Most of the material in this lecture is taken from the book “Fast Parallel Algorithms for Graph Matching Problems” by Karpinski-Rytter.

We will only be considering simple undirected finite graphs unless stated otherwise. Graphs will be denoted as $G = (V, E)$.

1 Some preliminary definitions

Definition 1 Let $G = (V, E)$ be a graph. $M \subseteq E$ is called as a **matching** of $G$ if $\forall \ v \in V$ we have $|\{e \in M : v \text{ is incident on } e \in E\}| \leq 1$.

Definition 2 A matching $M$ of $G$ is said to be **maximal** if $\forall \ e \in E \setminus M$ the set of edges given by $M \cup \{e\}$ is not a matching of $G$.

Definition 3 The **size** of a matching $M$ of $G$ is the number of the edges it contains and is denoted by $|M|$.

Definition 4 A matching $M$ of $G$ is said to be **maximum** if $\forall$ matching $M'$ of $G$ we have $|M| \geq |M'|$. A maximum matching is always maximal but not vice-versa.

Definition 5 Let $M$ be matching of $G$. A vertex $v \in V$ is said to be **$M$-saturated** if $M$ contains an edge incident on $v$. Otherwise $v$ is said to be **$M$-unsaturated**.

Definition 6 A matching $M$ of $G$ is said to be **perfect** if all vertices of $G$ are $M$-saturated. A graph with an odd number of vertices can never admit a perfect matching.

Definition 7 A matching $M$ of $G$ is said to be **near-perfect** if exactly one vertex of $G$ is $M$-unsaturated. A graph with an even number of vertices can never admit a near-perfect matching.

Definition 8 Let $A \subseteq V$. A matching $M$ of $G$ is said to be **$A$-perfect** if each vertex in $A$ is $M$-saturated. A perfect matching is a $V$–perfect matching.
2 Some Remarks

Remark 9 We will later look at weighted graphs i.e graphs with a weight function \( w : E \rightarrow \mathbb{R}^+ \cup \{0\} \). There we will be interested in finding matchings of maximum weight where the weight of a matching is the sum of weights of edges which are in the matching.

Remark 10 We will consider three types of problems

- **Decision** - Does \( G \) have a matching of size \( \geq k \) ?
- **Search** - Find a matching in \( G \) of size \( \geq k \)
- **Counting** - How many matchings of \( G \) have size \( \geq k \)

3 Augmenting Paths

Definition 11 A path \( P \) in \( G \) is said to be **M-alternating** if the edges of \( P \) alternate wrt membership in \( M \).

Definition 12 A path \( P \) in \( G \) is said to be **M-augmenting** if \( P \) is a maximal \( M \)-alternating path starting and ending at vertices which are \( M \)-unsaturated. Clearly, every \( M \)-augmenting path must have odd length.

Lemma 13 Let \( G \) be a graph whose maximum degree is atmost 2. Then every component of \( G \) is either an isolated point, a path or a cycle.

**Proof:** Consider any non-isolated vertex \( v \) of \( G \). Its atmost two neighbours further have degree atmost 2 and so on. So the component of \( G \) containing \( v \) is either a path or a cycle. This holds true for all non-isolated vertices of \( G \) and hence we are done. ■

Lemma 14 (Berge 1957) A matching \( M \) is maximum iff \( G \) has no \( M \)-augmenting path.

**Proof:** Suppose \( M \) is maximum and there exists a \( M \)-augmenting path \( P \). Consider the symmetric difference \( M \oplus P \) (edges which are present in exactly one of \( M \) or \( P \)). Since \( P \) is an \( M \)-augmenting path, \( M \oplus P \) is also a matching of \( G \) and \( |M \oplus P| = |M| + 1 \).

Suppose \( G \) has no \( M \)-augmenting path and \( M \) is not maximum. Let \( M' \) be a maximum matching and so we have \( |M'| > |M| \). Consider \( M \oplus M' \). Each vertex has degree atmost 2 in \( M \oplus M' \) as each of \( M \) and \( M' \) can contribute atmost 1 each to degree of each vertex in \( M \oplus M' \). By Lemma 13, \( M \oplus M' \) consists of cycles and paths and isolated vertices. But edges of \( M \oplus M' \) are alternate in belonging exclusively to \( M \) and \( M' \). Hence each cycle must be even. So \( M' \) can score over \( M \) in size only from the paths. So, there exists atleast one path in \( M \oplus M' \) which has more number of edges from \( M' \) than from \( M \). But such a path is \( M \)-augmenting which gives a contradiction. ■
Corollary 15 (Hopcroft-Karp) Let $M^*$ be a matching of $G$. Then for any matching $M$ of $G$ such that $|M^*| \geq |M|$, we have $|M^*| - |M|$ vertex-disjoint $M$-augmenting paths.

Proof: Refer to proof of Lemma 14. Every cycle of $M \oplus M^*$ is even and every path of $M \oplus M^*$ which is not $M$-augmenting must have equal number of edges from $M$ and $M^*$ as $M^*$ is maximum. Also note that each $M$-augmenting path has exactly one edge more from $M^*$ than from $M$. So we need $|M^*| - |M|$ such paths which are all vertex-disjoint as we defined (see Definition 12) augmenting paths as maximal paths starting and ending at unsaturated points.

Corollary 16 Let $M^*$ be a maximum matching and $M$ be any matching. If $M$ is not maximum, then the shortest $M$-augmenting path has length $\leq \frac{|V|}{|M^*| - |M|} - 1$

Proof: From Corollary 15 we know that there are $|M^*| - |M|$ vertex (and hence edge)-disjoint $M$-augmenting paths. By Pigeonhole Principle, one of the paths must have at most $\frac{|V|}{|M^*| - |M|}$ vertices and thus has length at most $\leq \frac{|V|}{|M^*| - |M|} - 1$.

4 Algorithm for finding maximum matching using augmenting paths

Consider the following algorithm whose correctness follows immediately from Lemma 14

1. $M = \emptyset$
2. while there is an $M$-augmenting path $P$
   do $M \leftarrow M \oplus P$
3. return $M$

5 An $O(n^3)$ algorithm for finding maximum matching in bipartite graphs

Let $G = (A \cup B, E)$ be a bipartite graph and let $M$ be a matching of $G$. We want to find a maximum matching of $G$. Denote by $A_0, B_0$ the sets of $M$-unsaturated vertices in $A, B$ respectively. We consider a new directed graph $H$ on the vertex set $A \cup B$ and edge set $E$. Edges which are in $M$ are directed $A \rightarrow B$ and edges not in $M$ are directed $B \rightarrow A$.

Claim 17 $G$ has a $M$-augmenting path iff $H$ has a path from $B_0$ to $A_0$.  

1-3
Proof: Suppose \( G \) has a \( M \)-augmenting path say from \( u \in A_0 \) to \( v \in B_0 \). The same path directed from \( v \) to \( u \) is clearly a path in \( H \) from \( B_0 \) to \( A_0 \). 

Suppose \( H \) has a path from \( b \in B_0 \) to \( a \in A_0 \). The underlying undirected path from \( a \) to \( b \) is clearly a \( M \)-augmenting path. 

So we do \( DFS \) from \( B_0 \) and stop as soon as we reach some vertex in \( A_0 \) thus giving us a \( M \)-augmenting path \( P \). Augment the \( M \)-augmenting path and repeat same process wrt new matching \( M \oplus P \). If we ever do not reach any vertex of \( A_0 \), then we can conclude from Claim 17 that \( G \) has no \( M \)-augmenting path i.e. \( M \) is maximum.

Let us now analyse the time complexity of our algorithm. Denote \( |V| = n \) and \( |E| = m \).

1. Given an augmenting path we can augment it easily in \( O(m) \) time.

2. Assume \( |B| \leq |A| \) as otherwise we could have just swapped roles of \( A \) and \( B \) in our algorithm. Thus \( |B_0| \leq |B| \leq \frac{n}{2} \). Also at each stage of our algorithm by augmenting, we saturate a previously-unsaturated vertex from \( B \) without doing anything to the vertices which are already saturated. So we need atmost \( |B_0| \leq \frac{n}{2} \) stages.

3. At each stage the maximum number of times we need to do \( DFS \) is \( |B_0| \) as in the worst-case only the last vertex from \( B_0 \) we apply \( DFS \) to may lead to a path in \( A_0 \). Recollect that a single \( DFS \) can be done in \( O(n + m) \) time.

Thus the Total Time taken by algorithm is \( \leq \frac{n}{2} \left[ O(m) + |B_0| \times O(n + m) \right] \). However now we use a trick to shave off the \( |B_0| \) factor. Add a super-vertex \( \beta \) and draw edges directed from \( \beta \) to every point in \( B_0 \). Thus we need to apply \( DFS \) only once viz. for vertex \( \beta \). Thus the time complexity becomes \( O\left(\frac{n}{2}[m + (n + m)]\right) = O\left(n^2 + nm\right) \). Since a bipartite graph on \( n \) vertices can contain atmost \( \left(\frac{n^2}{4}\right) \) edges the time complexity of our algorithm is \( O(n^3) \).

6 Hopcroft-Karp Algorithm for finding a maximum matching in bipartite graphs in \( O(n^{2.5}) \) time

In the algorithm given in the previous section we looked for a single augmenting path at a time and augmented it. Instead we will now find a maximal family of vertex-disjoint shortest-length augmenting paths and augment all of them together in a single step. This improvement will help us to bring the time complexity down to \( O(n^{2.5}) \).

Consider the following algorithm

1. \( M = \emptyset \)
2. while there is an $M$-augmenting path, find a maximal family $\mathcal{F}$ of vertex-disjoint shortest $M$-augmenting paths

   do $M \leftarrow M \oplus \mathcal{F}$

3. return $M$

The correctness of the algorithm follows from Lemma 14

**Lemma 18** Let $M$ be a matching of $G$ and let $P$ be a $M$-augmenting path of shortest length. Let $P'$ be a $(M \oplus P)$-augmenting path. Then $|P'| \geq |P| + |P \cap P'|$

**Proof:** Consider $N = (M \oplus P) \oplus P'$. Then $N$ is clearly a matching and $|N| = |M| + 2$. Thus by Corollary 15 $M \oplus N$ contains 2 vertex-disjoint $M$-augmenting paths say $P_1$ and $P_2$. Note that $M \oplus N = P \oplus P'$ and thus we have $|P \oplus P'| \geq |P_1| + |P_2|$. But $P_1, P_2$ are both $M$-augmenting paths and $P$ is shortest $M$-augmenting path. Therefore $|P \oplus P'| \geq 2|P|$. However $|P \oplus P'| = |P| + |P'| - |P \cap P'|$ and so the desired inequality follows.

**Lemma 19** Let $M_0 = \emptyset$ and $M_1$ be a matching of $G$. Consider the sequence $M_0, M_1, M_2, M_3, ...$ where $P_i$ is shortest $M_i$-augmenting path and $M_{i+1} = M_i \oplus P_i \forall i$. Then $|P_i| \leq |P_j|$ for $i < j$ and $|P_i| = |P_j|$ implies $P_i$ and $P_j$ are vertex-disjoint.

**Proof:** Suppose that $|P_i| = |P_j|$ for some $i < j$ and $P_i$ and $P_j$ are not vertex-disjoint. Then there exist some $k, l$ such that $i \leq k < l \leq j$ and $P_k$ and $P_l$ are not vertex-disjoint and further for all $m$ between $l$ and $k$ we have $P_m$ is vertex-disjoint from both $P_k$ and $P_l$. Therefore $P_l$ is a $(M_k \oplus P_k)$-augmenting path and so by Lemma 18 we have $|P_l| \geq |P_k| + |P_l \cap P_k|$. However we are given that $|P_l| = |P_k|$ which implies that $|P_l \cap P_k| = 0$ i.e. $P_l$ and $P_k$ have no edges in common. However since $P_l$ and $P_k$ are not vertex-disjoint, they have a common vertex say $x$ and then they must have in common the edge from $M_k \oplus P_k$ which is incident on $x$ leading to a contradiction.

**Lemma 20** Let $\mathcal{F}$ be maximal (wrt inclusion) family of vertex-disjoint shortest $M$-augmenting paths. Let their common length be $l_1$. Let $l_2$ be length of shortest $(M \oplus \mathcal{F})$-augmenting path. Then $l_2 \geq l_1 + 2$

**Proof:** Let $\mathcal{F} = \{P_1, P_2, ..., P_r\}$. Let $P'$ be a shortest $(M \oplus \mathcal{F})$-augmenting path. Note that $M \oplus \mathcal{F} = (...) (M \oplus P_1) \oplus P_2) (...) \oplus P_r$. Suppose $P'$ is disjoint from each element of $\mathcal{F}$. Then $P'$ is also a $M$-augmenting path and thus $l_2 \geq l_1$. If we however have $l_1 = l_2$ then we could have added $P'$ to $\mathcal{F}$ thus contradicting its maximality. So, let us assume $P'$ has a vertex in common with atleast one path in $\mathcal{F}$. By Lemma 19 we have $l_2 > l_1$. Finally note that $l_1, l_2$ are both lengths of augmenting paths and hence must both be odd.
Let us look at the graph $H$ considered at beginning of Section 5. Let $|A \cup B| = n$ and $|E| = m$. Note that Claim 17 holds.

**Lemma 21** The algorithm described at start of Section 6 makes atmost $2\sqrt{n}$ iterations

**Proof:** Let $M^*$ be a maximum matching and let $M$ be matching after $\sqrt{n}$ iterations. By Lemma 20 length of shortest augmenting path $\geq (2\sqrt{n} - 1) \geq \sqrt{n}$. By Corollary 16 we have $\sqrt{n} \leq \text{length of shortest } M \text{-augmenting path} \leq \frac{n}{|M^*| - |M|}$ and so $|M^*| - |M| \leq \sqrt{n}$. From this point onwards even if we augment just one path in each iteration we can finish in $2\sqrt{n}$ iterations as each augmentation increases size of matching by 1.

**Lemma 22** Each iteration of the algorithm can be implemented in $O(m)$ time.

**Proof:** First we will use BFS to find the length $k$ of shortest path from $B_0$ to $A_0$ and to produce the sequence of disjoint layers $B_0 = L_0, L_1, L_2, ..., L_k \subseteq A_0$ where for all $0 \leq i < k$ the set of vertices at distance $i$ from $B_0$ and $L_k$ is the subset of $A_0$ which is at distance $k$ from $B_0$. Add a super-vertex $\beta$ and draw edges from it to all vertices of $B_0$. Start a BFS from $\beta$ to get distance of $\beta$ from $A_0$. Subtract one to get length of shortest path from $B_0$ to $A_0$. This takes $O(m)$ time.

Now consider a modified DFS which starts at a vertex $v \in B_0$ and stops as soon as it reaches a vertex say $w$ in $L_k$ and outputs this $v \rightarrow w$ path. Add this $M$-augmenting path to $F$ and delete all vertices visited in the modified DFS. (Let $x$ be a vertex seen at some $L_j$ in DFS started from $v \in B_0$. If $x$ is not on a $M$-augmenting path of length $k$ starting at $v$ then $x$ cannot be on any $M$-augmenting path of length $k$ and so we can delete all vertices visited in modified DFS which began at $v$). Redo the whole procedure now starting at another vertex in $B_0$. This clearly gives us a maximal family of vertex-disjoint shortest-length augmenting paths. Let $m_i$ be the number of edges visited in the $i^{th}$ DFS which takes $O(m_i)$ time. Noting that $m \geq \sum_i m_i$ the time taken is $O(m)$

**Theorem 23** The algorithm runs in $O(n^{2.5})$ time.

**Proof:** From Lemma 22 we know that each phase can be implemented in $O(m)$ time. Also from Lemma 21 we know that there are atmost $2\sqrt{n}$ iterations. Thus time taken by our algorithm is $O(\sqrt{n}) \cdot O(m) = O(n^{2.5})$