**DISCLAIMER:** These notes are not necessarily an accurate representation of what I said during the class. They are mostly what I intend to say, and have not been carefully edited.

Main topics:
- Linear inequality and linear programs.
- Shortest path as a linear program.

Announcements:
- Office hours moved around this and next week, check Piazza and course website (which has a schedule).
- Test next Wednesday, Oct 18, 3:05pm - 4:15pm, alternate location Howey Physics L4, try not to all go there.
- Monday Oct 16 will be a review class.
- Review session Friday Oct 13 in Klaus 2447.

## 1 Linear Programs

The main idea of linear programming is to formula a problem as a set of linear inequalities, and invoke a high powered algorithm (whose details we won’t go into) to solve.

There are two main primitives in linear programs:

1. variables $x_1 \ldots x_n$.
2. linear inequalities of the form

$$\sum_j a_{ij} x_j \leq b_j.$$ 

The goal is to minimize/maximize some objective subject to the constraints. The objective is also linear,

$$\sum_j c_j x_j.$$
Here the $i$ in $a_{ij}$ indexes into constraints, whose number we will denote with $m$. A linear program in standard form is formally given $a_{ij}$ for $1 \leq i \leq m, 1 \leq j \leq n$, and costs $c_j$ for $1 \leq j \leq n$, and solve for $x_1 \ldots x_n$ that:

$$\text{maximize} \quad \sum_{j} c_j x_j$$

subject to

$$\sum_{j} a_{ij} x_j \leq b_i \quad \forall 1 \leq i \leq m$$

There are some very good packages for optimizing small to medium sized integer programs, such as [http://cvxr.com/cvx/](http://cvxr.com/cvx/).

## 2 Maximum Independent Set as a LP, and Fractional Solutions

Consider the maximum independent set problem, where we want to find the maximum number of vertices that aren’t pairwise adjacent. Here one way to set up a linear program is to pick $x_u$ as indicator for whether $u$ is picked. Then the first constraint is

$$0 \leq x_u \leq 1,$$

where the objective is to maximize $\sum_{u} x_u$. The fact that any edge $uv$ has at most one end point chosen can be encoded as

$$x_u + x_v \leq 1$$

for all edges $uv \in E$. So combining this together gives the linear program:

- **Variables**: $x_u$ for each vertex $u$.
- **Objective**: maximize $\sum_{u \in V} x_u$
- **Constraints**:
  - For each vertex $u$, $x_u \geq 0$, and $x_u \leq 1$.
  - For each edge $uv$, $x_u + x_v \leq 1$. 

2
An important issue in linear programming is that the solutions are fractional. For some programs such as shortest paths and maximum flow, we can show that any fractional solution can be converted to an integral one. However, this is not always the case, and the maximum independent set problem is a counter example.

Consider the triangle on three vertices, $a$, $b$, and $c$. The linear program is:

\[
\begin{align*}
\text{maximize} & \quad x_a + x_b + x_c \\
\text{subject to:} & \quad x_a + x_b \leq 1 \\
& \quad x_b + x_c \leq 1 \\
& \quad x_a + x_c \leq 1 \\
& \quad x_a \leq 1 \\
& \quad x_b \leq 1 \\
& \quad x_c \leq 1 \\
& \quad x_a \geq 0 \\
& \quad x_b \geq 0 \\
& \quad x_c \geq 0 
\end{align*}
\]

We can only pick one of the vertices, but the solution of $x_u = 1/2$ is a valid fractional solution with objective $3/2$. Unlike solving fractional linear programs, solving integer linear programs, or integer programs, is much harder. We will see next month that a polynomial time algorithm for solving integer programs has a wide range of consequences.

3 Shortest Path as a Linear Program

Consider the shortest path problem again. For each arc $u \to v$, we can have a variable $x_{u \to v}$

denoting whether the arc from $u$ to $v$ is used or not. If we restrict $x$ to be $0/1$, this can be interpreted as whether $u \to v$ is on the path or not. Otherwise, we can view it as the amount of goods sent from $u$ to $v$, while the overall goal is to send 1 unit of good from $s$ to $t$, while minimizing the total cost:

\[
\sum_{u \to v} c_{u \to v} x_{u \to v}.
\]

Then for the constraints, we have:

\[
x_{u \to v} \geq 0
\]

as the edges are directed. Furthermore, we need to have nothing accumulating in intermediate vertices. The amount of goods entering a vertex $u$ is

\[
\sum_{u \to w} x_{w \to u}.
\]
while the amount leaving is
\[ \sum_{u \to v} x_{u \to v}. \]

If we want a path from \( s \) to \( t \), then it should have 1 unit leaving \( s \), 1 unit entering \( t \), and have entering/leaving cancel out everywhere else. This gives the constraint
\[
- \sum_{w \to u} x_{w \to u} + \sum_{u \to v} x_{u \to v} = \begin{cases} 
1 & \text{if } u = s \\
-1 & \text{if } u = t \\
0 & \text{otherwise}
\end{cases}
\]

Then we also need \( x_{u \to v} \geq 0 \) (can’t take an arc backwards), as well as the objective
\[
\sum_e c_e x_e
\]
where \( c_e \) is the cost of taking edge \( e \).

As it turns out, this program suffices for solving the shortest path problem.