CSC 373 Tutorial Solutions for Week 12 Winter 2019

1. Let OPT(x) be the maximum value of any solution and let A(x) be the value of the solution returned by the algorithm, on input x. Then, A(x) <= OPT(x) (by definition of OPT) and the approximation ratio is the factor by which A(x) is smaller than OPT(x): A(x) >= OPT(x) / r(n) -- or equivalently, OPT(x) <= r(n) * A(x).

2. Let G = (V,E) where V = {v_1,...,v_n}, E = {(v_1,v_i) : i >= 2}. Then {v_1} is a vertex cover of size OPT = 1 in G but, if the algorithm begins with v_1, it will output {v_2,...,v_n} instead, with size A = n-1. The ratio is therefore at least A(G)/OPT(G) = (n-1)/1 = n-1.

   The algorithm must examine every edge in the graph, for every node that is considered -- this takes time Omega(mn).

3. (a) Repeatedly pick a vertex of largest degree and put it in C, removing all edges incident on that vertex, until no edge remains.

   (b) Counting degree of each vertex: O(n + m) -- linear time.
   Placing each vertex in a priority queue, by degree -- O(n).
   Extracting each vertex from the priority queue -- O(n log n).
   Updating degrees for each vertex removed -- O(m log n) overall.
   TOTAL = O((n + m) log n).

   (c) Ratio not constant.

   - Start with edges (a_1,b_1), ..., (a_k,b_k)

   - Add vertices c_1, ..., c_j as follows:
     . create groups of 3 {b_1,b_2,b_3}, {b_4,b_5,b_6}, ...
       and connect each one to a new c vertex
       (this may leave some b vertices unconnected, which is OK)
     . create groups of 4 {b_1,b_2,b_3,b_4}, ...
       and connect each one to a new c vertex
       (this may leave some b vertices unconnected, which is OK)
     . ...
     . create a group of k-1 {b_1,b_2,...,b_{k-1}}
       and connect it to a new c vertex
     . create a group of k {b_1,b_2,...,b_k}
       and connect it to a new c vertex

   - The last c vertex created has degree k, but every other vertex has degree at most k-1, so the last c vertex will be picked first. After it is removed, every vertex will have degree at most k-2 except for the second-last c vertex (with degree k-1), so it will be picked next. This will continue picking c vertices one by one (they will always have the largest remaining degree) until only the edges (a_1,b_1),...,(a_k,b_k) remain, at which point one vertex will be picked from each edge.

   - Total number of vertices picked = k + number of c vertices. Number of c vertices = floor(k/3) + floor(k/4) + ... + floor(k/k) and it’s possible to prove that this grows like the Harmonic series (k + k/2 + ... + k/k = k * H_k), i.e., there are

Theta(k log k) c vertices.
So the algorithm picks Theta(k log k) vertices when the minimum vertex cover {b_1,...,b_k} has size only k.
This means that in general, approximation ratio >= log k.

- In particular, try this with k = 10: the graph will contain 12 c vertices and the algorithm will pick 22 vertices, more than twice the minimum of 10.

* Proof that approximation ratio is <= log n? See Sections 9.2.1 and 5.4 in the textbook. [This is NOT required reading, just for your own reference.]