Algorithms
Professor John Reif

ALG 5.2

Breadth-First Search of Graphs:

(a) Single Source Shortest Path
(b) Graph Matching

Main Reading Selections:
CLR, Chapter 25

Auxillary Reading Selections:
AHU-Design, Sections 5.6-5.10
AHU-Data, Sections 6.3-6.4
Handouts: "Matchings" and "Path-Finding Problems"

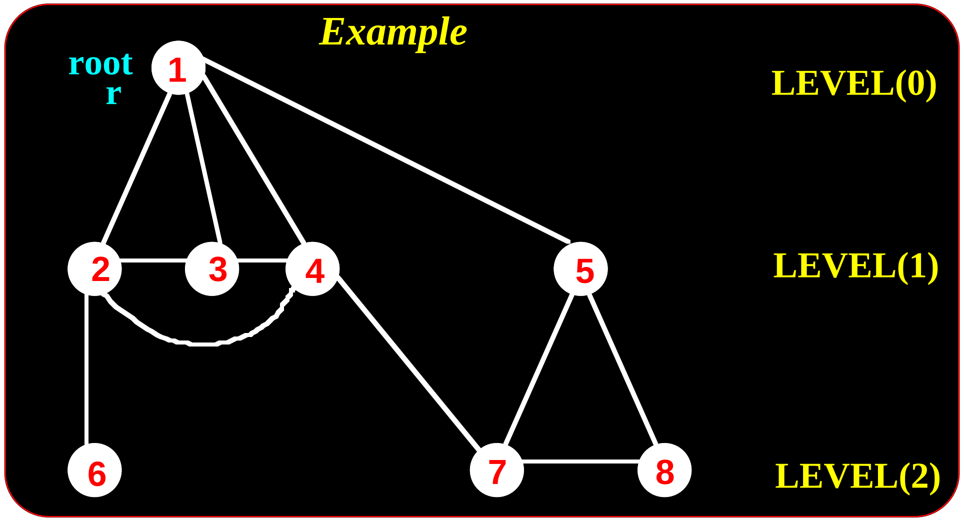
Algorithm **Breadth First Search**

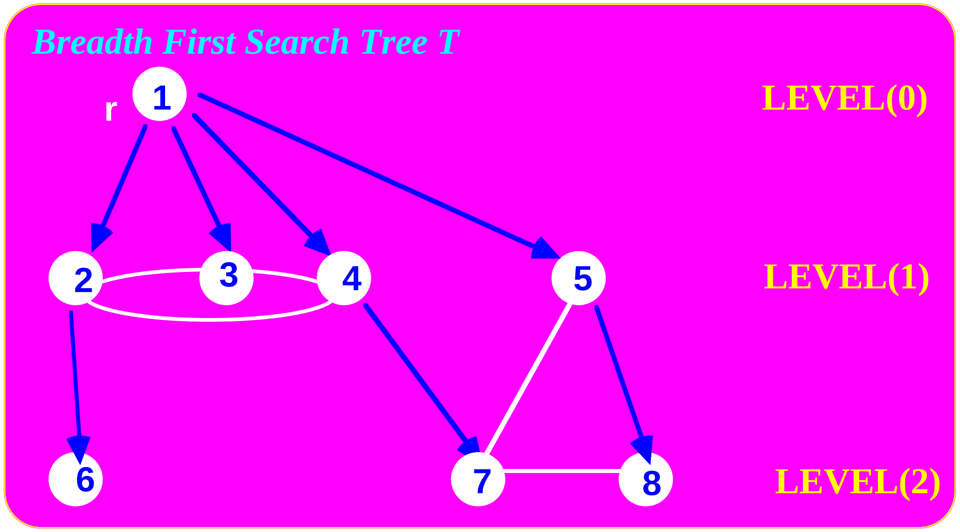
input undirected graph G = (V,E) with root r ε V

```
initialize:   L ← 0
    for each ε V do visit(v) ← false
    LEVEL(0) ← {r} ; visit (r) ← true
    while LEVEL(L) ≠ {} do
        begin
            LEVEL(L+1) ← {}
            for each v ε LEVEL(L) do
                begin
                    for each {v,u} ε E s.t. not visit (u)
                        do
                        add u to LEVEL(L+1)
                        visit (u) ← true
                        od
                end
            L ← L+1
        end
```

output LEVEL(0), LEVEL(1), ..., LEVEL(L - 1)
O(|V|+|E|) time cost

All edges {u,v} ε E have level distance ≤ 1





Single Source Shortest Paths Problem

input

digraph G=(V,E) with root r ε V
weighting d:E → positive reals

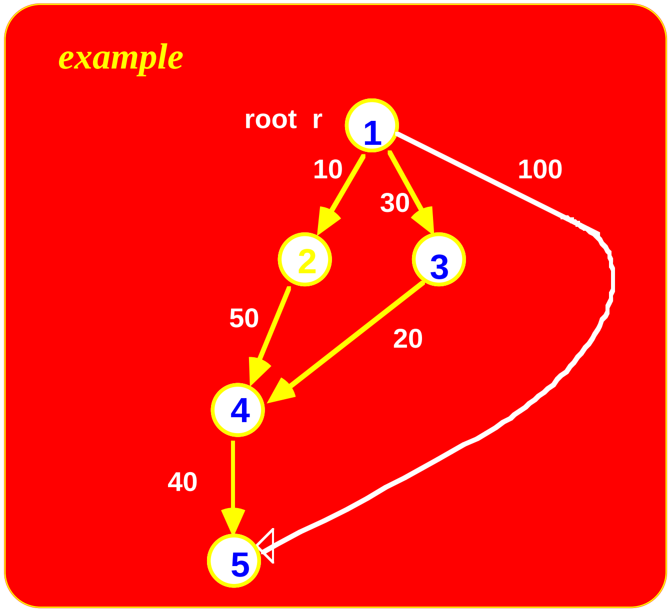
Dijkstra's Greedy algorithm

initialize:

Q ← {}
for each v ε V-{r} *do* D(v) ← ∞
D(r) ← 0
until no change do
choose a vertex u ε V-Q
with minimum D(u)
add u to Q
for each (u,v) ε E s.t. v ε V-Q *do*
D(v) ← min(D(v), D(u) + d(u,v))

output ∀ v ε V

D(v) = weight of min. path from r to v



Q	u	D(1)	D(2)	D(3)	D(4)	D(5)
Φ	1	0	∞	∞	∞	∞
{1}	2	0	10	30	∞	100
{1,2}	3	0	10	30	60	100
{1,2,3}	4	0	10	30	50	100
{1,2,3,4}	5	0	10	30	50	90

proof of Dijkstra's Algorithm

use induction hypothesis:

(1) $\forall$ v ε V,
D(v) is *weight* of the minimum cost of path p from r to v, where p visits only vertices of Q ∪ {v}

(2) $\forall$ v ε Q,
D(v) = minimum cost path from r to v

basis D(r) = 0 for Q={r}

induction step

if D(u) is minimum for all u ε V-Q

then *claim:*

(1) D(u) is minimum cost of path from r to u in G

suppose not: then path p with weight < D(u) and such that p visits a vertex w ε V-(Q ∪ {u}). Then D(w) < D(u) , contradiction.

(2) is satisfied by D(v) = min (D(v),D(u)+d(u,v))
for ∀ v ε Q∪{u} (u,v) ε E

Time Cost: per iteration

- O(log |V|) to find u ε V-Q with min D(u)
- O(degree(u)) to update weights

Since there are |V| iterations,

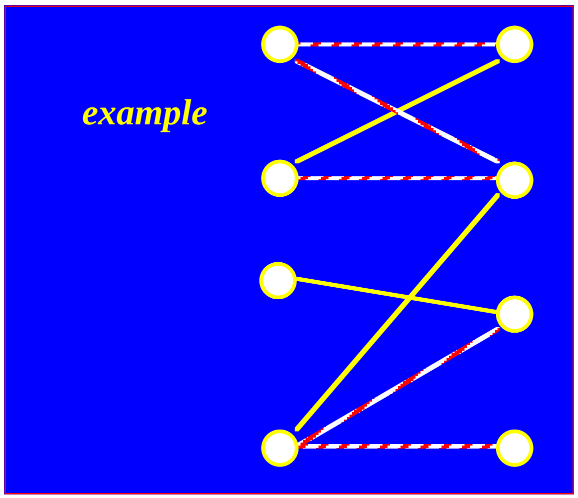
Total Time O(|V|(log |V|) + |E|)

Graph G = (V,E)

matching M is a subset of E satisfies

if e_1, e_2 distinct edges in M

Then they have no vertex in common



Graph Matching Problem:

Find a *maximum* size matching

Let G = (V,E) have matching M

goal: find a larger matching

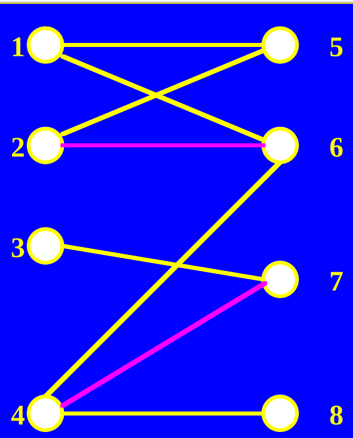
definitions

vertex v is *matched* if
v is in an edge of M

An *augmenting path* $p=(e_1, e_2, \ldots, e_k)$

require
- begins and ends at unmatched vertices
- $e_1, e_3, e_5, \ldots, e_k \in E - M$
- $e_2, e_4, \ldots \quad e_{k-1} \in M$

initial matching M in G

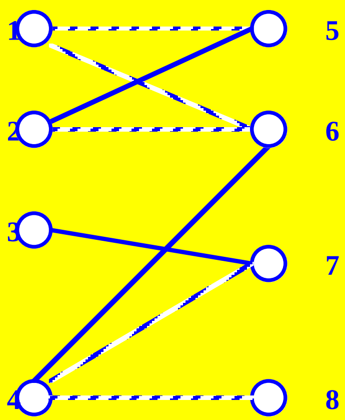


$|M| = 2$

augmenting path

p = ((5,2), (2,6), (6,4), (4,7), (7,3))

new matching $M' = P \oplus M = (P \cup M) - (P \cap M)$



$|P \oplus M| = 3$

Theorem

M is *maximum* matching iff there is *no* augmenting path relative to M

proof **(1) If M is smaller matching and p is an augmenting path for M, then $M \oplus P$ is a matching size > |M|**

(2) If M, M' are matchings with |M| < |M'| then

Claim **$M \oplus M'$ contains an augmenting path for M.**

proof **The graph $G' = (V, M \oplus M')$ has only paths with edges alternating between M and M'.**

Repeatedly delete a cycle in G' (with equal number of edges in M,M')

Since |M|<|M'| must eventually get augmenting path remaining for M.

Algorithm Maximum Matching

input graph G=(V,E)

[1] M ← {}

[2] *while* there exists an augmenting path p relative to M

do M ← M ⊕ P

[3] *output* maximum matching M

Remaining problem:
Find augmenting path

Assume *weighting* $d: E \to R^+$ = pos. reals.

Theorem

Let M be maximum weight among matchings of size |M|. Let p be an augmenting path for M of maximum weight. Then matching M ⊕ P is of maximum weight among matchings of size |M|+1.

proof

Let M' be any maximum weight matching of size |M|+1. Consider the graph G'=(V, M ⊕ M'). Then the maximum weight augmenting path p in G' can be shown to give a matching M ⊕ P of the same weight as M'.

Assume **G is *bipartite graph* with matching M**

Use *Breadth-First Search:*

LEVEL(0) = some unmatched vertex r

for *odd L* > 0,

LEVEL(L) = {u | {v,u} ε E-M
when v ε LEVEL(L-1)
and u in no lower level}

for *even L* > 0

LEVEL(L) = {u | {v,u} ε M
where vε LEVEL(L-1)
and u in no lower level}

Cases

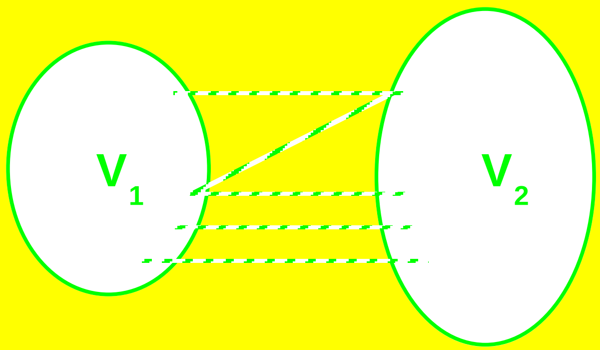
(1) If for some odd L>0,
LEVEL(L) contains an unmatched vertex u
then the Breadth First Search tree T has
an augmenting path from r to u

(2) Otherwise no augmenting path exists, so
M is maximal.

Bipartite Graph G=(V,E)

$V = V_1 \cup V_2$, $V_1 \cap V_2 = \Phi$

E is a subset of { {u,v} | $u \in V_1$, $v \in V_2$ }



Theorem

If G=(V,E) is a bipartite graph, then the maximum matching can be constructed in O(|V||E|) time.

proof

Each stage requires O(|E|) time for time for Breadth First Search construction of augmenting path.

Generalizations:

(1) Compute Edge Weighted Maximum Matching

(2) Edmonds gives a polynomial time algorithm for maximum matching of *any* graph

Let M be matching in general graph G

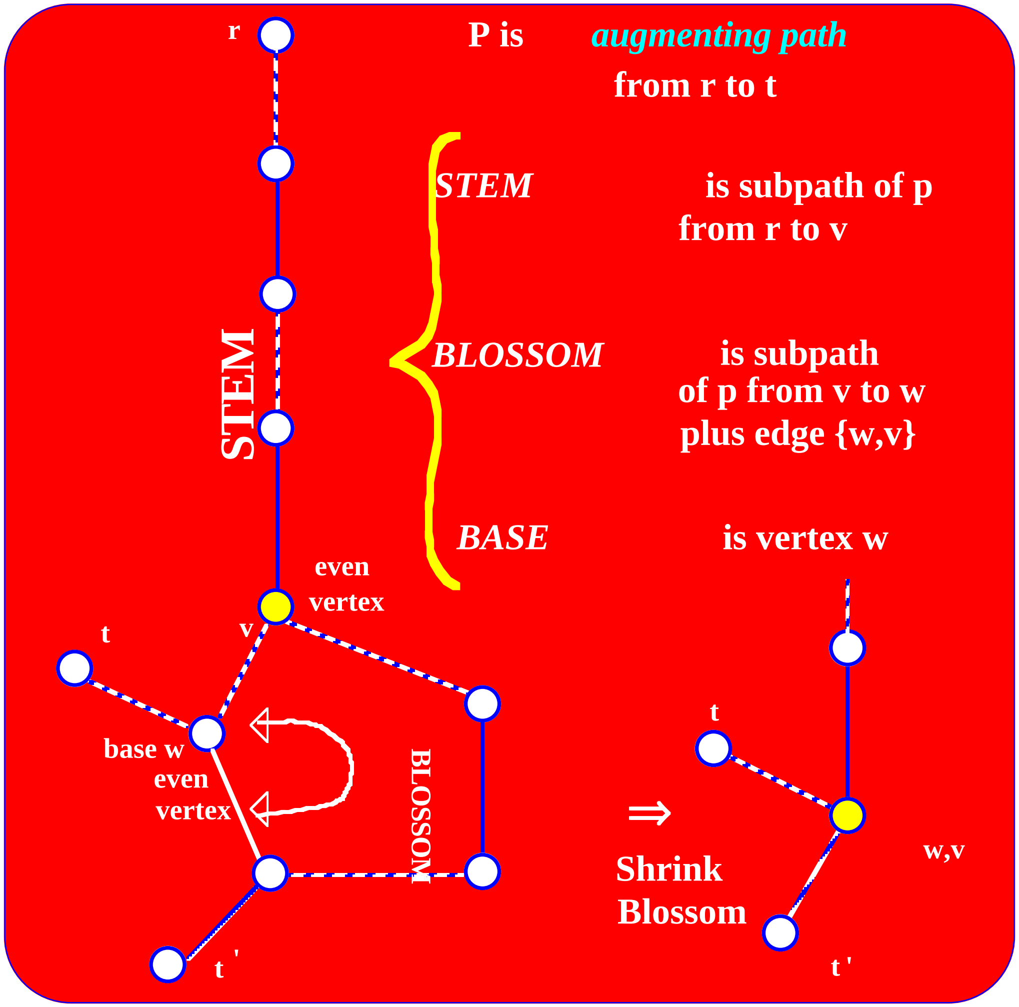
Fix *starting vertex* r
unmatched vertex

Let vertex v ε V be *even* if
∃ even length augmenting path from r to v

and *odd* if
∃ odd length augmenting path from r to v.

Case
G is bipartite
⇒ *no* vertex is both even and odd

Case
G is *not* bipartite
⇒ some vertices may be both even and odd!



Theorem

If G' is formed from G by shrinking of blossom B, then G contains an augmenting path iff G' does.

proof

(1) If ***G' contains an augmenting path*** **p, then if p visits blossom B we can insert an augmenting subpath p' within blossom into p to get a new** ***augmenting path*** **$\hat{p}$ for G**

(2) If ***G contains an augmenting path,*** **then apply Edmond's blossom shrinking algorithm to find an** ***augmenting path in G'.***

Edmond's Blossom Shrinking Algorithm

input **Graph G=(V,E) with matching M**

initialization **$\vec{E}$ = {(v,w),(w,v) | {v,w} ε E}**

comment **Edmond's algorithm will construct a forest of trees whose paths are partial augmenting paths**

Note: **We will let P(v) = parent of vertex v**

[0] ***for*** **each unmatched vertex v ε V**
do **label v as "even"**

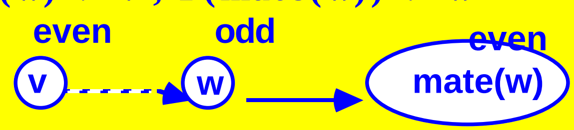
[1] ***for*** **each matched v ε V** ***do***
label v "unreached"
set p(v) = ***null***
if **v is matched to edge {v,w}**
then **mate (v) ← w**
od

Edmond's *Main Loop:*

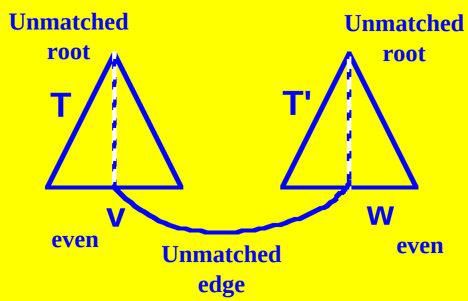
Choose **an unexplored edge (v,w) ε $\vec{E}$**
where v is "even"
(*if* none exists, then terminate and output current matching M, since there is no augmenting path)

Case 1 ***if*** **w is "odd"** ***then*** **do nothing.**

Case 2 ***if*** **w is "unreached" and matched** ***then*** **set w "odd" and set mate (w) "even"**
P(w) ← v , P(mate(w)) ← w



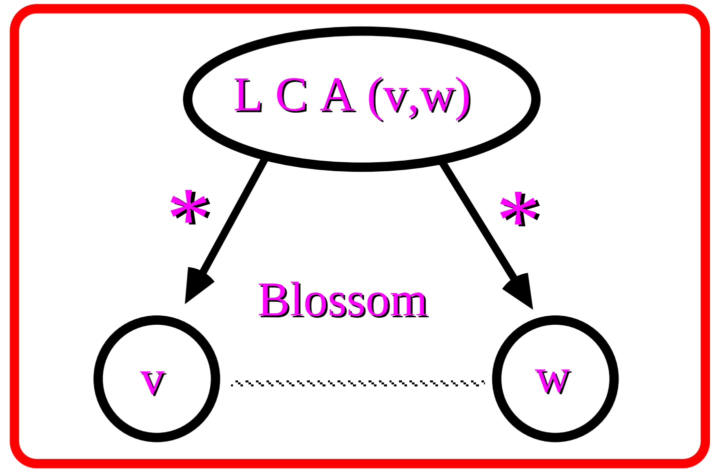
Case 3 ***if*** **w "even" and v,w are in distinct trees T,T'** ***then output*** **augmenting path p from root of T to v, through {v,w}, in T' to root.**



Case 4 **w is "even" and v,w in same tree T then {v,w} forms a blossom B containing all vertices which are**

both (i) a descendant of LCA(v,w) and
(ii) an ancester of v or w

where LCA(v,w) = least common ancester of v,w in T



Shrink **all vertices of B to a single vertex b. Define p(b) = p(LCA(v,w)) and p(x) = b for all x ε B**

Lemma **Edmond's blossom-shrinking algorithm succeeds iff $\exists$ an augmenting path in G**

proof

Uses an induction on blossom shrinking stages

Time Bounds **: $O(n^4)$.**

[1] [Gabow and Tarjan] show

Can implement in *time O(nm)* all O(n) stages of matching algorithms taking O(m) time per stage for blossom shrinking

[2] [Micali and Vazirani] reduce

time to $O(\sqrt{n}\, m)$ for unweighted matching in general graphs.

(Idea: Use *network flow* to get augmented paths).