots & \alpha_{0,m} \\ \vdots & & & \vdots \\ \vdots & & & \vdots \\ \beta_m & \alpha_{m,1} & \dots & \alpha_{m,m} \end{bmatrix}$$

$$\mathrm{D}^* = \begin{bmatrix} A_B^{-1}b & A_B^{-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \dots & 0 & 1 & \dots & 1 \\ \beta_1 & \alpha_{1,1} & \dots & \dots & \alpha_{m-d+1,1} & \dots & \alpha_{m,1} \\ \vdots & \vdots & & & & & \vdots \\ \beta_m & \alpha_{m,1} & \dots & \dots & \alpha_{m-d+1,m} & \dots & \alpha_{m,m} \end{bmatrix}$$

Proof of uniqueness of initial basis



- Consider pivoting from B* to B. Indices B-B*= $k_1, \ldots, k_q$ must be brought into basis
- Every pivot will involve subtracting a certain row vector from cost row
- Since B is also optimum $\beta_{k_j} = 0$
- It follows that $(\alpha_{0,1}, \ldots, 0, m)$ is lexicographicallly negative

$$\mathrm{D}^* = \begin{bmatrix} A_B^{-1}b & A_B^{-1} \end{bmatrix} = \begin{bmatrix} 0 & 0 & \ldots & 0 & 1 & \ldots & 1 \\ \beta_1 & \alpha_{1,1} & \ldots & \ldots & \alpha_{m-d+1,1} & \ldots & \alpha_{m,1} \\ \vdots & \vdots & & & & & \vdots \\ \beta_m & \alpha_{m,1} & \ldots & \ldots & \alpha_{m-d+1,m} & \ldots & \alpha_{m,m} \end{bmatrix}$$