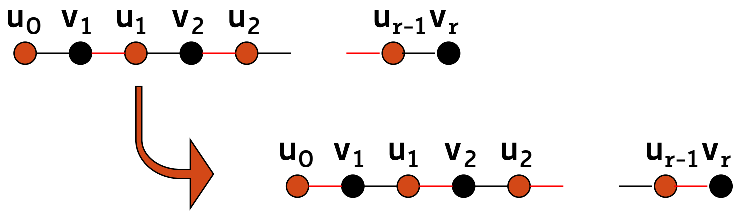
MAXIMUM MATCHING IN BIPARTITE GRAPHS (4)

(proof of the Hall theorem – contd) G bipartite with $|V_1| \le |V_2|$, G has a perfect matching iff $\forall\ S \subseteq V_1$, $|S| \le |adj(S)|$.

Continue in this way. As G is finite, we will eventually reach a node v_r that is free w.r.t. M. Each v_i is adjacent to at least one among $u_0, u_1, \ldots, u_{i-1}$.

Analogously to the case r=2:



MAXIMUM MATCHING IN BIPARTITE GRAPHS (5)

- The P. Hall Theorem <u>does not</u> provide an algorithmic method to construct a perfect matching (unless all subsets in V_1 are enumerated – exponential time issue).
- The perfect matching problem in a bipartite graph is equivalent to the maximum flow problem in a network:

  Given $G=(V=V_1 \cup V_2, E)$, construct flow network $G'=(V', E')$ as follows:

…

MAXIMUM MATCHING IN BIPARTITE GRAPHS (6)

V': V U {s} ∪ {t}

E': From the source s to all nodes in V_1 :

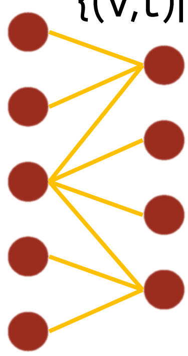
$\{(s,u) | u \in V_1\} \cup$

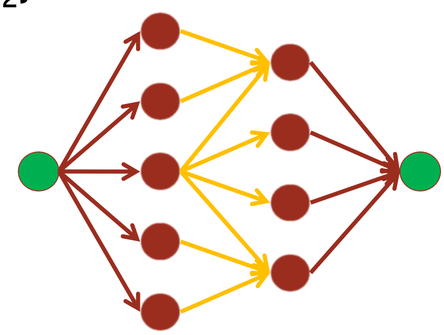
All edges in E:

$\{(u,v) | u \in V_1, v \in V_2, e\ (u,v) \in E\} \cup$

From all nodes in V_2 to the tale t:

$\{(v,t) | v \in V_2\}$



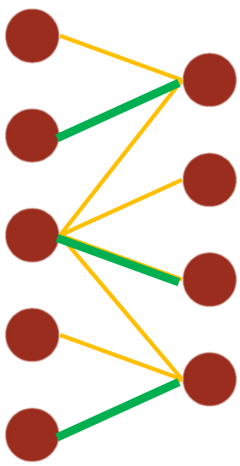


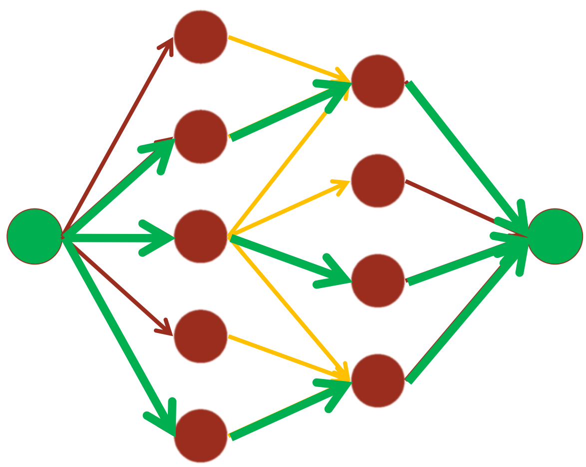
Capacity: c(u,v) = 1, for all (u,v) ∈ E'

MAXIMUM MATCHING IN BIPARTITE GRAPHS (7)

Fact: Let M be a matching in a bipartite graph G. There exists a flow f in the network G' s.t. |M|=|f|.

Vice-versa, if f is a flow of G', there exists a matching M in G s.t. |M|=|f|.





MAXIMUM MATCHING IN BIPARTITE GRAPHS (8)

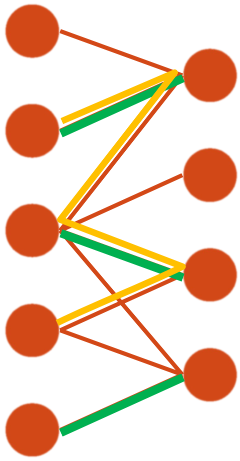
- Th.: (integrality) If the capacity c assumes only integer values, the max flow f is such that |f| is integer. Moreover, for all nodes u and v, f(u,v) is integer.
- Corol.: The cardinality of a max matching M in a bipartite graph G is equal to the value of the max flow f in the associated network G'.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (9)

- The algorithm by Ford-Fulkerson for the max flow in a network runs in O(m|f|) time.
- In our special setting, the max flow of G' has cardinality upper bounded by $\min\{|V_1|, |V_2|\}$.
- Hence, the complexity of an algorithm for the max matching exploiting the max flow runs in O(n m) time.

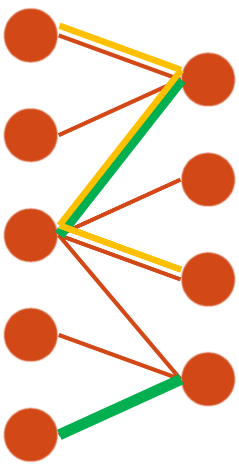
MAXIMUM MATCHING IN BIPARTITE GRAPHS (10)

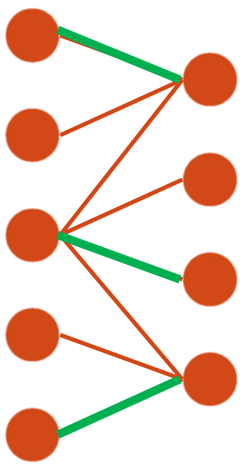
- **Def.** Given a matching M in a graph G, an alternating path w.r.t. M is the path alternating edges of M and edges in E\M.



MAXIMUM MATCHING IN BIPARTITE GRAPHS (11)

- **Def.** Given a matching M in a graph G, an augmenting path w.r.t. M is an alternating path starting and finishing in two free nodes w.r.t. M.





Swapping the role of the edges in M and in E\M,M has larger cardinality.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (12)

- Th. (Augmenting path) [Berge 1975] M is a max matching iff there are no augmenting paths w.r.t. M.
- Proof.
- (➔) If M max, then there are no augmenting paths.

  Negating, if there are some augmenting paths, then M is not max. This is obvious because we can swap the role of the edges in the augmenting path and increase the cardinality of M.
- ...

MAXIMUM MATCHING IN BIPARTITE GRAPHS (13)

(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

- (🡐) If there are no augmenting paths, then M is max.

  By contradiction, M is not max. Let M' s.t. |M'|>|M|.

  Consider graph H induced by M and M'. Edges that are both in M and in M' are put twice. So, H is a multigraph.
- ...

MAXIMUM MATCHING IN BIPARTITE GRAPHS (14)

(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

- H has the following property:
  - For each v in H, deg(v)≤2. (indeed, each node has at most one edge from M and one edge from M')
- So, each connected component of H is either a cycle or a path.
  - Cycles necessarily have even length; otherwise, a node would be incident to two edges of the same matching (M or M'); this is absurd by the definition of matching.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (15)

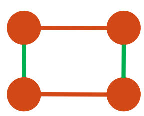
(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

- More in detail, the connected components of H can be classified into 6 kinds:

1. An isolated node

2. a 2-cycle

3. a 2k-cycle, k>1



…

MAXIMUM MATCHING IN BIPARTITE GRAPHS (16)

(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

…

4. a 2k-path

5. a (2k+1)-path whose extremes are incident to M

6. a (2k+1)-path whose extremes are incident to M’

MAXIMUM MATCHING IN BIPARTITE GRAPHS (17)

(Proof of Th. M is a max matching iff there are no augmenting paths w.r.t. M – contd)

- Reminder: |M|<|M’| by hp.
- Among all the components just defined, only 5 and 6 have a different number of edges from M and from M’; only 6 has more edges from M’ than from M.
- So, there is at least one component of kind 6

- This comp. is an augmenting path w.r.t. M: contradiction. ■

MAXIMUM MATCHING IN BIPARTITE GRAPHS (18)

- We exploit the Augmenting Path Th. to design an iterative algorithm.
- During each iteration, we look for a new augmenting path using a modified Breadth First Search starting from the free nodes.
- In this way, nodes are structured in layers.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (19)

Idea of the algorithm:

- Let M be an arbitrary matching (possibly empty)
  - Find an augmenting path P
- While there is an augmenting path:
  - Swap in P the role of the edges in and out of the matching
  - Find an augmenting path P

Complexity: it dipends on the complexity of finding an augmenting path.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (20)

Question: how to decide the existence of an augmenting path and how to find one, if one exists?

If $G=(V_1,V_2,E)$, direct edges in G according to M as follows:

- An edge goes from V_1 to V_2 if it does not belong to the matching M
- an edge goes from V_2 to V_1 if it does.

Call this directed graph D.

Claim. There exists an augmenting path in G w.r.t. M iff there exists a directed path in D between a free node in V_1 and a free node in V_2.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (21)

- Idea:
  - For each free node in V_1
  - Run a DFS on D:
    - As soon as a free node in V_2 has been encountered, a new augmenting path has been found.

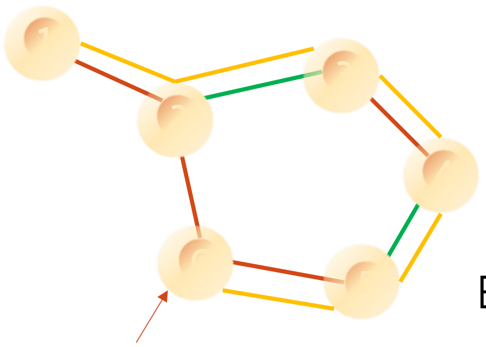
Complexity: $O(n+m)$

Complexity of the algorithm finding the max matching: $n/2[O(n+m)+O(n)]=O(nm)$

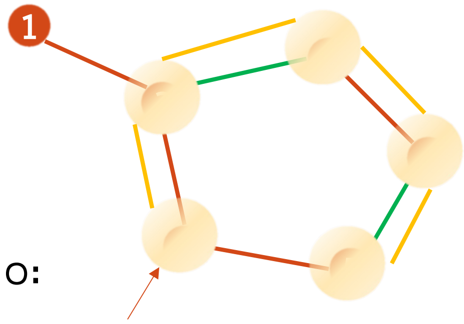
MAXIMUM MATCHING IN BIPARTITE GRAPHS (22)

(a parenthesis) What if G is not bipartite?

- For each free node
- Run a modified DFS:
  - Keep trace of the current layer
  - If the layer is even, use an edge in M
  - If the layer is odd, use an edge in E\M
  - As soon as a free node has been encountered, a new augmenting path has been found

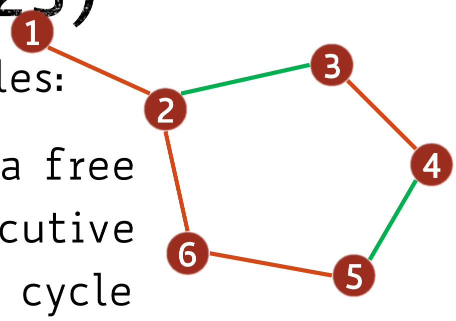


But also:



MAXIMUM MATCHING IN BIPARTITE GRAPHS (23)

Problem: the presence of odd cycles:



- In an odd cycle, there is always a free node adjacent to two consecutive edges not in M belonging to the cycle

- **If the search goes through the cycle along the "wrong" direction, the augmenting path is not detected.**

It is necessary to have graphs without odd cycles = bipartite graphs.

We will handle the general case later…

MAXIMUM MATCHING IN BIPARTITE GRAPHS (24)

- The Hopcroft-Karp algorithm (1973) finds a max matching in a bipartite graph in $O(m\sqrt{n})$ time (better than the previous $O(mn)$).
- The idea is similar to the previous one, and consists in augmenting the cardinality of the current matching exploiting augmenting paths.
- During each iteration, this algorithm searches not one but a maximal set of augmenting paths.
- In this way, only $O(\sqrt{n})$ iterations are enough.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (25)

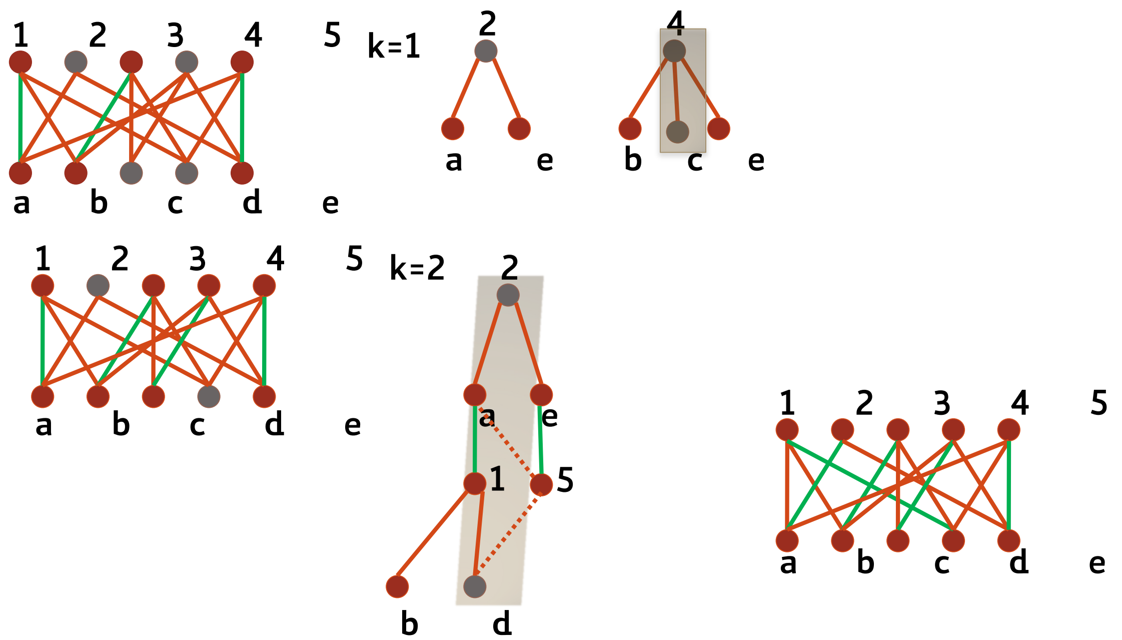
Hopcroft-Karp Algorithm

During the k-th step:

- Run a modified breadth first search starting from ALL the free nodes in V_1. The BFS ends when some free nodes in V_2 are reached at layer 2k-1.
- All the detected free nodes in V_2 at layer 2k-1 are put in a set F. Obs. v is put in F iff it is the endpoint of an aug. path
- Find a maximal set of length 2k-1 aug. paths node disjoint using a depth first search from the nodes in F back to the starting nodes in V_1 (climbing on the BFS tree).
- Each aug. path is used to augment the cardinality of M.
- The algorithm ends when there are no more aug. paths.

MAXIMUM MATCHING IN BIPARTITE GRAPHS (26)

Example: Hopcroft-Karp algorithm



MAXIMUM MATCHING IN BIPARTITE GRAPHS (27)

Analysis of the Hopcroft-Karp algorithm (sketch)

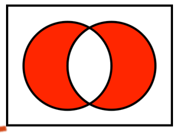
Each step consists in a BFS and a DFS. Hence it runs in Θ(n+m)= Θ(m) time.

How many steps?

- The first √n steps take Θ(m √n) time.
- Note. At each step, the length of the found aug. paths is larger and larger; indeed, during step k, ALL paths of length 2k-1 are found and, after that, only longer aug. paths can be in the graph.
- So, after the first √n steps, the shortest aug. path is at least 2√n+1 long.
- …

MAXIMUM MATCHING IN BIPARTITE GRAPHS (28)

Analysis of the Hopcroft–Karp algorithm (sketch) – contd



- The symmetric difference between a maximum matching and the partial matching M found after the first √n steps is a set of vertex-disjoint alternating cycles, alternating paths and augmenting paths.
- Consider the augmenting paths. Each of them must be at least √n long, so there are at most √n such paths. Moreover, the maximum matching is larger than M by at most √n edges.
- …

MAXIMUM MATCHING IN BIPARTITE GRAPHS (29)

Analysis of the Hopcroft–Karp algorithm (sketch) – contd

- …
- Each step of the algorithm augments the dimension of M by one, so at most √n furhter steps are enough.
- The whole algorithm executes at most 2√n steps, each running in Θ (m) time, hence the time complexity is Θ(m √n) in the worst case.