Chapter 36

Exercises - Approximation Algorithms

By Sariel Har-Peled, September 25, 2014[1] Version: 1.0

This chapter include problems that are realted to approximation algorithms.

36.1 Greedy algorithms as approximation algorithms

36.1.1 Greedy algorithm does not work for TSP with the triangle inequality.

(20 pts.)

In the greedy Traveling Salesman algorithm, the algorithm starts from a starting vertex $v_1 = s$, and in i-th stage, it goes to the closest vertex to v_i that was not visited yet.

1. (10 pts.) Show an example that prove that the greedy traveling salesman does not provide any constant factor approximation to the TSP.

   Formally, for any constant $c > 0$, provide a complete graph G and positive weights on its edges, such that the length of the greedy TSP is by a factor of (at least) c longer than the length of the shortest TSP of G.

2. (10 pts.) Show an example, that prove that the greedy traveling salesman does not provide any constant factor approximation to the TSP with *triangle inequality*.

   Formally, for any constant $c > 0$, provide a complete graph G, and positive weights on its edges, such that the weights obey the triangle inequality, and the length of the greedy TSP is by a factor of (at least) c longer than the length of the shortest TSP of G. (In particular, *prove* that the triangle inequality holds for the weights you assign to the edges of G.)

36.1.2 Greedy algorithm does not work for VertexCover.

(10 pts.)

Extend the example shown in class for the greedy algorithm for Vertex Cover. Namely, for any n, show a graph G_n, with n vertices, for which the greedy Vertex Cover algorithm, outputs a vertex cover which is of size $\Omega(Opt(G_n) \log n)$, where $\mathrm{Opt}(G_n)$ is the cardinality of the smallest Vertex Cover of G_n.

36.1.3 Greedy algorithm does not work for independent set.

(20 PTS.)

A natural algorithm, GREEDYINDEPENDENT, for computing maximum independent set in a graph, is to repeatedly remove the vertex of lowest degree in the graph, and add it to the independent set, and remove all its neighbors.

1. (5 PTS.) Show an example, where this algorithm fails to output the optimal solution.

2. (5 PTS.) Let G be a $(k, k+1)$-uniform graph (this is a graph where every vertex has degree either k or $k+1$). Show that the above algorithm outputs an independent set of size $\Omega(n/k)$, where n is the number of vertices in G.

3. (5 PTS.) Let G be a graph with average degree δ (i.e., $\delta = 2|E(G)|/|V(G)|$). Prove that the above algorithm outputs an independent set of size $\Omega(n/\delta)$.

4. (5 PTS.) For any integer k, present an example of a graph G_k, such that GREEDYINDEPENDENT outputs an independent set of size $\leq |OPT(G_k)|/k$, where $OPT(G_k)$ is the largest independent set in G_k. How many vertices and edges does G_k has? What it the average degree of G_k?

36.1.4 Greedy algorithm does not work for coloring. Really.

(20 PTS.)

Let G be a graph defined over n vertices, and let the vertices be ordered: $v_1, \ldots, v_n$. Let G_i be the induced subgraph of G on $v_1, \ldots, v_i$. Formally, $G_i = (V_i, E_i)$, where $V_i = \{v_1, \ldots, v_i\}$ and

$$E_i = \left\{ uv \in E \;\middle|\; u, v \in V_i \text{ and } uv \in E(G) \right\}.$$

The greedy coloring algorithm, colors the vertices, one by one, according to their ordering. Let k_i denote the number of colors the algorithm uses to color the first i vertices.

In the i-th iteration, the algorithm considers v_i in the graph G_i. If all the neighbors of v_i in G_i are using all the k_{i-1} colors used to color G_{i-1}, the algorithm introduces a new color (i.e., $k_i = k_{i-1} + 1$) and assigns it to v_i. Otherwise, it assign v_i one of the colors $1, \ldots, k_{i-1}$ (i.e., $k_i = k_{i-1}$).

Give an example of a graph G with n vertices, and an ordering of its vertices, such that even if G can be colored using $O(1)$ (in fact, it is possible to do this with two) colors, the greedy algorithm would color it with $\Omega(n)$ colors. (Hint: consider an ordering where the first two vertices are not connected.)

36.1.5 Greedy coloring does not work even if you do it in the right order.

(20 PTS.)

Given a graph G, with n vertices, let us define an ordering on the vertices of G where the min degree vertex in the graph is last. Formally, we set v_n to be a vertex of minimum degree in G (breaking ties arbitrarily), define the ordering recursively, over the graph $G \setminus v_n$, which is the graph resulting from removing v_n from G. Let $v_1, \ldots, v_n$ be the resulting ordering, which is known as MIN LAST ORDERING.

1. (10 PTS.) Prove that the greedy coloring algorithm, if applied to a planar graph G, which uses the min last ordering, outputs a coloring that uses 6 colors.[②]

[②] There is a quadratic time algorithm for coloring planar graphs using 4 colors (i.e., follows from a constructive proof of the four color theorem). Coloring with 5 colors requires slightly more cleverness.

2. (10 pts.) Give an example of a graph G_n with $O(n)$ vertices which is 3-colorable, but nevertheless, when colored by the greedy algorithm using min last ordering, the number of colors output is n.

36.2 Approxiamtion for hard problems

36.2.1 Even More on Vertex Cover

1. (3 pts.) Give an example of a graph for which Approx-Vertex-Cover always yields a suboptimal solution.

2. (2 pts.) Give an efficient algorithm that finds an optimal vertex cover for a tree in linear time.

3. (5 pts.) (Based on CLRS 35.1-3)

   Professor Nixon proposes the following heuristic to solve the vertex-cover problem. Repeatedly select a vertex of highest degree, and remove all of its incident edges. Give an example to show that the professor's heuristic does not have an approximation ratio of 2. [Hint: Try a bipartite graph with vertices of uniform degree on the left and vertices of varying degree on the right.]

36.2.2 Maximum Clique

(10 pts.)

Let $G = (V, E)$ be an undirected graph. For any $k \geq 1$, define $G^{(k)}$ to be the undirected graph $(V^{(k)}, E^{(k)})$, where $V^{(k)}$ is the set of all ordered k-tuples of vertices from V and $E^{(k)}$ is defined so that $(v_1, v_2, ..., v_k)$ is adjacent to $(w_1, w_2, ..., w_k)$ if and only if for each i $(1 \leq i \leq k)$ either vertex v_i is adjacent to w_i in G, or else $v_i = w_i$.

1. (5 pts.) Prove that the size of the maximum clique in $G^{(k)}$ is equal to the k-th power of the size of the maximum clique in G.

2. (5 pts.) Argue that if there is an approximation algorithm that has a constant approximation ratio for finding a maximum-size clique, then there is a fully polynomial time approximation scheme for the problem.

36.2.3 Pack these squares.

(10 pts.)

Let R be a set of squares. You need to pack them inside the unit square in the plane (i.e., place them inside the square), such that all the squares are interior disjoint. Provide a polynomial time algorithm that outputs a packing that covers at least $OPT/4$ fraction of the unit square, where OPT is the fraction of the unit square covered by the optimal solution.

36.2.4 Smallest Interval

(20 pts.)

Given a set X of n real numbers $x_1, \ldots, x_n$ (no necessarily given in sorted order), and $k > 0$ a parameter (which is not necessarily small). Let $I_k = [a, b]$ be the shortest interval that contains k numbers of X.

1. (5 pts.) Give a $O(n \log(n)$ time algorithm that outputs I_k.

2. (5 PTS.) An interval J is called 2-cover, if it contains at least k points of X, and $|J| \leq 2|I_k|$, where $|J|$ denote the length of J. Give a $O(n \log(n/k))$ expected time algorithm that computes a 2-cover.

3. (10 PTS.) (hard) Give an expected linear time algorithm that outputs a 2-cover of X with high probability.

36.2.5 Rectangles are Forever.

(20 PTS.)

A rectangle in the plane r is called *neat*, if the ratio between its longest edge and shortest edge is bounded by a constant α. Given a set of rectangles R, the induced graph G_R, has the rectangles of R as vertices, and it connect two rectangles if their intersection is not empty.

1. (5 PTS.) (hard?) Given a set R of n neat rectangles in the plane (not necessarily axis parallel), describe a polynomial time algorithm for computing an independent set I in the graph G_R, such that $|I| \geq \beta |X|$, where X is the largest independent set in G_R, and β is a constant that depends only on α. Give an explicit formula for the dependency of β on α. What is the running time of your algorithm?

2. (5 PTS.) Let R be a set of rectangles which are axis parallel. Show a polynomial time algorithm for finding the largest independent set in G_R if all the rectangles of R intersects the y-axis.

3. (10 PTS.) Let R be a set of axis parallel rectangles. Using (b), show to compute in polynomial time an independent set of rectangles of size $\Omega(k^c)$, where k is the size of the largest independent set in G_R and c is an absolute constant. (Hint: Consider all vertical lines through vertical edges of rectangles of R. Next, show that by picking one of them "cleverly" and using (b), one can perform a divide and conquer to find a large independent set. Define a recurrence on the size of the independent set, and prove a lower bound on the solution of the recurrence.)

36.2.6 Graph coloring revisited

1. (5 PTS.) Prove that a graph G with a chromatic number k (i.e., k is the minimal number of colors needed to color G), must have $\Omega(k^2)$ edges.

2. (5 PTS.) Prove that a graph G with m edges can be colored using $4\sqrt{m}$ colors.

3. (10 PTS.) Describe a polynomial time algorithm that given a graph G, which is 3-colorable, it computes a coloring of G using, say, at most $O(\sqrt{n})$ colors.