6.889 Sublinear Time Algorithms February 20, 2019

Lecture 5

Lecturer: Ronitt Rubinfeld *Scribe: Shiyin Wang*

1 Outline

- Relationship between Greedy Algorithm and Sublinear Algorithm through Maximal Matching
- Property Testing

2 Maximal Matching

2.1 Definition

Definition 1 (Maximal Matching) *$M \subset E$, s.t.*

1. *No 2 edges in M share an endpoint*
2. *$\forall (u, v) \in E \setminus M$, either u or v is matched by M*

We have to emphasis that the matching we discussed in this lecture is maximal, not maximum. Maximum matching is a NP-hard problem.

2.2 Relationship to Vertex Cover

It is interesting that the answers of these two graph problem can be bounded by each other. Specifically,

$$\#Maximal\ Matching \leq \#Minimum\ Vertex\ Cover \leq 2\#Maximal\ Matching \tag{1}$$

According to the definition of maximal matching, if we choose all the two nodes in a maximal matching to cover the graph, then it is a vertex cover. Because every edge in $E \setminus M$ has one endpoint in M.

On the other hand, just consider covering the edges in the M. Because such edges don't share an endpoint, obviously we need to choose one endpoint of this edges to be the cover vertex. Thus the number of cover vertex chosen should be no smaller than the number of edges in a maximal matching.

2.3 Greedy Algorithm and Oracles

Algorithm 1: Greedy Algorithm for Maximal Matching

```
Input  : Graph $G = (V, E)$
Output: Maximal Matching $M$
1 $M \leftarrow \emptyset$;
2 for $(u, v) \in E$ do
3     if Neither $u$ or $v$ previously matched in $M$ then
4         Add $e$ to $M$
5 Return $M$
```

This algorithm seems to be very simple and correct, right? But there are two things we need to consider: The first one is what is the order of iterate all the edges in E? The second one is how can we check whether an edge is in M? For the first oracle, we can rank the edges in an arbitrary order.

2.4 Oracle Reduction Framework

A general approach of making this problem sub-linear is to sample a portion of data and calculate the size of maximal matching and scale the result to the whole graph, as we did in previous lectures. It seems to be obvious but actually we need an oracle to check whether edge (v, w) is in M, which we will discuss in the following section.

Algorithm 2: Maximal Matching

```
1 $S \leftarrow \frac{8}{\epsilon^2}$ random chosen nodes;
2 for $v \in \mathcal{S}$ do
3     $x_v \leftarrow 0$;
4     for $\forall w \in Neighbor(V)$ do
5         if $(v, w) \in M$ then
6             $x_v \leftarrow 1$
7 Output: $\frac{n}{2S} \sum_{v \in S}(x_v) + \frac{\epsilon n}{2}$
```

2.5 Implementing the Oracle

Under the sub-linear framework, it is impossible to store all the edges in a list/set and refer to it every time. Luckily, we have a clever approach given the edge selection order.

Algorithm 3: Check whether $e \in M$

```
  Input  : Graph $G = (V, E)$
  Output: Maximal Matching $M$
1 $M \leftarrow \emptyset$;
2 for $(u, v) \in E$ do
3     if Neither $u$ or $v$ previously matched in $M$ then
4         Add $e$ to $M$
5 Return $M$
```

If a edge has the minimal rank among all the edges adjacent to its two endpoints, then it is in M.

Given an edge, we can do tree search on all its two endpoints. If the edge has the lowest rank then we stop this branch. Otherwise, continue to search along the edges with lower rank.

Claim 1 *To check whether a specific edge e is in M, the expected number of queries is $2^{O(d)}$ in our algorithm.*

Proof Since we run tree search along the two endpoints of e, each path has decreasing rank.

$$Pr(path\ of\ length\ k\ has\ decreasing\ order) = \frac{1}{k!}$$

For the graph with bounded degree d, there are $(2d)^k$ in each layer. So the total expectation of queries is

$$E[\#queries] = \sum_{k \geq 1} \frac{(2d)^k}{k!} \leq \frac{e^{2d}}{(k+1)!} = 2^{O(d)}$$

Under bounded degree, the expectation of queries is constant. ■

3 Property Testing

Definition 2 *Graph G is $\varepsilon-$ close to having property P if we can add/delete no more than $\varepsilon d_{max} n$ edges to get $G' \in P$.*

Our goal is to test whether a graph has property P with accurate result $\geq \frac{2}{3}$: $\forall G \in P$, pass G; $\forall G$ which is ϵ-far from P, we should declare a failure.

Definition 3 (minor) *H is a minor of G if we can obtain H from G through vertex removals, edge removals or edge contractions. Else we can call G is H-minor free.*

Definition 4 (minor-closed property) *If $G \in P$, then all minors of G are also in P.*

Theorem 2 (Robertson & Seymour) *Every minor closed property is expressible as constant number of excluded minors.*

For example, planar graph is equivalent to the absence of K_5 and $K_{3,3}$.

Definition 5 ((ϵ, k)-hyperfinite) *A graph is said to be (ϵ, k)-hyperfinite if we can remove $\leq \epsilon n$ edges such that the graph can be transformed into components which all have size no larger than k. k can be a function of ϵ.*

This property can be interpreted as we can split the graph into bounded size components after deleting a small fraction of edges.