

# CS 49/149: Approximation Algorithms

Problem set 1. Due: 7th April, 6:59pm

**General small print:** Please submit all homework electronically in PDF format ideally typeset using LaTeX. You need to submit only the problems above the line. Please try to be concise – as a rule of thumb do not take more than 1 (LaTeX-ed) page for a solution. We highly encourage students to also do the problems below the line for a better understanding of the course material.

**Collaboration Policy:** You are allowed to discuss with other students but are *not allowed* to exchange full solutions. You are also not supposed to search for solutions on the web and refer to any published papers or course notes other than the ones posted. You can refer the textbooks but not their solution manuals.

**Topics in this HW:** Introduction, Greedy

**Problem 1.** (The Travelling Salesman Path Problem (TSPP).) Given a collection on  $n$  points  $V$  in a metric space with two special points source  $s$  and destination  $t$ , the TSPP is to find the cheapest path from  $s$  to  $t$  which visits every other point exactly once. Before proceeding, convince yourself that the TSP algorithms done in class do not directly imply TSPP algorithms.

1. Given a graph  $G = (V, E)$  with  $s, t \in V$  such that degree of  $s$  and  $t$  are odd while the degree of every other vertex is even, prove there is a walk starting at  $s$  and ending at  $t$  which visits every edge exactly once. Use this to show that if we find such a graph  $F$ , then there is a TSPP solution of cost at most  $c(F)$ .
2. Observe that the cost of the minimum spanning tree is still a lower bound on  $OPT$ . Use this to get a factor 2-approximation for TSPP. Is the factor better than 2?
3. Get a strictly better than factor 2-approximation using matchings. Think about the vertices which need to be matched, and how you will argue about the existence of matchings of small cost. This is slightly trickier than what we did in class, and you may want to go and take a harder look at part (b).

**Problem 2.** Recall submodular functions which we defined in class. These are set functions  $f$  which prescribe a non-negative value to subsets of a finite universe  $E$ . These are denoted as  $f : 2^E \rightarrow \mathbb{R}_{\geq 0}$  where  $2^E$  denotes the collection of all subsets of  $E$ . We also assume  $f(\emptyset) = 0$ . Such a set function  $f$  is called

- **Monotone** if for any  $A \subseteq B$ , we have  $f(A) \leq f(B)$ .
- **Submodular** if for any  $A \subseteq B$  and  $i \notin B$ , we have  $f(A \cup i) - f(A) \geq f(B \cup i) - f(B)$ .

The next two problems are about submodular functions and generalizations of the two greedy algorithms we did in class.

1. Given any monotone, submodular function  $f$  over a universe  $E$ , and given costs  $c_1, \dots, c_m$  for every element in  $E$ , find a set  $A$  of minimum cost such that  $f(A) = f(E)$ . What factor approximation can you get for this problem?
2. Given any monotone, submodular function  $f$  and a parameter  $k$ , find a subset  $A \subseteq U$  with  $|A| = k$  and  $f(A)$  as large as possible. What factor approximation can you get for this problem?

**Problem 3.** Give an algorithm to find a Eulerian tour in an Eulerian graph. What is the running time of your algorithm?

**Problem 4.** The 2-factor approximation algorithm for maximum matching done in class runs in  $O(m)$  time where  $m$  is the number of edges. Generalize the algorithm to obtain a  $3/2$ -factor approximation algorithm running in  $O(m)$  time. Hint: A matching  $M$  which has no length 3 alternating paths has the property that  $|M| \geq \frac{2}{3}\text{opt}$ .

**Problem 5.** (Finding tight examples.) We design and analyze algorithms in class. One question we must ask whether we can analyze better. Doing this exercise will help you understand the algorithms. Below, when I ask to find a “ $\alpha$ -bad example” for an approximation algorithm, what I want is an instance or a family of instances (of growing size), such that the algorithm (for a minimization problem) returns a solution whose cost is at least  $\geq \alpha(1 - o(1)) \cdot \text{OPT}$ . Here the  $o(1)$  often comes to picture when the size of the instances go to infinity. Find

1. A  $3/2$ -bad example for the Christofides algorithm.
2. A  $\ln n$ -bad example for the Greedy algorithm for Set Cover. (We already say an  $\Omega(\ln n)$ -bad example in class.)
3. A  $(1 - (1 - 1/k)^k)$ -bad example for Max- $k$ -Coverage.