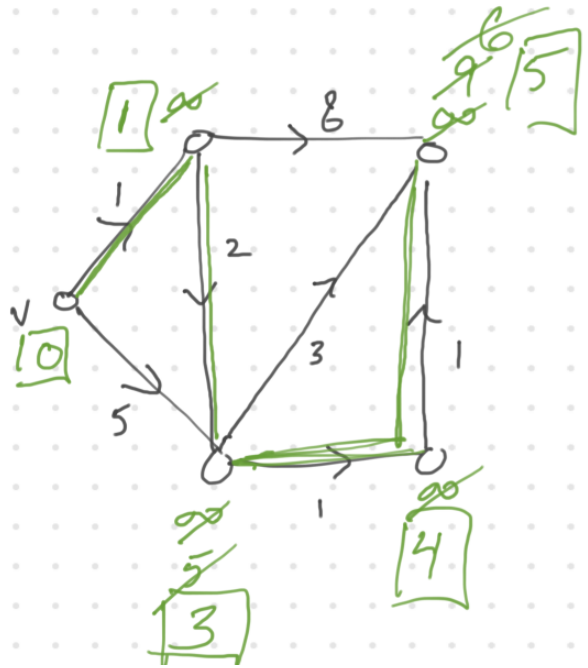
Dijkstra's algorithm

given a directed graph G w/ $w: E(G) \to \mathbb{R}^+$ & $v \in V(G)$
find min weight of a directed path from $v \to x \quad \forall x \in V(G)$



Dijkstra (G, w: , v)

Dist. Temp initiated to 0 for v
∞ otherwise

Dist. Final array initiated to ∞ for all

While Dist. temp $\neq \emptyset$

x = extract_min (Dist. temp)

Dist. final [x] = x. key

For all y. st $(x,y) \in E(G)$

if Dist. temp (y) < dist. final [x] + $w(x,y)$

Reduce ~~Dis~~ y. key to
dist. final [x] + $w(x,y)$

How many extract-mins do we do?

$O(n)$ times.

How many key reductions?

at most $O(m)$ key reductions because each edge ~~is cut~~ can cause a key reduction at most once when the tail is Dist_temp min + after that, the tail is removed + the edge isn't considered again.

Heap implementation for Dist_temp

overall complexity =

$O(n \log n)$ extractions

$O(m \log n)$ key reductions

$O(n)$ overhead

$\longrightarrow$ Total = $O((n+m)\log n)$

Fibbonacci heap

overall complexity

$O(n \log n)$ extractions

$O(m)$ for all key reductions

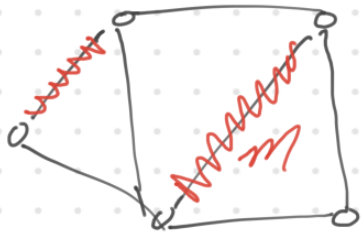
$O(n)$ overhead

$\longrightarrow$ Total $O(n \log n + m)$

Graph Algorithms

specifically, matchings in graphs

Def a matching in a graph is a set M of edges with no common endpoint between them



M is a maximal matching if $\nexists$ a matching M' s.t. $M' \supsetneq M$

M is a maximal matching

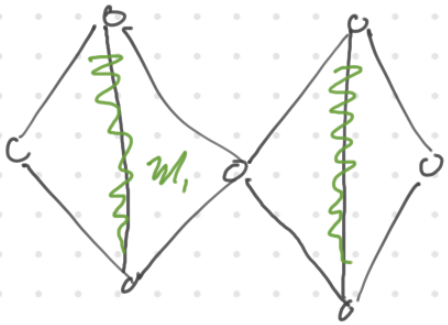
M is a maximum matching if $\nexists$ a matching M' s.t. $|M'| > |M|$

M is a maximum matching as well

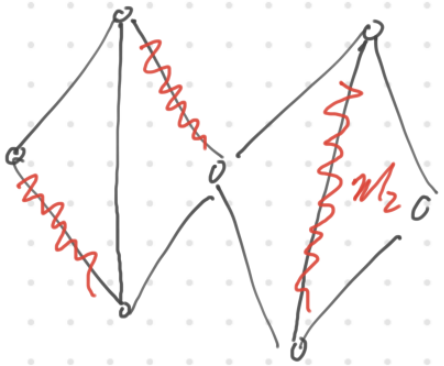
(There can't be a larger matching because $|V(G)| = 5$ & matching of size ≥ 3 would need ≥ 6 vertices)

In general, a maximal matching need not be maximum

Maximal matching vs. Maximum matching



M_1 is maximal but not maximum



M_2 is maximum.

finding a maximal matching is easy.

Alg (G a graph)

M = be empty

for $e \in E(G)$

if $M \cup e$ is a matching

$M = M \cup e$

return M.

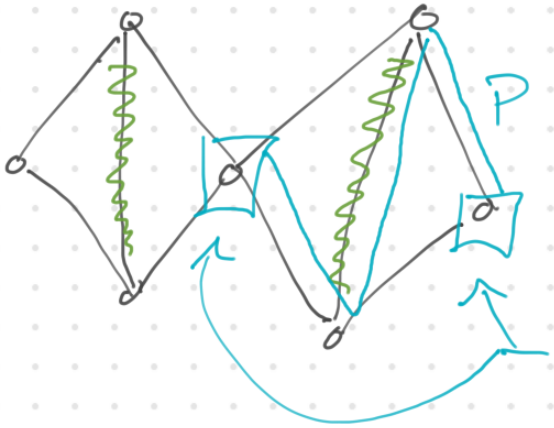
The problem of finding a maximum matching is much more interesting.

M a matching in a graph G

we say $x \in V(G)$ is <u>unmatched</u> if no edge of M touches x.

<u>Def</u> an <u>M- alternating path</u> to be a path where edges alternate between

edges in M & not in M



P is an alternating path

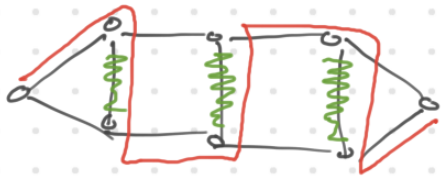
endpoints of P are unmatched

$\Rightarrow$ P is an augmenting path

Def M a matching, an M-**augmenting path** is an alternating path starting & ending at an unmatched vertex.

Obs Let M be a matching in a graph G. If $\exists$ an M-alternating path, Then $\exists$ a matching M' w/ $|M'| > |M|$.

pf



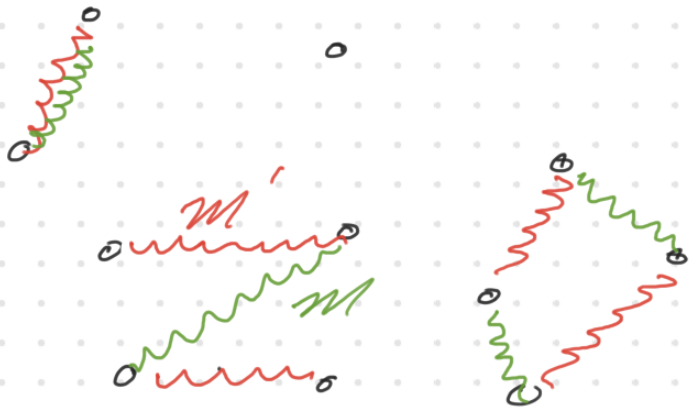
Let P be the augmenting path. Then $(M \setminus E(P)) \cup (E(P) \setminus M)$ will still

be a matching, and it will have strictly more edges. ∎

Prop G a graph, M a matching in G. Then $\exists$ a matching M' with $|M'| > |M| \iff \exists$ an M-augmenting path.

pf "$\Leftarrow$" observation above.

"$\Rightarrow$" Consider the symmetric difference of M & M' ($M \triangle M'$) defines a subgraph



$M \triangle M'$ defines a subgraph of G with vertices of deg 0, 1, or 2

$\Rightarrow$ every component is either a path or a cycle

if it is a path, it is a alternating path.

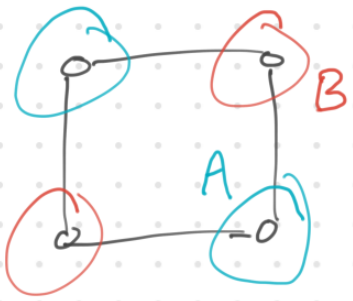
also, every cycle in $M \triangle M'$ must alternate between edges of M & $M' \Rightarrow$ each cycle in $M \triangle M'$ is even.

Conclusion: since M' has strictly more edges than M, then there must be a component of $M \triangle M'$ with strictly more M' edges than M edges $\Rightarrow \exists$ a path in $M' \triangle M$ with more M' edges $\Rightarrow$

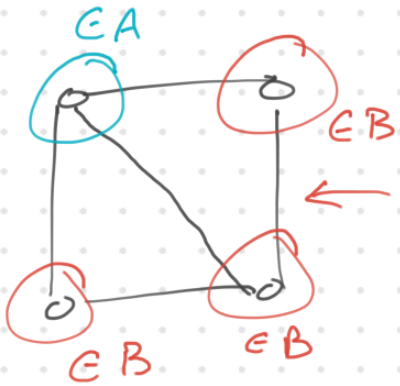
such a component starts & ends w/ an M' edge $\Rightarrow$ it is an M-augmenting path, proving the claim.

Matchings in bipartite graphs

Def a <u>bipartite</u> graph we can partition $V(G)$ in two sets A, B st every edge has one end in A & one in B.



Bipartite



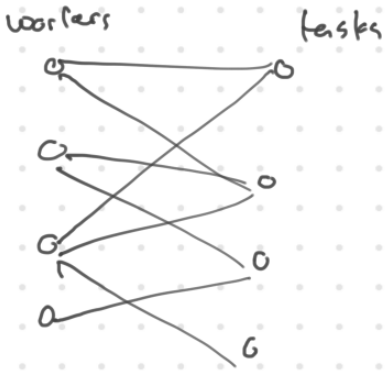
← This edge has both ends ∈ B

Not bipartite

a assignment which covers all the tasks is a matching covering the task vertices.

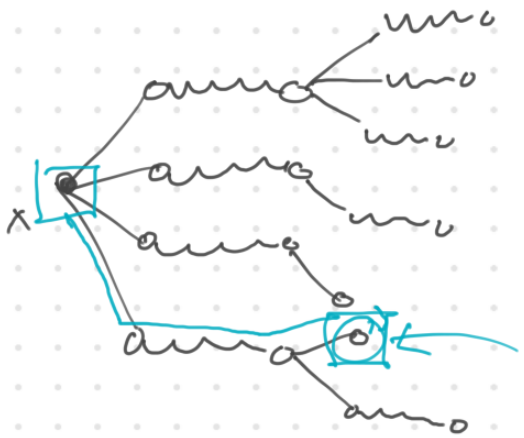
To find a maximum matching in a bipartite graph, it suffices to find an M-augmenting path given a bipartite graph G & matching M, or determine that no such path exists.

Prop G is bipartite $\Leftrightarrow$ G has no odd cycle subgraph.



& every worker is capable of completing some subset of the tasks →

We want to grow a modified BFS tree $\mathcal{B}$ w/ root x to an unmatched vertex



if $\exists$ a leaf which is unmatched, Then the tree contains a $\mathcal{M}$-augmenting path P

Alg $(G, \mathcal{M}, x)$

Q a queue $= \emptyset$ P an array w/ $P[u] = \text{NIL}$ $u \neq x$, $P[x] = x$

$Q.\text{push}(x)$

while $Q \neq \emptyset$ Do

$v.\text{pop}()$

if $(v, P[v]) \in \mathcal{M}$ or $v = x$

$\forall\, u \sim v$: if $P[u] = \text{NIL}$ { $Q.\text{push}(u)$, $P[u] = v$ }

else if $(v, P[v]) \notin \mathcal{M}$

if $\exists\, u$ ST $uv \in \mathcal{M}$ & $p[u] = \text{NIL}$

$P[u] = v$, $Q.\text{push}(u)$

Prop Let T be the BFS tree given by alg above. If $\exists$ an M-augmenting path starting at x, then $\exists$ an M-augmenting path in T

Pf exercise

Given the prop, it is easy to see if $\exists$ an M-augmenting path starting at x: grow the modified BFS tree with the algorithm & check if $\exists$ an unmatched leaf.

We get a alg to find maximum matching in bipartite graphs

we add weights to the edges $w: E(G) \to \mathbb{R}$

- maximum weight matching
  (where $w(M) = \sum_{e \in M} w(e)$)

  we can assume $w(e) \geq 0$

- maximum weight perfect matching
  (matching covering all the vertices)
  (where $w(M) \in \mathbb{R}$)

- minimum weight perfect matching
  ($w(e) \in \mathbb{R}$)

assume returns FALSE if $\nexists$ a perfect matching

Max weight perfect matching
$w: E \to \mathbb{R}$

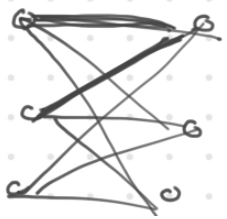
min weight perfect matching
weights $\in \mathbb{R}$

reduction: def $w'(e) = -w(e)$ + find a max weight (in w') perfect matching

max weight matching
$w: E \to \mathbb{R}^+$

add vertices to the smaller side s.t. both sides have the same # of vertices and then add an edge of weight 0 between every pair of non-adjacent vertices

max weight perfect matching in a complete bipartite graph



let $N = \max_{e \in E(G)} |w(e)|$

for every pair of non-adj vertices, add an edge w/ weight $-2N \cdot |V(G)|$

in complete bipartite graph,
if a ~~~~ p.m. uses one of
the added edges, weight
will be less than $-N \cdot |V(G)|$
if max weight perfect matching
in complete bipartite graph has
weight $< -N \cdot |V(G)|$ return FALSE
otherwise, return matching obtained.

Conclusion: it suffices to resolve
max weight perfect matching
in complete bipartite graphs
w/ weights in $\mathbb{R}$