CS120: Intro. to Algorithms and their Limitations Hesterberg & Vadhan

Section 7: Matchings

Harvard SEAS - Fall 2022 *Oct. 20, 2022*

1 Interval Scheduling

Recall the definition of independent set:

Definition 1.1. Let $G = (V, E)$ be a graph. An *independent set* in G is a subset $S \subseteq V$ such that there are no edges entirely in S. That is, $\{u, v\} \in E$ implies that $u \notin S$ or $v \notin S$.

This gives rise to the following computational problem:

Input	**:** A graph $G = (V, E)$
Output	**:** An independent set $S \subseteq V$ in G of maximum size

Computational Problem Independent Set

We can try a greedy algorithm for independent set, and obtain the following bounds for both its runtime and output size:

Theorem 1.2. *For every graph G with n vertices and m edges,* `GreedyIndSet(`G`)` *can be implemented in time $O(n + m)$ and outputs an independent set of size at least $n/(d_{max} + 1)$, where d_{max} is the maximum vertex degree in G.*

Recall the optimization version of the Interval Scheduling problem:

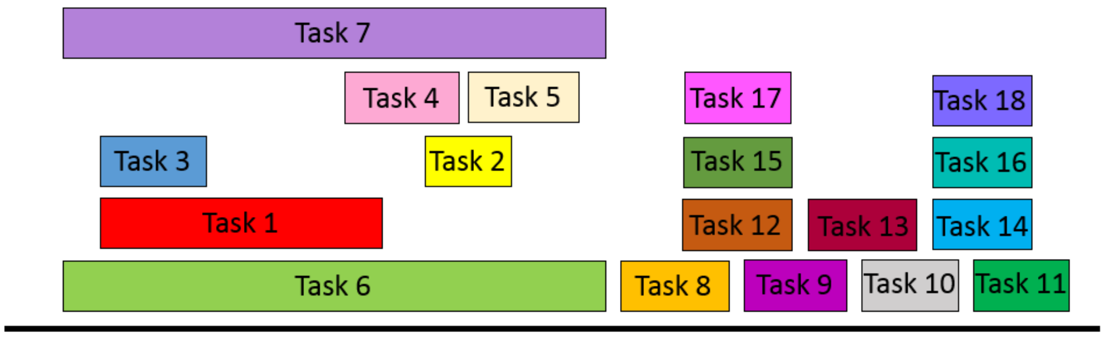
Input	**:** A collection of intervals $[a_0, b_0], \dots, [a_{n-1}, b_{n-1}]$, where each $a_i, b_i \in \mathbb{Q}$ and $a_i \leq b_i$
Output	**:** A maximum-size subset $S \subseteq [n]$ such that $\forall i \neq j \in S$, $[a_i, b_i] \cap [a_j, b_j] = \emptyset$.

Computational Problem IntervalScheduling-Optimization

This allows us to obtain the better bound:

Theorem 1.3. *If the input intervals are sorted by increasing order of end time b_i, then we have that that* `GreedyIntervalScheduling(`x`)` *will find an optimal solution to IntervalScheduling-Optimization, and can be implemented in time $O(n \log n)$.*

Question 1.4. (Example Interval Scheduling) This question illustrates an example of greedy interval scheduling. Take the following tasks, what is the greedy solution sorted by increasing start-time, increasing end-time, and decreasing start-time?



2 Matchings

Recall the definition of matching:

Definition 2.1. For a graph $G = (V, E)$, a *matching* in G is a subset $M \subseteq E$ such that every vertex $v \in V$ is incident to at most one edge in M. Equivalently, no two edges in M intersect.

Input	: A graph $G = (V, E)$
Output	: A matching $M \subseteq E$ in G of maximum size

Computational Problem Maximum Matching

How can we solve Maximum Matching? In class, we saw how Maximum Matching can be solved on **bipartite graphs**.

Theorem 2.2. *Maximum Matching can be solved in time $O(mn)$ on bipartite graphs with m edges and n vertices.*

Let's review how to prove this theorem step by step. First, what is a bipartite graph? Recall we saw examples of bipartite graphs in the kidney-donor problem in lecture.

Definition 2.3. A graph $G = (V, E)$ is *bipartite* if it is 2-colorable. That is, there is a partition of vertices $V = V_0 \cup V_1$ (with $V_0 \cap V_1 = \emptyset$) such that all edges in E have one endpoint in V_0 and one endpoint in V_1.

Recall also alternating walks and augmenting paths from lecture.

Definition 2.4. Let $G = (V, E)$ be a graph, and M be a matching in G. Then:

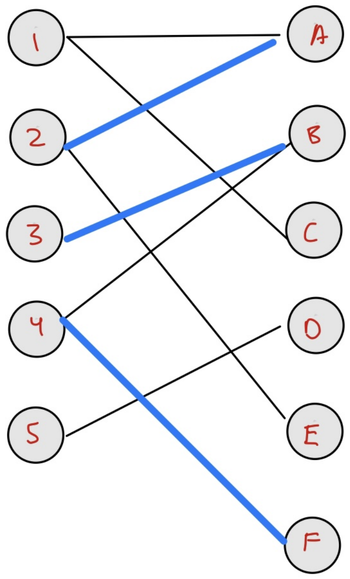
1. An *alternating walk* W in G with respect to M is a walk $(v_0, v_1, \ldots, v_\ell)$ in G such that for every $i = 1, \ldots, \ell - 1$, $\{v_{i-1}, v_i\} \in M \Leftrightarrow \{v_i, v_{i+1}\} \in E - M$.

2. An *augmenting path* P in G with respect to M is an alternating walk in which v_0 and v_ℓ are unmatched by M, and in which all of the vertices in the walk are distinct.

We will use augmenting paths to find a maximum matching using the following lemma.

Lemma 2.5. *Given a graph $G = (V, E)$, a matching M, and an augmenting path P with respect to M, we can construct a matching M' with $|M'| = |M| + 1$ in time $O(n)$.*

Question 2.6. (Example Matching Given P and M) This question illustrates an example of finding a matching given an augmenting path. Given the following bipartite graph G, which has a matching M indicated in blue (bold). Find and draw an augmenting path P with respect to M. Then, use P to construct a larger matching M'. Repeat to find an augmenting path P' with respect to M' to obtain a maximum matching M''.



In lecture, we presented the following matching algorithm using augmenting paths and Lemma 2.5:

```
1  MaxMatchingAugPaths($G$)
   Input  : A bipartite graph $G = (V, E)$
   Output : A maximum-size matching $M \subseteq E$
2  Remove isolated vertices from $G$;
3  Let $V_0, V_1$ be the bipartition (i.e. 2-coloring) of $V$;
4  $M = \emptyset$;
5  repeat
6      Let $U$ be the vertices unmatched by $M$, $U_0 = V_0 \cap U$, $U_1 = V_1 \cap U$;
7      Try to find an augmenting path $P$ that starts in $U_0$ and ends in $U_1$;
8      if $P \neq \bot$ then augment $M$ using $P$ via Lemma 2.5;
9  until $P = \bot$;
10 return $M$
```

Theorem 2.7 (Berge's Theorem)**.** *Let $G = (V, E)$ be a graph, and $M \subseteq E$ be a matching. If (and only if) M is not a maximum-size matching, then G has an augmenting path with respect to M.*

Lemma 2.8. *Let $G = (V_0 \cup V_1, E)$ be bipartite and let M be a matching in G that is not of maximum size. Let U be the vertices that are not matched by M, and $U_0 = V_0 \cap U$ and $U_1 = V_1 \cap U$. Then:*

1. *G has an alternating walk with respect to M that starts in U_0 and ends in U_1.*

2. Every shortest alternating walk from U_0 to U_1 is an augmenting path.

Correctness:
We can show correctness of `MaxMatchingAugPaths` using Lemma 2.7. If M is not a maximum matching, then there is an augmenting path from U_0 to U_1, which we will find and use to increase the size of M by 1. Since a matching can have at most $n/2$ edges, `MaxMatchingAugPaths` will always halt within $n/2$ iterations of the loop, and when it does, it will output a maximum-size matching (since no augmenting path from U_0 to U_1 exists).

Runtime:
We can find augmenting paths by finding shortest alternating walks, which is a special case of a k-rotating walk (where $k = 2$, as in PS4). So, a shortest alternating walk can be found in $O(m)$ time. (After removing isolated vertices, we have $n \leq 2m$, so $O(n + m) = O(m)$.) Then, augmenting the matching using the augmenting path takes time $O(n)$ by Lemma 2.5. The process has to be repeated at most $n/2$ times (as in the correctness proof), so the total runtime is $(n/2) \cdot O(m) = O(mn)$.

3 Problems

Question 3.1. (Bipartite matching for assigning workers to jobs.) Every year, Harvard OCS decides which students should get which jobs. They know that some students are suited for some jobs but not for other, and they have a list for every student of which companies they could work at, and which they couldn't.

1. Draw a non-trivial example scenario graph with 5 students and 5 companies such that a 1:1 matching is possible
   Draw another scenario where there is no 1:1 matching

   How would you reduce these to instances of Independent Set? (See lecture 13)

Question 3.2. Prove the following theorem: Berge's Theorem: Let $G = (V, E)$ be a graph, and $M \subseteq E$ be a matching. If (and only if) M is not a maximum-size matching, then G has an augmenting path with respect to M.

Question 3.3. Prove the following theorem: Let $G = (V_0 \cup V_1, E)$ be bipartite and let M be a matching in G that is not of maximum size. Let U be the vertices that are not matched by M, and $U_0 = V_0 \cap U$ and $U_1 = V_1 \cap U$. Then:

1. G has an alternating path that starts in U_0 and ends in U_1.
2. Every shortest alternating path from U_0 to U_1 is an augmenting path.

Question 3.4. (Non-Augmenting Path.) Come up with an example of a non-bipartite graph G, a matching M in G, and a shortest alternating path P with respect to M that starts and ends at unmatched vertices but is not an augmenting path (due to repeated vertices).