Matchings.

These are far from perfect. Do point out any errors on EdStem or by emailing the professor.

1 Maximum Weighted Matchings

Given a weighted bipartite graph $G = (U, V, E)$ with weights $w : E \to \mathbb{R}$ the problem is to find the maximum weight matching in G. A matching assigns every vertex in U to at most one neighbor in V; equivalently it is a subgraph of G with induced degree at most 1.

In the case that all edge weights are equal to 1, the problem is simply to find a matching with as many edges as possible (a "maximum-cardinality matching"). This is a standard problem in undergraduate algorithms.

1.1 The path: "backsolving".

The path is a central object in graphs and graph algorithms (and linear algebra.)

Here is a simple problem: Let $x_1 = 1$, and $x_{i+1} = x_i \forall i \in [1, n-1]$.

What's the solution? Clearly, $x_i = 1$ for $x_i \in [1, n]$.

You did it quickly due to being able to do induction. That is, you went to elementary school and no that in the natural numbers there is a next number and this goes on forever.

But an algorithm might be to repeatedly assign, $x_{i+1} = x_i$ starting from $i = 1$ and going until $i = n - 1$. This is Gaussian elimation, or specically backsolving as these equations are in lower triangular form. Which means that the values of x_i is only determined by the values of x_j where $j \leq i$.

Now consider the slightly more complicated equations:

$$x_1 = 1 \text{ and } x_i + x_{i-1} = 1 \tag{1}$$

Again, these are "lower triangular", i.e., $x_{i+1} = 1 - x_i$.

Now consider the problem of finding a matching on a path graph, i.e, a graph that consists of a sequence of vertices with edges between successive vertices. The (perfect) matching problem is assigning a variable with x_e to each edge, and having equalities for vertex v, $x_{(u,v)} + x_{(v,w)}$, for the edges (u, v) and (v, w) that are incident to v. Or if x_i corresponds to the ith edge of the path, the same equations as 1.

Moreover, one can see the algorithm of backsolving would "alternate": that is assign $x_1 = 1$, and $x_2 = 0$ and $x_3 = 1$ and so on.

This is all something to keep in mind in the following.

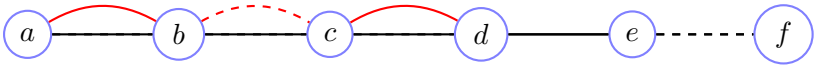

Figure 1: An augmenting path is illustrated in red, where the previous matching is marked by dashed and non-dashed lines.

1.2 Maximum-Cardinality Matching

An algorithm for finding a maximum unweighted matching is to start with any matching (e.g. empty) and repeatedly improve it by finding *augmenting paths*.

An *augmenting path* is a path whose endpoints are unmatched vertices and which alternates between unmatched and matched edges. Changing every edge on an augmenting path – that is, removing every edge on the path that was matched, and instead inserting the edges that were unmatched – must result in a valid matching with exactly one more edge. A more subtle fact is that, conversely, if there are no augmenting paths, then the matching has the maximum possible cardinality.

Augmenting paths can be found by a breadth-first (or depth-first) search starting at unmatched vertices in U alternating between unmatched and matched edges. If the search reaches an unmatched vertex in V we have found an augmenting path and can extend the matching.

See figure 1.

1.3 Maximum Weighted Matchings

Now, we'll allow the edges to have weights $w(e) \in \mathbb{R}$. The goal is to maximize the total weight of edges in our matching. By adding edges with weight 0 we can assume without loss of generality that G is a complete bipartite graph, and by adding dummy nodes we can similarly assume that $|U| = |V|$.

The maximum weight matching problem is solved using the primal dual framework. It is useful to think in terms of upper bounds on the weight of a matching. The sum of the weights of the maximum weight edges incident on U is clearly an upper bound on the weight of a matching. The bound is not tight: in the path of length 3 where the middle edge has weight 2 and the others have weight 1, the upper bound is 3 while the maximum weight of a matching is 2 (Figure 2).

The upper bound is generalized by assigning prices $p(u)$ to the nodes, such that the sum of the prices of the endpoints exceeds the weight of an edge, that is for all edges (u, v) we have $p(u) + p(v) \geq w(u, v)$. The sum of the prices over all the vertices is an upper bound on the weight of a matching,

$$\sum_{(u,v)\in M} w(u,v) \leq \sum_{(u,v)\in M} p(u) + p(v) \leq \sum_{w\in G} p(w)$$

This form of upper bound can be made tight for the length 3 path from Figure 2, by assigning 1 to the middle two nodes in the path.

The best upper bound on the weight of a matching using this approach corresponds to the smallest weighted cover for G, that is the smallest sum over prices such that all edges

Figure 2: Example of non-optimal and optimal price functions. Black edges have weight 1, and the blue edge has weight 2.

are covered; $p(u) + p(v) \geq w(u, v)$. The weight of any cover is greater than the weight of a matching, and this fact will be crucial for designing the algorithm.

The primal dual algorithm maintains a cover and iteratively reduces the weight of the cover by finding maximum matchings on tight edges,

1. The initial cover is chosen to be $p(u) = \max_{v:(u,v)\in E} w(u, v)$ for $u \in U$, and $p(v) = 0$ for $v \in V$. This is a cover as $p(u) + p(v) \geq w(u, v)$ is true for all edges.

2. The edges (u, v) for which $p(u) + p(v) = w(u, v)$ are called the tight edges. Compute a maximum-cardinality matching M on the tight edges. If the matching M has size $|U|$, the problem is solved as we found a matching and cover having the same weight.

3. Perform a breadth first search starting from unmatched vertices in U alternating between unmatched and matched edges. Let U' and V' be the vertices in U and V that are reachable in this manner.

   There cannot be a tight edge (u, v) between U' and $V \setminus V'$. A tight edge not in M would cause v to be in V'. A tight edge not in M would mean that u must have been reached through v, so v must also be reachable.

   Let δ be the minimum excess over edges between U' and $V \setminus V'$:

$$\delta = \min_{\substack{u\in U' \\ v\in V\setminus V'}} p(u) + p(v) - w(u, v).$$

4. Update the prices by setting $p(u) \to p(u) - \delta$ for $u \in U'$ and $p(v) \to p(v) + \delta$ for $v \in V'$. The new prices form a cover due to the choice of δ and the edges in the maximum matching continue to be tight.

   The total price has decreased due to this step as $|U'| > |V'|$, this follows as V' consists of matched vertices and the neighbors of these vertices belong to U'. At least one edge between U' and $V \setminus V'$ becomes tight for every update. Repeat step 2 with the new set of tight edges.

Analysis: Progress is made as the size of the maximum matching over tight edges is monotonically increasing. With some data structures, we can ensure that the size of the matching increases in $O(m \log n)$ time. Thus, in time $O(nm \log n)$ we have a perfect matching and a corresponding feasible price function of the same value. That is, we have a maximum weight matching.

1.4 Maximum Matching: linear programs and duality.

A linear program corresponds to a set of linear inequalities, and a linear function. One wishes to find a point that satisfies the set of linear inequalites and (say) minimizing the objective function. One formulation is specifed by a matrix, A and vectors b and c and corresponds to the following problem of finding a vector x where:

$$Ax \leq b, \max cx, x \geq 0.$$

With this linear program, the "dual" linear program is that of finding y where:

$$y^T A \geq c, \min y^T b, y \geq 0.$$

The objective value ($y^T b$) of any feasible y (where $y^T A \geq c$) provides an upper bound on the value of the primal linear program as for a feasible x (where $Ax \leq b$) we have the following:

$$y^T b \geq y^T Ax \geq cx. \tag{2}$$

The first ineqality follows from $Ax \leq b$ and $y \geq 0$. The second from $y^A \geq c$ and $x \geq 0$. Recalling the following complementary slackness conditions for a pair of feasible solutions, (x, y):

1. $y_i \geq 0 \Rightarrow a_i x = b_i$.
2. $x_j \geq 0 \Rightarrow y^T a^{(j)} = c_j$.

Exercise: convince yourself that the above conditions imply that both x and y are optimal. Use equation 2, and show each complementary slackness implies that one of the inequalities holds with equality.

The maximum matching problem can be formulated as a linear program as well. In particular, for a graph, $G = (V, E)$, with weights, w_e, we introduce a variable x_e for each edge where when $x_e = 1$ when e is in a matching. The constraints are

$$\forall v \in V, \sum_{e=(u,v)} x_e \leq 1,$$

which ensure that the number of matched edges to any vertex is at most 1. The objective function is

$$\max \sum_e w_e x_e,$$

which "counts" the number of edges in the matching. We pause to note there is no reason that the solution is integer, i.e., in this cae that $x_e = \{0, 1\}$. For example a graph that connsists of a cycle has a solution where $x_e = 1/2$ for all edges. We note that this particular linear program has an integer solution; one proof being the algorithm in the previous section.

In any case, the dual of the linear program is is composed of variables p_u for each vertex in V and the constraints,

$$\forall e = (u, v), p_u + p_v \geq w_e,$$

and the objective function

$$\max \sum_v p_v.$$

This is the price function from the previous section. The action of the algorithm uses this formulation and in particular complementary slackness conditions to produce a solution to maximum weight matching. It is combined with the combinatorial algorithm that is used for maximum matching: in particular, either an augmenting path adjusts the values of x_e's to increase the size of the matching in "tight" edges or a cut is found which allows the adjustment of prices, p_u's, or dual variables.