15.053 Exam 1 Notes

- Linear program terminology
  - **Decision variables** – the variables we need to determine
  - **Input parameters** – the costs, times taken for things, etc…
  - **Objective function** – the thing that needs to be maximised or minimized.
  - **Unbounded** – feasible choices of the decision variables can produce arbitrarily good objective function values.
  - **Linear program** – constraints and objective function are linear (ie: weighed sums of the decision variables).
- Solving problems graphically
  - **Optimal solution** – optimal line minimised only at "corner point".
  - **Plotting a line** of the form $ax + by = C$
    - Get the x by itself and set y to 0. Vice versa.
    - For the objective function, let $C =$ some random value.
- Step sizes and directions
  - As long as the step size is finite, there's no indication that the program is unbounded.
  - To find the conditions required for a feasible direction Δw at a given point…
    - Find the active (tight) constraints at that point
    - Require that $\Delta w \cdot \left(\text{constraint vector}\right) \geq = \leq 0$ as needed.
  - To check whether a direction is improving, dot it with the objective function vector, and look at the sign of the result.
  - To find the maximum step size, λ, assuming you make the change $\lambda \Delta w$, change all the variables accordingly, solve, and take the smallest one.
- Improving search
  - The feasible region of an LP is **convex** [the line segment between every pair of feasible points falls entirely within the feasible region].

  - This means that **every local optimal solution is a global optimal solution**.
  - Algorithm
    - **Initialisation** – choosing stating solution
    - **Check for local optimality** (no improving solution)
    - **Find improving and feasible direction**
    - **Find step size**
    - **Advance**
- Standard-form LP
  - All variables must be non-negative, and only equalities are included.
  - Add slack variables to make up for it.
  - Objective function is $\boldsymbol{c} \cdot \boldsymbol{x}$ and constraints are $\boldsymbol{Ax} = \boldsymbol{b}, \boldsymbol{x} \geq 0$.
  - For a negative variable, make a new variable $\overline{x} = -x$. For a variable that can be positive or negative, make two new variables $x = x^+ + x^-$.
  - For an absolute value $|x|$, replace it with a new variable z and add constraints $z \geq x$ and $z \geq -x$, with z_1, z_2, $z_3 \geq 0$.
  - For a **maximin** (maximise the minimum), introduce a new variable f, and maximise f subject to $f \leq$ [said variable].
  - n is the number of decision variables, m is the number of constraints, c_j is the objective function coefficient of x_j and $a_{i,j}$ is the coefficient of x_j in the i^{th} constraints and b_j is the RHS of main constraint i.
- Types of points
  - **Interior point** – no inequality is active
  - **Boundary point** – at least one inequality constraint is satisfied as equality at a given point
  - **Extreme points** of convex sets – those that do not lie within the line segment between any two other points in the set. Generally a solution of a system of n equations and n variables. Some can be determined by different sets of active constraints.

  - **Adjacent extreme points** – determined by active constraints differing in only 1 element.
- Simplex – intro
  - Effectively an improving search algorithm
  - Starts at an extreme point, and moves to adjacent extreme point with better objective value…
  - …until no adjacent extreme point has better objective value.
  - Algorithm
    - **Initialization** – choose starting feasible solution
    - If **no improving feasible direction**, stop
    - Construct **improving feasible direction**
    - Choose **step size** [if no limit, stop]
    - **Advance**
- Simplex – standard display

	x_1	x_n	
Max **c**	c_1	c_n	**b**
	a_{11}		b_1
A			
		a_{nn}	b_n
Basic var?			
$\boldsymbol{x}^{(0)}$			$\boldsymbol{c} \cdot \boldsymbol{x}^{(0)}$
$\Delta \boldsymbol{x}$ for $\boldsymbol{x}_?$			$\bar{c}_?$
	…ratios…		
New bas. var?			
$\boldsymbol{x}^{(1)}$			$\boldsymbol{c} \cdot \boldsymbol{x}^{(1)}$

- Simplex – basic solutions
  - Fix $n - m$ variables (**nonbasic** variables) to 0. Obtain a unique solution for the remaining system of m variables (**basic** variables) and m equations.
  - Qualifications

    - A basic solution is **feasible** (BFS) if it satisfies all non-negativity constraints.
    - Sometimes, we can't even get a basic solution, if the system of equations obtained after setting nonbasic variables to 0 doesn't have a unique solution.
  - For a standard-form LP, the BFSs are exactly the extreme points of the feasible region. We need to cycle through them.
- Simplex – first phase
  - This phase finds a basic solution, by creating an artificial linear program.
  - Procedure
    - Multiply constraints by –1 as necessary to make **b** positive.
    - Add a non-negative **artificial variable** for each constraint, and set to objective function to *minimise* the *sum* of these artificial variables.
    - This has an easy to find BFS – set all the non-artifical variables to 0.
  - This is good because
    - It can't be infeasible (because it has a BFS)
    - It can't be unbounded (because the variables are ≥ 0)
  - Solve.
  - If the solution of A doesn't make all artificial variables 0, then the original program is infeasible.
  - If the solution of A has all the artificial variables 0, then what's left over is a BFS for the original program.
- Simplex – finding an improving direction
  - In improving a direction, we choose one nonbasic variable, and move it out of the basis. To decide which, we see which improves our cost best.
  - Procedure
    - Choose a nonbasic variable x_j
    - Move direction

$$\Delta x_i = \begin{cases} +1 & i = j \\ 0 & i \neq j \quad (x_i \text{ nonbasic}) \\ ? & \text{otherwise} \end{cases}$$

- Need

$$\boldsymbol{A}\left(\boldsymbol{x} + \lambda \Delta \boldsymbol{x}\right) = \boldsymbol{b}$$
$$\boxed{\boldsymbol{A}\Delta \boldsymbol{x} = \boldsymbol{0}}$$

m variable, m equations, has unique solution because $\boldsymbol{x}$ is a BFS.

- The change in our objective function is

$$\boldsymbol{c} \cdot \left(\boldsymbol{x} + \lambda \Delta \boldsymbol{x}\right) - \boldsymbol{c} \cdot \boldsymbol{x} = \lambda \boldsymbol{c} \cdot \Delta \boldsymbol{x}$$

And so we calculate the **reduced cost** for the variable j

$$\overline{c}_j = \boldsymbol{c} \cdot \Delta \boldsymbol{x}$$

- We calculate those for **each nonbasic variable**, and then choose the one with the **best reduced cost** to move into the basis.
- If there is no improving direction – stop. We're done

- Simplex – step size
  - This is an LP, so the improving directions are improving forever, and we constructed the system such that equality constraints are satisfied, so problems must come from violating non-negativity. We want to increase λ until we violate one of those.
  - Increasing λ can only lead to bad things for $\Delta x_j < 0$.
  - We therefore use step size

$$\lambda = \min\left\{\frac{x_j^{(t)}}{-\Delta x_j} : \Delta x_j < 0\right\}$$

Calculate this ratio for every variable in the basis for our chosen direction, add to standard display, and pick the minimum one.

- Simplex – updating the basis
  - $\boldsymbol{x}^{(t+1)} \leftarrow \boldsymbol{x}^{(t)} + \lambda \Delta \boldsymbol{x}$
  - Nonbasic variable used to generate direction becomes basic.
  - Basic variable that determines step size becomes nonbasic.
- Simplex – degeneracy

  - Happens if more than the required number of constraints are active at a given extreme point.
  - In other words, one of the basic variables is 0.
  - The simplex method may generate a step size of 0 (if the simplex direction involves decreasing a variable that is already equal to 0) and can then "get stuck" for a few steps.
  - Computations will (usually) escape these zero-length moves and eventually produce a direction where improving progress can be made.
  - Thus, the simplex method does *not* necessary move to an **adjacent extreme point**, but it does move to an **adjacent basis**.