Lydia Feng

CS323

Writ 5

23 April 2017

1. Flow Check

   Look for an augmenting path by checking to see if there is a path from s to t, such that the flow can be increased on forward edges or decreased on backward edges. To see whether another augmenting path exists, run a breadth-first search from vertex s. As vertices are discovered, check their residual capacity. Only add that vertex to the BFS queue if the forward edge is not full or if the backward edge is not empty. This finds a path, if one exists, from s to t only using the available edges. This takes time proportional to V + E, because it checks every edge and may enqueue every vertex. If the BFS cannot find a path, that means that all paths from s to t are blocked by a full forward edge or an empty backward edge. If this is the case, then flow *f* is a maxflow. If a path exists, then flow *f* is not a maxflow.

2. Dual LP’s

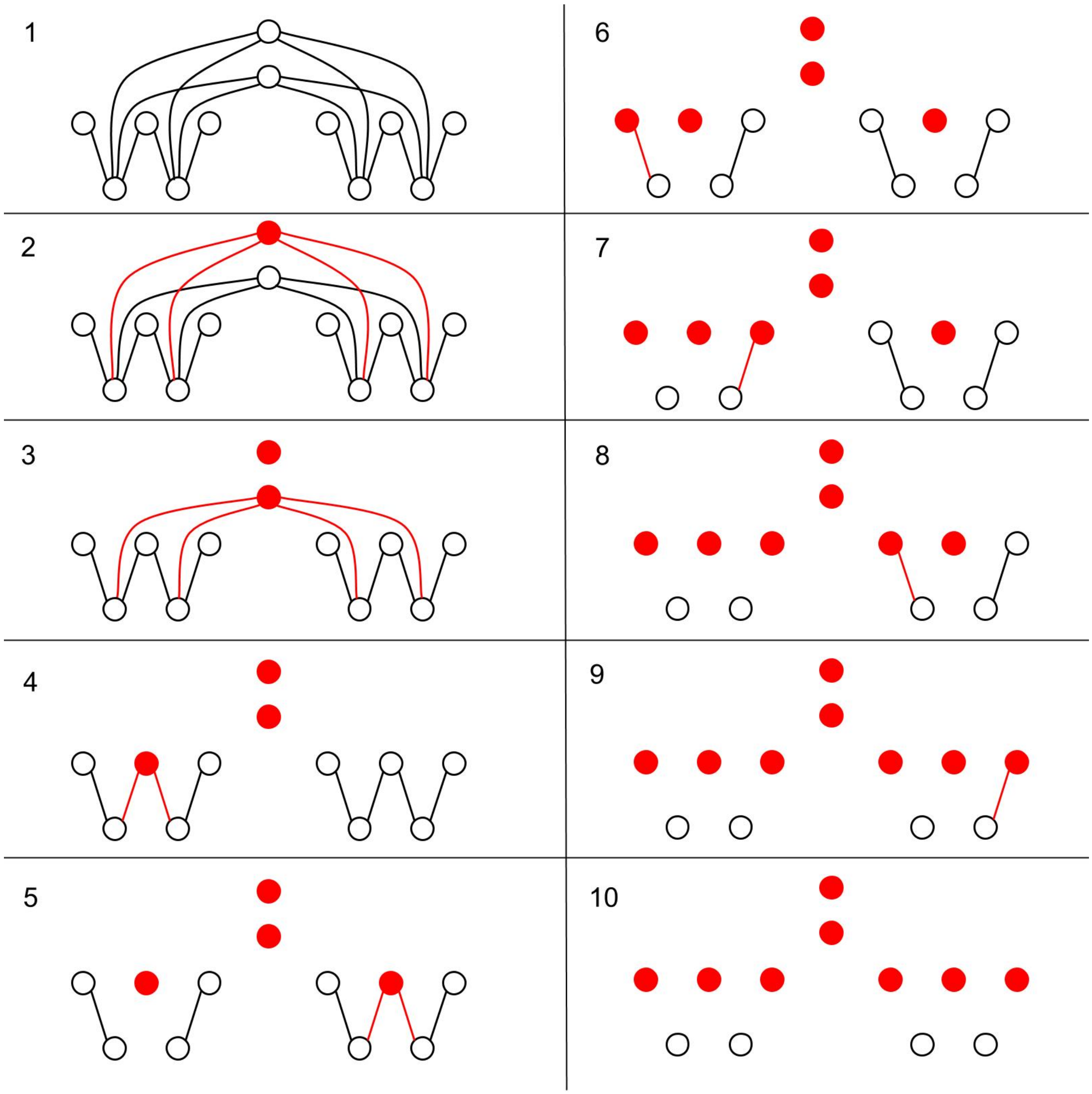
    a. See bm-min.lp file

    b. Select as few vertices as possible such that each edge is touched by at least one vertex. The duality theorem states that the min bipartite matching problem is the same size as the max bipartite matching dual.

3. Greedy VC

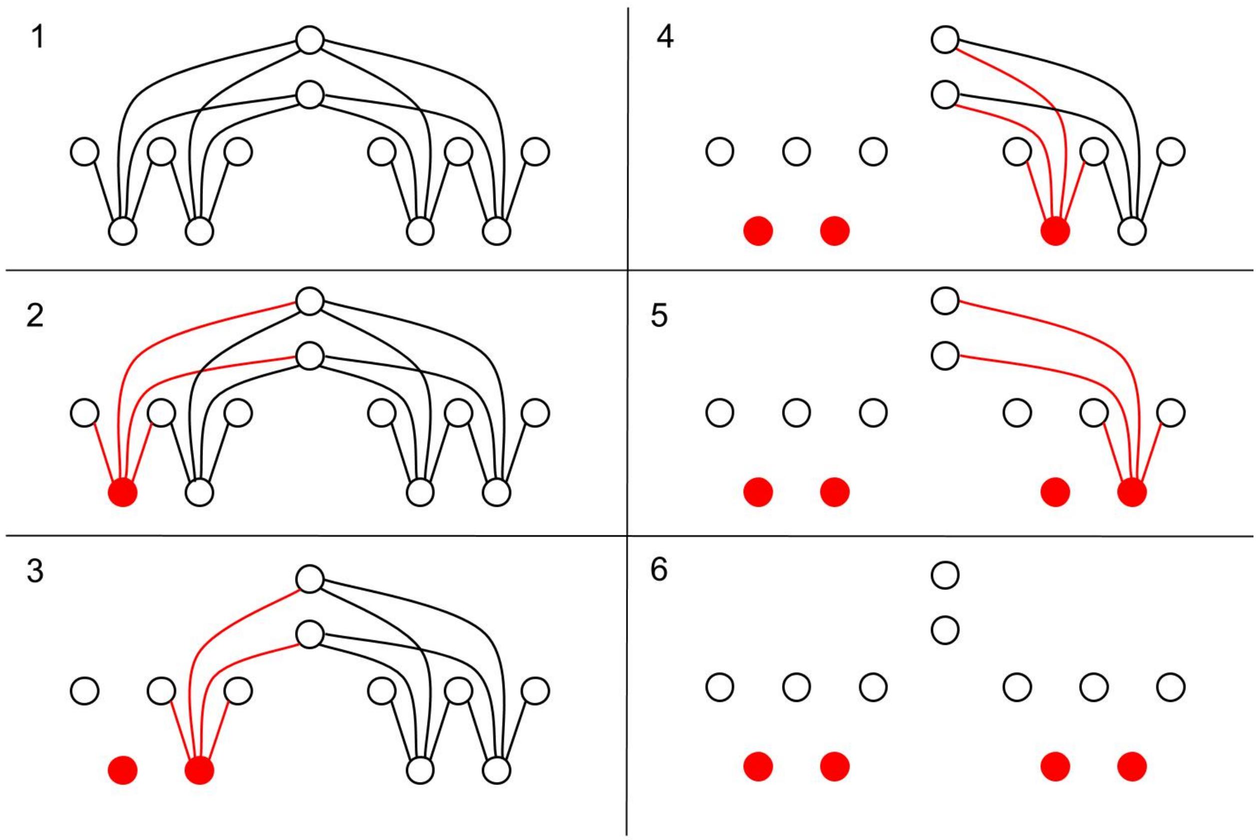
   a. Trace of GreedyVC(G); red vertices are in C

      Vertex cover is 8 vertices.



b. Trace of optimal vertex cover of G; red vertices are in C

Vertex cover is 4 vertices.



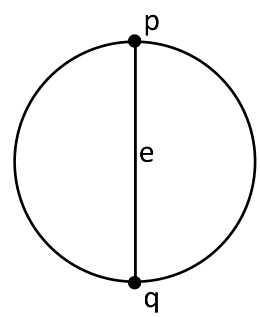
4. Numbers Game

    a. 1 3 20 2

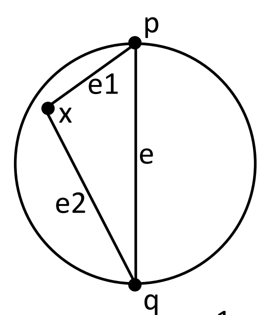
    If A executes a greedy move, A would claim 2 since it is larger than 1. This would allow B to claim 20 and win the game. If A instead claims 1, B would be forced to claim either 3 or 2. Either way, A would win the game by claiming 20 in their next move.

    b. If both players play optimally, A's total income and A's best first move can be calculated in O(N^2) time The set of available numbers can be represented as the set $\{c_i, c_{i+1}, \ldots, c_{j-1}, c_j\}$ where i and j are non-negative integers that are less than the total number of starting number N. In A's turn, A can select either c_i or c_j. To decide which choice is optimal, A must calculate the maximum value earned if c_i is chosen and the maximum value earned if c_j is chosen. To solve this, we can store the value of the intermediate subgames in an N x N array. The base case is if i == j and there is just one number in the subset. In this case, the value of the subset is the value of the card. This occurs on the diagonal of the N x N array which will be referred to as arr[][]. The rest of the table is filled in by finding the max value of choosing c_i and the max value of choosing c_j. The value of choosing c_i is c_i plus the min of the resulting subgame's value (since the max will be chosen by the opponent on their turn). This is the min of arr[i+2][j] and arr[i+1][j-1]. The value of choosing c_j is c_j plus the min of arr[i+1][j-1] and arr[i][j-2]. In this way, the table is built on its prior values,and is done with a nested for loop which will take O(N^2) time. A's total outcome is the value at arr[0][N-1] because it will be the value for the entire game $\{c_0, \ldots, c_N\}$. A's best first move can be found by comparing the values of arr[0][N-2] and arr[1][N-1]. Once the table is built, its values can be accessed in constant time.

5. EMST



Given a minimum spanning tree of S with edge ***e***, consider a point ***x*** in S that is in the interior of the circle whose diameter is the edge ***e*** from point ***p*** to point ***q***. Since ***e*** is the diameter and since the point ***x*** is inside the circle, the edge ***e1*** from point ***p*** to any arbitrary point ***x*** is shorter than ***e***. Likewise, the edge ***e2*** from any arbitrary point ***x*** to point ***q*** is shorter than ***e***. By definition, a minimum spanning tree is an acyclic spanning tree with the lowest weighted edges. By definition of a Euclidean MST, these edge weights are distances. Therefore, since ***e1*** and ***e2*** are spanning edges and since they are both shorter than ***e***, edges ***e1*** and ***e2*** would be in the EMST and edge ***e*** would not be in the EMST. This is a contradiction, since we assumed that edge ***e*** was in the EMST.



e1, e2 < e