1 Weighted Bipartite Matching

For weighted bipartite graphs, we seek to find a matching that maximizes the sum of the edge weights in the matching. This is also called the **assignment problem**, as it emulates some optimal assignment between between two groups. If we reformulate our problem into considering only $K_{n,n}$ bipartite graphs by inserting edges of weight zero where edges don't exist, we can use the **Hungarian Algorithm**.

---

procedure HUNGARIAN(X, Y-bigraph G with weight $w(i,j)$ for each edge (i,j))
  $G \leftarrow$ zero weight edges to make it $K_{n,n}, n = \max(|X|, |Y|)$
  for all $i \in X, j \in Y$ **do**
    $W_{i,j} \leftarrow G(w(i,j))$ ▷ Weight between vertices i and j
  $A \leftarrow W$ ▷ Initialize A to copy of W
  for all $i \in |X|$ **do**
    $u_i \leftarrow \max_j(W_{i,j})$ ▷ Max weight in row i of W
  for all $i \in |Y|$ **do**
    $v_i \leftarrow 0$
  do
    for all $(i,j) \in |A|$ **do**
      $A_{i,j} \leftarrow u_i + v_j - W_{i,j}$
    $Z \leftarrow$ all $e(i,j) \in A_{i,j} : A_{i,j} = 0$ ▷ Create edges in Z of zero entry indices in A
    $M \leftarrow \text{MaxMatch}(Z)$ ▷ Perform maximum match on Z
    if $|M| < n$ **then** ▷ Don't have perfect match
      $Q \leftarrow \text{VertexCover}(M)$ ▷ Create vertex cover using match
      $R \leftarrow X \cap Q$
      $T \leftarrow Y \cap Q$
      $\epsilon \leftarrow \min(A_{i,j} : i \in (X - R), j \in (Y - T))$
      for all $i \in (X - R)$ **do**
        $u_i \leftarrow u_i - \epsilon$
      for all $j \in T$ **do**
        $v_j \leftarrow v_j + \epsilon$
  while $|M| < n$
  return M

---

2 Stable Matching

Instead of using explicit weights, we now consider the problem of finding a **stable matching** between two equally sized sets of elements X and Y, where $|X| = |Y| = n$, given an ordering of preferences for each element. A matching is not stable if there is an element $x \in X$ which prefers some given element $y \in Y$ over the element to which x is already

matched, and y also prefers x over the element to which y is already matched. This is also referred to as the *stable marriage problem*, as it emulates a circumstance in which two groups have preferences of partners in the opposite group. We can solve this problem using the **Gale-Shapley Algorithm**. This algorithm guarantees that all elements end up matched and all matches are stable. However, the algorithm does not necessarily guarantee optimality for all elements in each set. Since there are n elements in X, and each element might potentially perform n proposals to each element in Y, the algorithm runs in $O(n^2)$.

---

procedure GALESHAPLEY(Sets X, Y and lists of preferences $L, \forall x \in X$ and $y \in Y$)
  $M \leftarrow \emptyset$ ▷ Match initially empty
  for all $x \in X, y \in Y$ **do**
    $Free(x), Free(y) \leftarrow$ **true** ▷ All initially unmatched
    $Prop(x, y) \leftarrow$ **false** ▷ All x not yet proposed to any y
  while $\exists x \in X, y \in Y : Free(x) =$ **true and** $Prop(x, y) =$ **false do**
    $y \leftarrow$ first $y \in L(x) : Prop(x, y) =$ **false**
    $Prop(x, y) \leftarrow$ **true**
    if $Free(y) =$ **true then**
      $M \leftarrow M + (x, y)$
      $Free(x), Free(y) \leftarrow$ **false**
    else
      $x' \leftarrow M(x', y)$ ▷ y is already in M with x'
      if $L(y)(x) < L(y)(x')$ **then** ▷ y prefers x to x'
        $M \leftarrow M - (x', y)$ ▷ Remove y's old match
        $M \leftarrow M + (x, y)$ ▷ Create new match for y, x
        $Free(x) \leftarrow$ **false**
        $Free(x') \leftarrow$ **true**
  return M
