• 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:
  – Maximum matching and formulation as maximum flow.
  – Vertex cover = max matching.

Let’s start with the following puzzle: given a subset of the chessboard, place the maximum of knights s.t. no two pieces can attack each other. (A knight can attack another piece if the horizontal/vertical difference in coordinates form the set \{1, 2\}.)

There is a similar version of this for rooks (which attack anything horizontally/vertically), but that requires obstacles to become interesting.

This problem is a special case of independent set: given a graph with some edges, find the maximum number of vertices such that no two adjacent vertices are picked. Here we can turn each valid grid on the chess board into a vertex, and put an edge between any two vertices within a knight’s move. Then placing the maximum number of knights becomes finding the maximum independent set.

Finding max independent set is difficult in general. However, in this case the graph is also bipartite: its vertices can be partitioned into \(A\) and \(B\) such that all edges are between some \(a \in A\) and \(b \in B\). The primary goal of this class is to show:

\[
\left| \text{max matching} \right| = |A| + |B| - \left| \text{max independent set} \right|.
\]

We first reduce finding maximum bipartite matching to computing maximum flow. We add a supersource \(s\), a supersink \(t\), and add the edges

1. \(s \rightarrow a\) with capacity 1.
2. \(b \rightarrow t\) with capacity 1.
3. \(a \rightarrow b\) for each edge \(ab\) with capacity \(\infty\).

The goal is to have each matched edge \(ab\) correspond to a path

\[
s \rightarrow a \rightarrow b \rightarrow t.
\]
The first two constraints mean that each \( a \) and \( b \) can be used at most once. The third constraint gives the allowed edges: its capacities can also work with 1, but this version is easier for the mincut conversion.

We can check that a matching corresponds to a flow of the same value, and an integer flow can also be converted to a matching. So the max flow value in this network equals to the size of the max matching.

Now consider an \( s \to t \) cut of finite value in this graph, let the \( S \) side of it be:

\[
\{s\} \cup S_A \cup S_B.
\]

Similarly let

\[
T_A = A \setminus S_A,
\]
\[
T_B = B \setminus S_B
\]

The fact that there are no infinite capacity edges in this cut means that there are no edges from \( S_A \) to \( T_B \). In other words,

\[
S_A \cup T_B
\]

is an independent set.

On the other hand, the cut also contains all edges from \( s \) to \( T_A \), and all edges from \( S_B \) to \( t \). Since each of these edges have capacity 1, the total value of the cut is

\[
|T_A| + |T_B| = |A| + |B| - |S_A \cup T_B|.
\]

So we established that we can find an independent set of size at least the size of the maximum matching.

\[
|A| + |B| - |\text{max independent set}|.
\]

We can also take this construction in the reverse direction to show that any independent set corresponds to a \( s \to t \) cut in this graph. However, there is a much more direct approach: note that for an edge \( e \), its endpoints cannot both be in the independent set. For a set of \( M \) edges in a matching, each of those edges certify that at most one of its endpoints can be chosen. This means that the max size of an independent set is

\[
\leq |A| + |B| - |M|,
\]

which gives the upper bound.