Algorithm  

**Breadth First Search**

**Input:** undirected graph $G = (V, E)$ with root $r \in V$

**Initialize:**

$L \leftarrow 0$

for each $v \in V$ do

visit($v$) $\leftarrow$ false

LEVEL(0) $\leftarrow \{r\}$; visit (r) $\leftarrow$ true

while LEVEL(L) $\neq \{\}$ do

begin

LEVEL(L+1) $\leftarrow \{\}$

for each $v \in$ LEVEL(L) do

begin

for each $\{v,u\} \in E$ s.t. not visit ($u$) do

add $u$ to LEVEL(L+1)

visit ($u$) $\leftarrow$ true

end

end

$L \leftarrow L+1$

end

**Output:** LEVEL(0), LEVEL(1), ..., LEVEL(L-1)

$O(|V|+|E|)$ time cost
Single Source Shortest Paths Problem

**input**

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

**Dijkstra’s Greedy algorithm**

**initialize:**

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

**output**

∀ v ∈ V  
D(v) = weight of min. path from r to v
proof of Dijkstra's Algorithm

use induction hypothesis:

1. \( \forall v \in V, \) D(v) is weight of the minimum cost path from r to v, where p visits only vertices of \( Q \cup \{v\} \)

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

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

<table>
<thead>
<tr>
<th>Q</th>
<th>u</th>
<th>D(1)</th>
<th>D(2)</th>
<th>D(3)</th>
<th>D(4)</th>
<th>D(5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \Phi )</td>
<td>1</td>
<td>0</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
<td>( \infty )</td>
</tr>
<tr>
<td>{1}</td>
<td>2</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td>( \infty )</td>
<td>100</td>
</tr>
<tr>
<td>{1,2}</td>
<td>3</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td>60</td>
<td>100</td>
</tr>
<tr>
<td>{1,2,3}</td>
<td>4</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td>50</td>
<td>100</td>
</tr>
<tr>
<td>{1,2,3,4}</td>
<td>5</td>
<td>0</td>
<td>10</td>
<td>30</td>
<td>50</td>
<td>90</td>
</tr>
</tbody>
</table>
**Induction step**

if $D(u)$ is minimum for all $u \in V - Q$ 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 $w \in V - (Q \cup \{u\})$. Then $D(w) < D(u)$, contradiction.

2. is satisfied by $D(v) = \min\{D(v), D(u) + d(u,v)\}$ for $\forall v \in Q \cup \{u\}$ \((u,v) \in E\)

**Time Cost:** per iteration

\[
\begin{array}{l}
\quad - O(\log |V|) \text{ to find } u \in V - Q \text{ with min } D(u) \\
\quad - O(\text{degree}(u)) \text{ to update weights}
\end{array}
\]

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

---

**example**

---

**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

\[ e_1, e_3, e_5, \ldots, e_k \in E-M \]

\[ e_2, e_4, \ldots, e_{k-1} \in M \]
Theorem

M is maximum matching if and only if there is no augmenting path relative to M.

Proof

1. If M is a smaller matching and p is an augmenting path for M, then M + p is a matching size $|M| + 1$.

2. If M and M' are matchings with $|M| < |M'|$ then M ⊕ M' contains an augmenting path for M.

Claim

M ⊕ M' contains an augmenting path for M.

Proof

The graph $G' = (V, M ⊕ M')$ 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] $M \leftarrow \{\}$

[2] while there exists an augmenting path $p$ relative to $M$
   do $M \leftarrow M \oplus P$

[3] output maximum matching $M$

Remaining problem:
Find augmenting path

Assume weighting $d:E \rightarrow \mathbb{R}^+ = \text{pos. reals.}$

Theorem
Let $M$ be maximum weight among matchings of size $|M|$. Let $p$ be an augmenting path for $M$ of maximum weight. Then matching $M \oplus P$ 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 $G'=(V, M \oplus M')$. Then the maximum weight augmenting path $p$ in $G'$ can be shown to give a matching $M \oplus P$ 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\} \in E-M \}
 when \( v \in \text{LEVEL}(L-1) \)
 and u in no lower level

for even \( L > 0 \),
 LEVEL(L) = \{ u | \{v,u\} \in M \}
 where \( v \in \text{LEVEL}(L-1) \)
 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 \( G = (V, E) \)

\[ V = V_1 \cup V_2, \quad V_1 \cap V_2 = \emptyset \]

\( E \) 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 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 $v \in V$ be **even** if

\[ \exists \text{ even length augmenting path from } r \text{ to } v \]

and **odd** if

\[ \exists \text{ odd length augmenting path from } r \text{ to } v. \]

**Case**

G is bipartite

\[ \Rightarrow \text{ no vertex is both even and odd} \]

**Case**

G is **not** bipartite

\[ \Rightarrow \text{ some vertices may be both even and odd!} \]
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** $\overline{E} = \{(v,w),(w,v) \mid \{v,w\} \in E\}$

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

**Note:** We will let $P(v) = \text{parent of vertex } v$

\[
\begin{align*}
[0] & \text{ for each unmatched vertex } v \in V \\
& \text{ do } \text{ label } v \text{ as } "\text{even}" \\
[1] & \text{ for each matched } v \in V \text{ do } \\
& \text{ label } v \text{ "unreached" } \\
& \text{ set } P(v) = \text{null } \\
& \text{ if } v \text{ is matched to edge } \{v,w\} \\
& \quad \text{ then } \text{mate}(v) \leftarrow w \\
& \text{ od }
\end{align*}
\]
Edmond's Main Loop:

Choose an unexplored edge \((v, w) \in \bar{E}\) 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"

\[P(w) \leftarrow v, \ P(\text{mate}(w)) \leftarrow w\]

\(v\) even  \(w\) odd  \(\text{mate}(w)\) even

**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 \(\text{LCA}(v, w)\) and (ii) an ancestor of \(v\) or \(w\)

where \(\text{LCA}(v, w) = \text{least common ancestor of} v, w\) in \(T\)

Shrink all vertices of \(B\) to a single vertex \(b\). Define \(p(b) = p(\text{LCA}(v, w))\) and \(p(x) = b\) for all \(x \in B\)
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).