DESIGN AND ANALYSIS OF ALGORITHMS
Homework 10

Aman Choudhary
MTech Coursework, CSA 2020
Sr No: 17920

January 4, 2021

1 Problem 1

1.1 Notation

- $G = (V, E)$ is any general graph, which is the input to our algorithm.
- M is the matching returned by the algorithm.
- M_{opt} is the cardinality of maximum matching in G.

1.2 Claim 1: M is a maximal matching.

Let us assume otherwise, that M is not a maximal matching. This means that there exists at least one edge $e \in E$, such that $M \cup \{e\}$ is also a matching for G. However, if that were really the case, then our algorithm would have considered it in STEP 2, and included it in M in the first place. Therefore, we contradict our assumption, and conclude that our claim is true.

1.3 Claim 2: $|M| \geq |M_{opt}|/2$

For, every edge $e \in M_{opt}$, at least one of its end vertices must be matched in M. If this is not true, then both end vertices of e are not incident with any edge in M. This would mean that we could have added e to M, and hence the matching reported by our algorithm is not maximal. However, this contradicts our established Claim 1. As a result, we can say there would never be any such edge. This leads to the fact, that at least $|M_{opt}|$ vertices are matched in M. We can match $|M_{opt}|$ vertices using $|M_{opt}|/2$ edges. Thus, we can claim that M consists at least $|M_{opt}|/2$ edges.

1.4 Approximation Ratio

For a maximization problem, an algorithm is said to achieve an approximation ratio of α if and only if for any problem instance I of P,

$$ALG(I) \geq \alpha \, OPT(I)$$

In our case, $ALG(I)$ represents the size of maximal matching M returned by given algorithm, while $OPT(I)$ represents the cardinality of maximum matching, M_{opt}.

From Claim 2, we have shown that,

$$|M| \geq \frac{1}{2}|M_{opt}|$$

Hence, the best possible approximation ratio for the following algorithm is $\alpha = 0.5$.

2 Problem 2

3 Notation

- m is the no. of given subsets, while $n = |U|$, and k is the cardinality constraint.
- OPT_k denotes the value of an optimal solution (i.e., the size of the union of any k given subsets is at most OPT_k).
- b_i denotes the no. of elements covered after picking i sets.
- C is the collection of sets, returned by the algorithm.
- C_i is the set, which maximizes the number of newly covered elements in the i^{th} iteration.

3.1 Algorithm

1. $C = \phi$
2. **for** i = 1 to k
3. Let C_i be the subset which maximizes the number of newly covered elements. Perform $C = C \cup C_i$
4. **return** $C = \{C_1, C_2, ..., C_k\}$

3.2 Analysis

- The above algorithm is a **greedy approximation** to the **maximum-coverage problem**, whose approximation factor is $\left(1 - \frac{1}{e}\right)$ (all intermediate results used here, and below have been proved in class).
- The algorithm runs for k iterations, which is bounded by m.
- In every iteration, for each set (that is not yet picked), we measure the marginal increase in the number of covered elements, if that set is picked. Finally, we pick C_i, which maximizes the number of newly covered elements.
- The check in previous step, can be performed by taking union of the set in consideration with the already covered elements, and see the difference. Taking union of two sets whose sizes can atmost be n is $O(n^2)$. The number of unions per iteration depends on the number of sets not yet picked, which is bounded by m.
- The time-complexity is thus:

  No. of iterations * No. of sets not-yet picked * Time to take union = $k * O(m) * O(n^2) = O(m^2 n^2)$.

  Hence, the algorithm runs in **polynomial time**.

- We know that, if we pick k elements using this greedy algorithm, then, $b_k \geq OPT_k - \left(1 - \frac{1}{k}\right)^k OPT_k$. Now, we need to pick k' sets, such that,

$$b_{k'} \geq OPT_{k'} - \left(1 - \frac{1}{k'}\right)^{k'} OPT_{k'} \geq OPT_k$$

  We need to show that, $k' = 2k\ logn$ satisfies the above condition.
