CS 7880: Algorithms for Big Data (Fall'22) **Northeastern University**

Lecture 7

September 30, 2022

Instructor: Soheil Behnezhad *Scribe: Anamay Chaturvedi*

1 Sublinear Algorithms via Maximal Independent Set

In this lecture, we'll see examples of sublinear time algorithms for the *maximal independent set* (MIS) problem. Let us start with the definition of a maximal independent set.

Definition 1 (Maximal independent set (MIS))**.** Given a graph $G = (V, E)$, an *independent* set $I \subseteq V$ is a set such that if $u, v \in I$, then $(u, v) \notin E$. In other words, there should be no edges between vertices in I. A *maximal* independent set (MIS) is an independent set such that adding any vertex $v \in V \setminus I$ to I would violate the independence property of I. Alternatively, I is an MIS if it is independent, and for every vertex $v \in V \setminus I$, there is an edge between v and some vertex $u \in I$.

One can similarly define the problem of finding a *maximum independent set*, i.e. a MIS of maximum size. However, this problem turns out to be NP-complete.[1]

We first describe a simple greedy algorithm `GreedyMIS` that for any ordering of the vertices of G (say according to a permutation π), iterates over the vertices and adds them to the solution as long as the independence property is not violated.

```
GreedyMIS(Graph $G$, permutation $\pi$).
  1. $I \leftarrow \emptyset$
  2. Iterate over the nodes of $G$ in the order of $\pi$:
      (a) Let $v_i$ be the current vertex
      (b) If $v_i$ has no neighbours in $I$, add $v_i$ to $I$
  3. Return $I$
```

Proposition 2. *For all π, `GreedyMIS`(G, π) returns an MIS for G.*

Proof. Suppose to the contrary that the set I returned by `GreedyMIS` is not an MIS. If I is not independent, then there must be some $u, v \in I$ such that $(u, v) \in E$. If $\pi(u) < \pi(v)$ then when v is processed it will be found that u is already in I and v would not be added, which is a contradiction. On the other hand if I is independent but not maximal, then let v be any vertex which can be added to I without violating independence. Then, when v is processed by `GreedyMIS` (as every vertex must be), it would have been added to I as none of its neighbours lie in I, which is again a contradiction. $\square$

We will be particularly interested in the output of GreedyMIS when the ordering is chosen uniformly at random from the set of all possible permutations. This is known as the *randomized greedy MIS* algorithm.

It is not hard to see that both GreedyMIS and RandomizedGreedyMIS require $\Theta(m + n)$ time to implement in general. In fact, it can be shown that *any* algorithm for computing any MIS needs $\Omega(m + n)$ time.

[1] In fact the maximum independent set problem is hard to even approximate within a $n^{1-\epsilon}$ factor!

However, consider a more relaxed problem of finding out whether a given vertex v is part of GreedyMIS over some given permutation π. Can we answer these kind of queries without constructing the whole MIS? As we will see, the following claim, first observed by Onak and Nguyen [2], provides a way to do so.

Claim 3. *For any vertex $v \in G$, $v \in \texttt{GreedyMIS}(G, \pi)$ if and only if it is the case that for all neighbours u of v such that $\pi(u) < \pi(v)$, $u \notin \texttt{GreedyMIS}(G, \pi)$.*

Proof. If for any neighbour u of v it is the case that $u \in \texttt{GreedyMIS}(G, \pi)$ then $v \notin \texttt{GreedyMIS}(G, \pi)$ by the independence of `GreedyMIS`. On the other hand, when we process v according to permutation π, only the neighbours u with $\pi(u) < \pi(v)$ have been seen. So if none of these is in I, v will be added. □

Building on this claim, Onak and Nguyen [2] they introduced the following local procedure which given a vertex v of a graph G and a permutation π determines if $v \in \texttt{GreedyMIS}(G, \pi)$.

```
IsInMIS(vertex $v$, permutation $\pi$) [2]:
  1. Let $u_1, \dots, u_d$ be neighbours of $v$ in $G$ such that $\pi(v_1) < \cdots < \pi(v_d) < \pi(v)$
  2. For $i = 1, \dots, d$:
      (a) If IsInMIS($u_i, \pi$) $= True$:
            i. return $False$
  3. Return $True$
```

But is this local procedure really faster than constructing the whole MIS? This is not the case for all permutations π. Indeed one can come up with a pathological permutation on which it is as slow[2] as constructing the whole MIS. Nonetheless, for random π the situation is different. Let us start with a definition.

Definition 4 (Query Complexity)**.** For any vertex v and any permutation π define $Q(v, \pi)$ to be the total number of times that function `IsInMIS` is recursively called when we call $\texttt{IsInMIS}(v, \pi)$.

Onak and Nguyen [2] proved the following bound on the query complexity $Q(v, \pi)$ for any vertex v provided that π is random.

Theorem 5 (Onak, Nguyen; FOCS'08, [2])**.** *For any vertex v and a random permutation π,*

$$\mathbb{E}_\pi[Q(v, \pi)] \leq \frac{e^\Delta}{\Delta},$$

where Δ is the maximum degree of the graph G.

Proof. We observe that if a vertex u that is k steps away from v is queried, then there must be a path from v to u whose ranks are decreasing as one moves from v to u. There are Δ^k paths of length k from v and each is decreasing with probability $1/(k+1)!$. Thus, by linearity of expectation we have that

$$\begin{aligned}\mathbb{E}_\pi[Q(v, \pi)] &= \sum_{k=0}^{\infty} \mathbb{E}[\text{\# of times vertices of distance } k \text{ from } v \text{ are called}] \\ &\leq \sum_{k=0}^{\infty} \frac{\Delta^k}{(k+1)!} \\ &\leq \frac{e^\Delta}{\Delta},\end{aligned}$$

[2]Or even a much slower exponential time, since we are not caching previous queries to the oracle!

wherein we use that $\exp(\Delta) = \sum_{k=0}^{\infty} \frac{\Delta^k}{k!} \geq \sum_{k=0}^{\infty} \frac{\Delta^k}{(k+1)!}$ (this is the Taylor-Maclaurin series of the function $\exp(x)$ in x). $\square$

In this analysis of the Query complexity of `IsInMIS`, we did not exploit the fact that the algorithm terminates as soon as it encounters a vertex for which the recursive call to `IsInMIS` returns *True*. In fact, Onak and Nguyen [2] left it as one of their open problems to see whether this early pruning has any effect on the query complexity or not. A year later, Yoshida, Yamamoto, and Ito [3] gave an ingenious analysis, proving that pruning indeed has a significant effect provided that the starting vertex is also chosen uniformly at random. They proved that:

Theorem 6 (Yoshida, Yamamoto, Ito; FOCS'09 [3])**.** *Let v be a vertex chosen uniformly at random from an n-vertex m-edge garph $G = (V, E)$, and let π be a random permutation (independent from v). Then*

$$\mathbb{E}_{v,\pi}[Q(v, \pi)] \leq 1 + \frac{m}{n}.$$

Note that in this theorem the expectation of the query complexity is also taken over the uniformly random vertex v that is sampled, so it cannot be directly compared to Theorem 5.

Remark. The more general problem of deriving a tighter bound on the query complexity for an arbitrary vertex v remains open. In particular, the following question is open:

Problem 1. Prove or disprove that $\mathbb{E}_{\pi}[Q(v, \pi)] \leq \text{poly}(\Delta, \lg n)$ for any vertex $v \in V$.

In the rest of this lecture, we prove Theorem 6. Our proof slightly deviates from the original proof of [3] and is closer to the analysis in [1]. However, the key ideas are the same as the original paper of [3].

Towards proving Theorem 6, let us first make the following auxiliary definitions.

Definition 7. For a vertex v, define $P(v, \pi)$ to be the total number of times that `IsInMIS`(v, π) is called if we call `IsInMIS`(u, π) for every $u \in V$. Similarly, for any directed vertex (u, v) define $P(u, v, \pi)$ to be the total number of times that calls to `IsInMIS`(u, π) directly call `IsInMIS`(v, π) if we call `IsInMIS`(w, π) for every $w \in V$.

Remark. Observe that, in a sense, $P(v, \pi)$ is the *complement* of $Q(v, \pi)$. In words, while $Q(v, \pi)$ counts the number of queries *out of* vertex v, $P(v, \pi)$ counts the number of queries *to* the vertex v. This definition is the key behind the analysis of [3]. Somehow, miraculously, Yoshida, Yamamoto, and Ito [3] realized that bounding $P(v, \pi)$ would be easier than $Q(v, \pi)$.

The following lemma wherein lies the bulk of the analysis, is the key to the proof of Theorem 6.

Lemma 8. *For any $(u, v) \in E$, $\mathbb{E}_{\pi}[P(u, v, \pi)] \leq 1/2$.*

Before proving Lemma 8, let us first see why it implies Theorem 6.

Proof of Theorem 6 by Lemma 8. We have that

$$\begin{aligned}\mathbb{E}_{v,\pi}[Q(v, \pi)] &= \frac{1}{n} \sum_{v \in V} \mathbb{E}_{\pi}[Q(v, \pi)] = \frac{1}{n} \sum_{v \in V} \mathbb{E}_{\pi}[P(v, \pi)] = \frac{1}{n} \sum_{v \in V} \left(1 + \sum_{u \in N(v)} \mathbb{E}_{\pi}[P(u, v, \pi)]\right) \\ &= \frac{1}{n} \sum_{v \in V} \left(1 + \frac{\deg(v)}{2}\right) \qquad \text{(by Lemma 8.)}\end{aligned}$$

$$= 1 + \frac{m}{n}.$$

In the above, we use linearity of expectation and the fact that $\sum_v Q(v,\pi) = \sum_v P(v,\pi)$ to get from the first equation to the second; this follows essentially from the definitions of the two terms $Q(\cdot,\cdot)$ and $P(\cdot,\cdot)$. To get from the second equation to the third, we simply use the definition of $P(\cdot,\cdot)$. □

Proof of Lemma 8. Our proof plan for bounding $\mathbb{E}[P(u,v,\pi)]$ is as follows: For any permutation π, we *blame* $P(u,v,\pi)$ other permutations over $[n]$. We then, as the main step of the proof, show that every permutation of $[n]$ is in total blamed at most once. This also implies that a random permutation π *blames* at most 1 other permutation and hence $\mathbb{E}[P(u,v,\pi)] \le 1$.[3]

Let us fix a permutation π. Suppose we call $\texttt{IsInMIS}(w,\pi)$ on some vertex w and at some point this recursively calls $\texttt{IsInMIS}(u,\pi)$ which itself calls $\texttt{IsInMIS}(v,\pi)$. At this point, consider the stack of recursive calls to function $\texttt{IsInMIS}(\cdot,\pi)$. It must be a path $P = (w,\dots,u,v)$; we call this a *query path*.

Now recall from the definition of $P(u,v,\pi)$ that we must have exactly $P(u,v,\pi)$ query paths that end at edge (u,v) in permutation π. For any such paths $P = (u_1,\dots,u_{|P|})$ ending in (u,v) (so $u_{|P|-1} = u$ and $u_{|P|} = v$) we say that we *blame* a permutation $BL(P,\pi)$ constructed as follows. For every vertex v that does not lie on P, $BL(P,\pi)(v) = \pi(v)$, and for every vertex $v = u_i$, $BL(P,\pi)(u_i) = u_{(i+2) \bmod |P|}$. In words, the labelling of each vertex according to the blamed permutation is simply the labelling of the vertex that lies two steps ahead on the query path according to the original permutation, and the labels of the second to last and last vertices are simply the labels of the first and second vertices (in that order). This process is called *rotating* the permutation π along the path P.

Note that there is no reason for the query path P to stop at v, but for our purposes it will suffice to clip the query path at v and simplify the construction as described above. For an example of what the blamed permutation looks like on the query path, see Figure 1.

The following claim basically proves that every permutation is blamed at most once (fixing u and v).

Claim 9. *If $BL(P,\pi) = BL(P',\pi')$ then $P = P'$ and $\pi = \pi'$.*

Proof. We will prove this claim by assuming its negation and arriving at a contradiction. It is easy to see that if $P = P'$, then $\pi = \pi'$, so we assume without loss of generality that $P \ne P'$, but $BL(P,\pi) = BL(P',\pi')$. To sketch this proof, we will trace one particular example, the path blamed in Figure 2, and two distinct (path, permutation) pairs (P,π) and (P',π') that both happen to blame $BL(P,\pi)$ (Figure 3). Since P and P' are distinct by assumption, they must diverge at some point (or one must be a prefix of the other).

We observe that $\pi(1) = \pi'(1),\dots,\pi(59) = \pi'(59)$ as the permutations only differ on the vertices that occur on the two possible query paths, by construction of $BL(\cdot,\cdot)$. The fact that the vertex marked by 80 in π queries 70 implies that it has no neighbours with rank smaller than 70 that are in $\texttt{GreedyMIS}(G,\pi)$ (for otherwise the algorithm would never query the vertex marked by 70 and P would not be a valid query path). Since π and π' are the same up to rank 59, it follows that the same vertex, which is marked by 60 in π' has no neighbours with rank up to 59 that lie in $\texttt{GreedyMIS}(G,\pi')$. We see that the vertex marked by 65 in π' would first query the vertex marked by 60, and by our observation will add the vertex marked by 60 to $\texttt{GreedyMIS}(G,\pi')$, and will terminate the recursive call made to 65. In particular, it will never query the next vertex on the path P', i.e. the vertex marked by 61, which contradicts the assumption that P' is a valid query path under π'. The same construction can be obtained if P' is a sub-path of P.

More formally, let $P = (v_{|P|},\dots,v_1)$ and $P' = (v'_{|P'|},\dots,v'_1)$ - we index these paths in decreasing order for ease of notation. Let the index of the vertex at which they converge be i^*, so $v_j = v'_j$ for $j \le i^*$. Note that it is possible that $i^* = |P|$ or $i^* = |P'|$. For ease of notation let the permutation $BL(P,\pi) = BL(P',\pi')$ be denoted β. Further let the notation $P(j)$ and $P'(j)$ denote $v_{j \bmod |P|}$ and $v'_{j \bmod |P'|}$, respectively. Since P and P' diverge at v_{i^*}, it follows that $P(i^*+1) \ne P'(i^*+1)$, i.e., in the two paths P and P', the

[3]The stronger upper bound of $\mathbb{E}[P(u,v,\pi)] \le 1/2$ follows because we will actually show that half of the permutations will not be blamed at all and the other half are blamed at most once each.

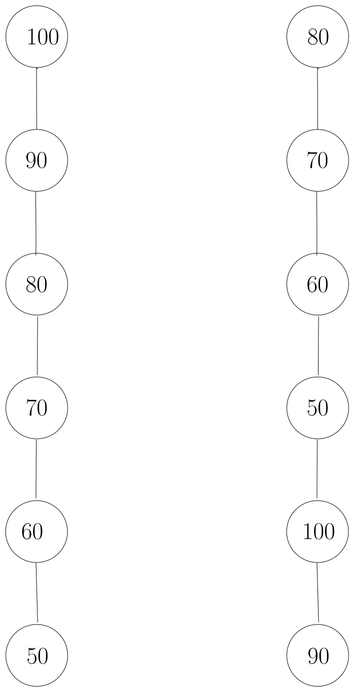

Figure 1: Query path P marked by permutation π on the left, and the marking according to the blamed permutation $BL(P, \pi)$ on the right (recall that all other vertices would be marked identically under both permutations).

vertices that immediately precede the first common vertex (in the rotation order) are distinct. It follows that $\beta(P(i^*+1)) \neq \beta(P'(i^*+1)$. We assume without loss of generality that $\beta(P(i^*+1)) < \beta(P'(i^*+1))$. It will also be helpful to observe that $\pi(u) = \pi'(u)$ for all $u \in V \backslash \{P(i^*-1), \ldots, P(|P|), P'(i^*-1), \ldots, P'(|P'|)\}$ (this follows from the fact that $BL(P, \pi = BL(P', \pi'))$.

Case 1: P is a prefix of P'. We observe that $\pi'(P'(|P'|-1))$ and $\pi'(P'(|P'|)$ are the second-largest and largest ranks on the set of all vertices considered. Further, we see that $\beta(P'(2)) = \pi'(P'(|P'|))$ and $\beta(P'(1)) = \pi'(P'(|P'|-1))$ and also that $\pi(P(i^*)) = \beta(P'(2)$ and $\pi(P(i^*-1) = \pi'(P'(|P'|-1))$. It follows that under the permutation π, before $P(i^*)$, the vertex with the highest rank, queries $P(i^*-1)$, the vertex with the second-highest rank, it must query $P'(i^*+1)$. By construction, $\pi(P'(i^*+1)) = \beta(P'(i^*+1)) = \pi'(P'(i^*-1)) = \pi'(P(i^*-1))$. Further, since $P('(i^*+1))$ queries $P'(i^*)$ under π', it follows that it has no neighbours with rank less than $\pi'(P'(i^*))$ that lie in $\texttt{GreedyMIS}(P, \pi')$. We see that since the ranks of all vertices less than $\min(\pi'(P'(i^*-1)), \pi(P(i^*-1))) \leq \pi'(P'(i^*-1))$ are identical for π and π', it follows that $P'(i^*+1)$ has no neighbours with rank less than $\pi'(P'(i^*-1))$ in $\texttt{GreedyMIS}(P, \pi)$. Since $\pi'(P'(i^*-1)) = \beta(P'(i^*+1)) = \pi(P'(i^*+1))$, it follows that when $P'(i^*+1)$ is queried by $P(i^*)$ under π, it is added to $\texttt{GreedyMIS}(P, \pi)$ and the algorithm terminates; this contradicts the assumption that P is a valid query path since $P(i^*-1)$ is not queried.

Case 2: P' is a prefix of P. Note that in the previous case we did not use appeal to the assumption that $\beta(P(i^*+1)) < \beta(P'(i^*+1)$, so this case follows from the previous case, by symmetry.

Case 3: Neither path is a prefix of the other. Here we will appeal to our assumption that $\beta(P(i^*+1)) < \beta(P'(i^*+1))$. The proof of this case follows essentially the sketched example described before. We see that $\pi'(P(i^*+1)) = \beta(P(i^*+1) < \beta(P'(i^*+1)) = \pi'(P'(i^*-1))$. It follows that under π', $P(i^*)$ will query its neighbour $P(i^*+1)$ before it queries $P'(i^*-1)$. Further, since $P(i^*+1)$ queries $P(i^*)$ under π, it follows that it has no neighbours with rank less than $\pi(P(i^*)) > \pi(P(i^*-1)) = \beta(P(i^*+1)) = \pi'(P(i^*+1))$ which lie in $\texttt{GreedyMIS}(G, \pi)$. Again, we see that since the ranks of all vertices less than $\min(\pi'(P'(i^*-1)), \pi(P(i^*-1))) =$

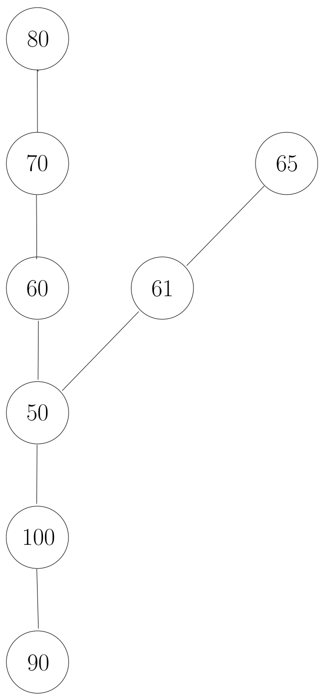

Figure 2: $BL(P, \pi)$

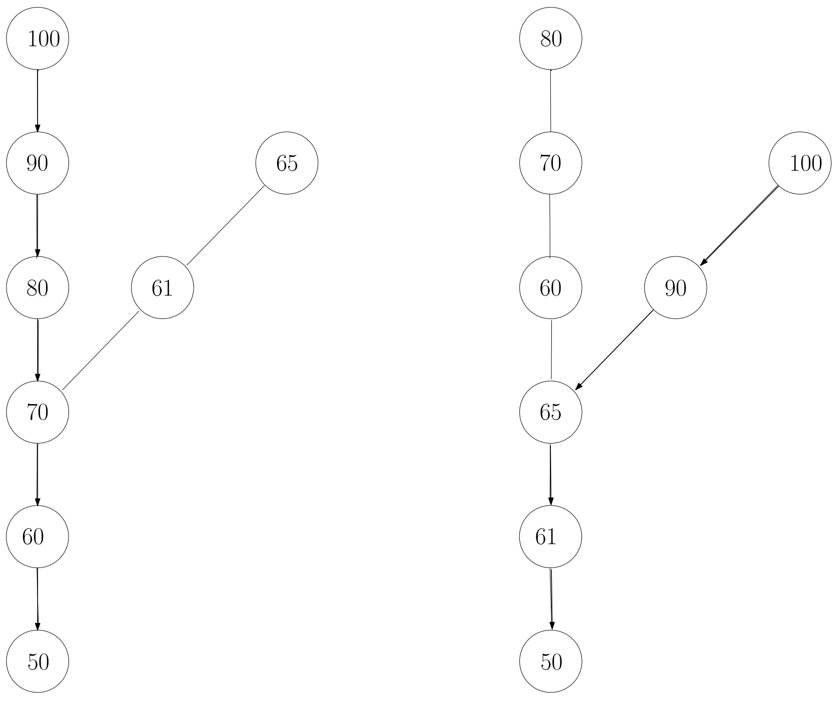

Figure 3: The two possible query paths, permutation pairs ((P, π) on the left and (P', π') on the right) which could have lead to $BL(P, \pi)$ illustrated in Figure 2.

$\min(\beta(P'(i^*+1)), \beta(P(i^*+1))) = \beta(P(i^*+1)) = \pi(P(i^*-1)) = \pi'(P(i^*+1))$ are identical for π and π', it follows that $P(i^*+1)$ has no neighbours with rank less than $\pi'(P'(i^*-1))$ in `GreedyMIS`(P, π'). It follows that when $P(i^*+1)$ is queried by $P'(i^*)$ under π', $P(i^*+1)$ must be added to `GreedyMIS`(P, π') and the algorithm must terminate; this contradicts P' being a valid query path as $P'(i^*-1)$ is never queried. □

Returning to the proof of Lemma 8,

$$\begin{aligned}
\mathbb{E}_\pi[P(u,v,\pi)] &= \frac{1}{n!}\sum_\pi P(u,v,\pi) \\
&= \frac{1}{n!}\sum_\pi \#\text{ of times that } \pi \text{ is blamed} \\
&\leq \frac{1}{n!}\left(\sum_{\pi:\pi(u)<\pi(v)} 0 + \sum_{\pi:\pi(u)>\pi(v)} 1\right) \\
&= 1/2.
\end{aligned}$$

In the above we used that for 1/2 of all permutations π, $\pi(u) < \pi(v)$ in which case u would never query v and so $P(u,v,\pi)$ would equal 0. □

In the next lecture, we will see some applications of this local algorithm for MIS.

References

[1] Soheil Behnezhad. Time-optimal sublinear algorithms for matching and vertex cover. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 873–884. IEEE, 2021. 3

[2] Huy N. Nguyen and Krzysztof Onak. Constant-time approximation algorithms via local improvements. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 327–336. IEEE Computer Society, 2008. 2, 3

[3] Yuichi Yoshida, Masaki Yamamoto, and Hiro Ito. An improved constant-time approximation algorithm for maximum matchings. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 225–234. ACM, 2009. 3