
# Design and Analysis of Algorithms (X)

Simple Unit-Capacity Networks

Guoqiang Li
School of Software

SHANGHAI JIAO TONG UNIVERSITY

# Bipartite Matching

# Matching



**Definition**

Given an undirected graph $G = (V, E)$, subset of edges $M \subseteq E$ is a matching if each node appears in at most one edge in $M$.

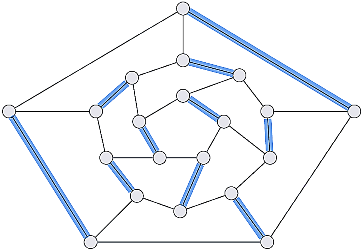
# Matching



**Definition**

Given an undirected graph $G = (V, E)$, subset of edges $M \subseteq E$ is a matching if each node appears in at most one edge in $M$.

Maximum matching. Given a graph $G$, find a max-cardinality matching.



# Bipartite Matching



**Definition**

A graph $G$ is bipartite if the nodes can be partitioned into two subsets $L$ and $R$ such that every edge connects a node in $L$ with a node in $R$.

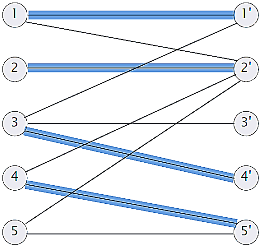
# Bipartite Matching



**Definition**

A graph $G$ is bipartite if the nodes can be partitioned into two subsets $L$ and $R$ such that every edge connects a node in $L$ with a node in $R$.

Bipartite matching. Given a bipartite graph $G = (L \cup R, E)$, find a max-cardinality matching.

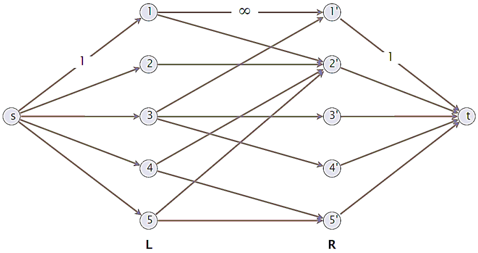


# Max-Flow Formulation

SHANGHAI JIAO TONG UNIVERSITY

Formulation.

- Create digraph $G' = (L \cup R \cup \{s, t\}, E')$.
- Direct all edges from $L$ to $R$, and assign infinite (or unit) capacity.
- Add unit-capacity edges from $s$ to each node in $L$.
- Add unit-capacity edges from each node in $R$ to $t$.



## Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

# Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Rightarrow$

## Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Rightarrow$

- Let $M$ be a matching in $G$ of cardinality $k$.
- Consider flow $f$ that sends 1 unit on each of the $k$ corresponding paths.
- $f$ is a flow of value $k$.

# Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Leftarrow$

# Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Leftarrow$

- Let $f$ be an integral flow in $G'$ of value $k$.
- Consider $M$ = set of edges from $L$ to $R$ with $f(e) = 1$.

## Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Leftarrow$

- Let $f$ be an integral flow in $G'$ of value $k$.
- Consider $M$ = set of edges from $L$ to $R$ with $f(e) = 1$.
  - each node in $L$ and $R$ participates in at most one edge in $M$.
  - $|M| = k$: apply flow-value lemma to cut $(L \cup \{s\}, R \cup \{t\})$.

## Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

## Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

**Corollary**

*Can solve bipartite matching problem via max-flow formulation.*

## Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

**Corollary**

*Can solve bipartite matching problem via max-flow formulation.*

*Proof.*

## Proof of Correctness



**Theorem**

*1-1 correspondence between matchings of cardinality $k$ in $G$ and integral flows of value $k$ in $G'$.*

**Corollary**

*Can solve bipartite matching problem via max-flow formulation.*

*Proof.*

- Integrality theorem $\Rightarrow$ there exists a max flow $f^*$ in $G'$ that is integral.
- 1-1 correspondence $\Rightarrow$ $f^*$ corresponds to max-cardinality matching.

# Quiz 1



What is running time of Ford–Fulkerson algorithms to find a max-cardinality matching in a bipartite graph?

**A.** $O(|E| + |V|)$

**B.** $O(|E||V|)$

**C.** $O(|E||V|^2)$

**D.** $O(|E|^2|V|)$

# Quiz 2



Which max-flow algorithm to use for bipartite matching?

**A.** Ford–Fulkerson: $O(|E| \cdot |V| \cdot C)$.

**B.** Capacity scaling: $O\left(|E|^2 \cdot \log C\right)$.

**C.** Shortest augmenting path: $O\left(|E|^2|V|\right)$.

**D.** Dinitz' algorithm: $O\left(|E||V|^2\right)$.

# Perfect Matchings in Bigraphs



**Definition**

Given a graph $G = (V, E)$, a subset of edges $M \subseteq E$ is a perfect matching if each node appears in exactly one edge in $M$.

# Perfect Matchings in Bigraphs



**Definition**

Given a graph $G = (V, E)$, a subset of edges $M \subseteq E$ is a perfect matching if each node appears in exactly one edge in $M$.

Q. When does a bipartite graph have a perfect matching?

# Perfect Matchings in Bigraphs



**Definition**

Given a graph $G = (V, E)$, a subset of edges $M \subseteq E$ is a perfect matching if each node appears in exactly one edge in $M$.

Q. When does a bipartite graph have a perfect matching?

Structure of bipartite graphs with perfect matchings.

- Clearly, we must have $|L| = |R|$.
- Which other conditions are necessary?
- Which other conditions are sufficient?

# Perfect Matchings in Bigraphs



Notation.

Let $S$ be a subset of nodes, and let $N(S)$ be the set of nodes adjacent to nodes in $S$.

# Perfect Matchings in Bigraphs



Notation.

Let $S$ be a subset of nodes, and let $N(S)$ be the set of nodes adjacent to nodes in $S$.

Observation. If a bipartite graph $G = (L \cup R, E)$ has a perfect matching, then $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

# Perfect Matchings in Bigraphs

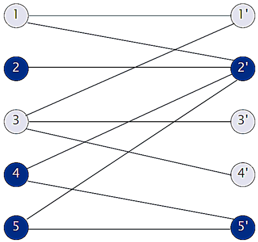


Notation.

Let $S$ be a subset of nodes, and let $N(S)$ be the set of nodes adjacent to nodes in $S$.

Observation. If a bipartite graph $G = (L \cup R, E)$ has a perfect matching, then $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.

*Proof.* Each node in $S$ has to be matched to a different node in $N(S)$.



# Hall's Marriage Theorem



**Theorem (Frobenius 1917, Hall 1935)**

*Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$. Then, graph $G$ has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.*

## Hall's Marriage Theorem

SHANGHAI JIAO TONG UNIVERSITY

**Theorem (Frobenius 1917, Hall 1935)**

*Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$. Then, graph $G$ has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.*

*Proof.* $\Rightarrow$

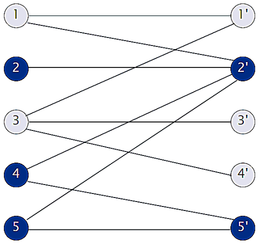
# Hall's Marriage Theorem

**Theorem (Frobenius 1917, Hall 1935)**

*Let $G = (L \cup R, E)$ be a bipartite graph with $|L| = |R|$. Then, graph $G$ has a perfect matching iff $|N(S)| \geq |S|$ for all subsets $S \subseteq L$.*

*Proof.* $\Rightarrow$

This was the previous observation.



# Hall's Marriage Theorem

SHANGHAI JIAO TONG UNIVERSITY

*Proof.* $\Leftarrow$

# Hall's Marriage Theorem

*Proof.* $\Leftarrow$

Suppose $G$ does not have a perfect matching.

# Hall's Marriage Theorem



*Proof.* $\Leftarrow$

Suppose $G$ does not have a perfect matching.

Formulate as a max-flow problem and let $(A, B)$ be a min cut in $G'$.

## Hall's Marriage Theorem

SHANGHAI JIAO TONG UNIVERSITY

*Proof.* $\Leftarrow$

Suppose $G$ does not have a perfect matching.

Formulate as a max-flow problem and let $(A, B)$ be a min cut in $G'$.

By max-flow min-cut theorem, $\mathrm{cap}(A, B) < |L|$.

# Hall's Marriage Theorem



*Proof.* $\Leftarrow$

Suppose $G$ does not have a perfect matching.

Formulate as a max-flow problem and let $(A, B)$ be a min cut in $G'$.

By max-flow min-cut theorem, $\text{cap}(A, B) < |L|$.

Define $L_A = L \cap A, L_B = L \cap B, R_A = R \cap A$.

## Hall's Marriage Theorem

SHANGHAI JIAO TONG UNIVERSITY

*Proof.* $\Leftarrow$

Suppose $G$ does not have a perfect matching.

Formulate as a max-flow problem and let $(A, B)$ be a min cut in $G'$.

By max-flow min-cut theorem, $\mathrm{cap}(A, B) < |L|$.

Define $L_A = L \cap A, L_B = L \cap B, R_A = R \cap A$.

$\mathrm{cap}(A, B) = |L_B| + |R_A| \Rightarrow |R_A| < |L_A|$

Min cut can't use $\infty$ edges $\Rightarrow N(L_A) \subseteq R_A$.

$|N(L_A)| \leq |R_A| < |L_A|$.

Choose $S = L_A$.

# Bipartite Matching



Problem. Given a bipartite graph, find a max-cardinality matching.

| year | worst case | technique | discovered by |
|---|---|---|---|
| 1955 | $O(\lvert E\rvert\lvert V\rvert)$ | augmenting path | Ford–Fulkerson |
| 1973 | $O\left(\lvert E\rvert\lvert V\rvert^{1/2}\right)$ | blocking flow | Hopcroft–Karp, Karzanov |
| 2004 | $O\left(\lvert V\rvert^{2.378}\right)$ | fast matrix multiplication | Mucha–Sankowsi |
| 2013 | $\tilde{O}\left(\lvert E\rvert^{10/7}\right)$ | electrical flow | Madry |
| 20xx | ??? | | |

## Quiz 3



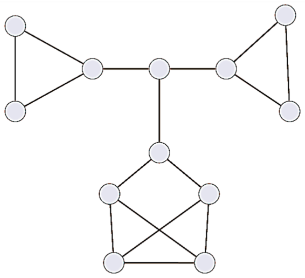
Which of the following are properties of the graph $G = (V, E)$?

**A.** $G$ has a perfect matching.

**B.** Hall's condition is satisfied: $|N(S)| \geq |S|$ for all subsets $S \subseteq V$.

**C.** Both A and B.

**D.** Neither A nor B.



# Nonbipartite Matching

Shanghai Jiao Tong University

Problem. Given an undirected graph, find a max-cardinality matching.

# Nonbipartite Matching



Problem. Given an undirected graph, find a max-cardinality matching.

- Structure of nonbipartite graphs is more complicated.
- But well understood. [Tutte–Berge formula, Edmonds–Gallai]
- Blossom algorithm: $O(n^4)$. [Edmonds 1965]
- Best known: $O(mn^{1/2})$. [Micali–Vazirani 1980, Vazirani 1994]

# Hackathon Problem



Hackathon problem.

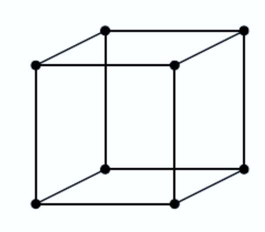
- Hackathon attended by $n$ Harvard students and $n$ Princeton students.
- Each Harvard student is friends with exactly $k > 0$ Princeton students; each Princeton student is friends with exactly $k$ Harvard students.
- Is it possible to arrange the hackathon so that each Princeton student pair programs with a different friend from Harvard?

## Hackathon Problem



Mathematical reformulation. Does every $k$-regular bipartite graph have a perfect matching?

Example. Boolean hypercube.



# Hackathon Problem



**Theorem**

*Every $k$-regular bipartite graph $G$ has a perfect matching.*

# Hackathon Problem



**Theorem**

*Every $k$-regular bipartite graph $G$ has a perfect matching.*

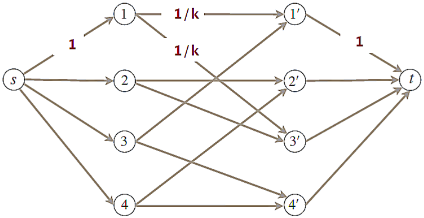
Proved by Hall's Marriage Theorem, DIY!

## Hackathon Problem: Another Proof

*Proof.*

- Size of max matching $=$ value of max flow in $G'$.
- It is easy to construct the following flow

$$f(u,v) = \begin{cases} 1 & \text{if } u = s \text{ or } v = t \\ 1/k & \text{otherwise} \end{cases}$$




- The value of flow $f$ is $n \Rightarrow G'$ has a perfect matching.

# Hall's Theorem by Max-Flow

## Simple Unit-Capacity Networks

# Simple Unit-Capacity Networks

SHANGHAI JIAO TONG UNIVERSITY

**Definition**

A flow network is a simple unit-capacity network if:

- Every edge has capacity 1.
- Every node (other than $s$ or $t$) has exactly one entering edge, or exactly one leaving edge, or both.

# Simple Unit-Capacity Networks



Property. Let $G$ be a simple unit-capacity network and let $f$ be a 0–1 flow. Then, residual network $G_f$ is also a simple unit-capacity network.

# Simple Unit-Capacity Networks



Property. Let $G$ be a simple unit-capacity network and let $f$ be a 0–1 flow. Then, residual network $G_f$ is also a simple unit-capacity network.

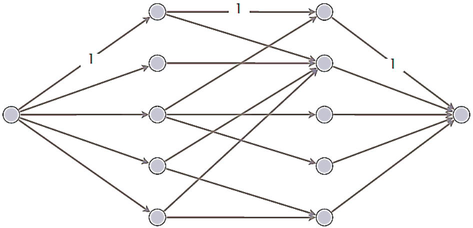
Example. Bipartite matching.

# Simple Unit-Capacity Networks



Property. Let $G$ be a simple unit-capacity network and let $f$ be a 0–1 flow. Then, residual network $G_f$ is also a simple unit-capacity network.

Example. Bipartite matching.



## Simple Unit-Capacity Networks



Shortest-augmenting-path algorithm.

- Normal augmentation: length of shortest path does not change.
- Special augmentation: length of shortest path strictly increases.

**Theorem (Even–Tarjan 1975)**

*In simple unit-capacity networks, Dinitz'algorithm computes a maximum flow in $O(|E||V|^{1/2})$ time.*

# Simple Unit-Capacity Networks



Shortest-augmenting-path algorithm.

- Normal augmentation: length of shortest path does not change.
- Special augmentation: length of shortest path strictly increases.

**Theorem (Even–Tarjan 1975)**

*In simple unit-capacity networks, Dinitz'algorithm computes a maximum flow in* $O(|E||V|^{1/2})$ *time.*

*Proof.*

# Simple Unit-Capacity Networks

Shortest-augmenting-path algorithm.

- Normal augmentation: length of shortest path does not change.
- Special augmentation: length of shortest path strictly increases.

**Theorem (Even–Tarjan 1975)**

*In simple unit-capacity networks, Dinitz'algorithm computes a maximum flow in $O(|E||V|^{1/2})$ time.*

*Proof.*

- Lemma 1. Each phase of normal augmentations takes $O(|E|)$ time.
- Lemma 2. After $|V|^{1/2}$ phases, $\mathrm{val}(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$.
- Lemma 3. After $\leq |V|^{1/2}$ additional augmentations, flow is optimal.

# Simple Unit-Capacity Networks

**Lemma 3**

After $\leq |V|^{1/2}$ additional augmentations, flow is optimal.

## Simple Unit-Capacity Networks



**Lemma 3**

After $\leq |V|^{1/2}$ additional augmentations, flow is optimal.

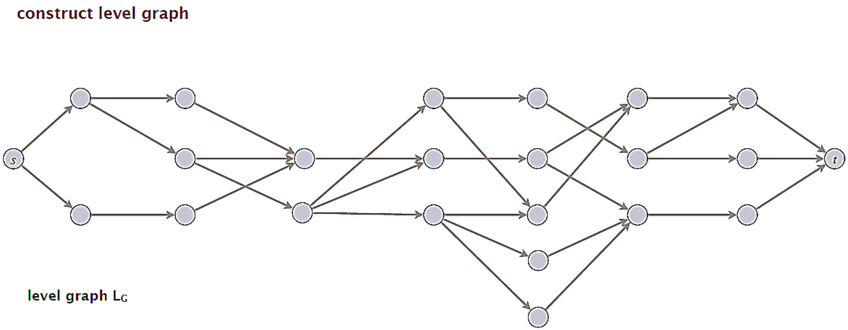
*Proof.* Each augmentation increases flow value by at least 1.

## Simple Unit-Capacity Networks



Phase of normal augmentations.

- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.

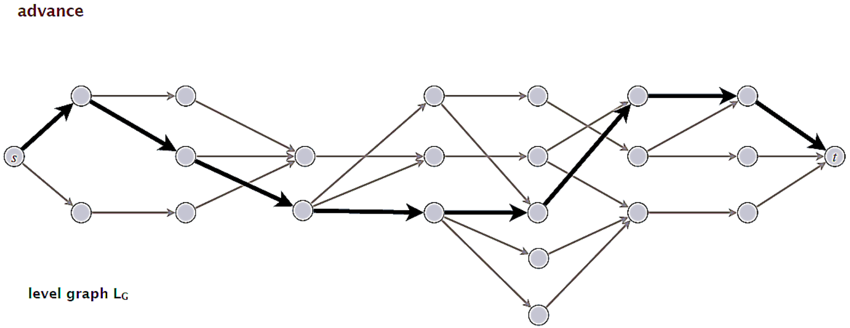


# Simple Unit-Capacity Networks



Phase of normal augmentations.

- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.

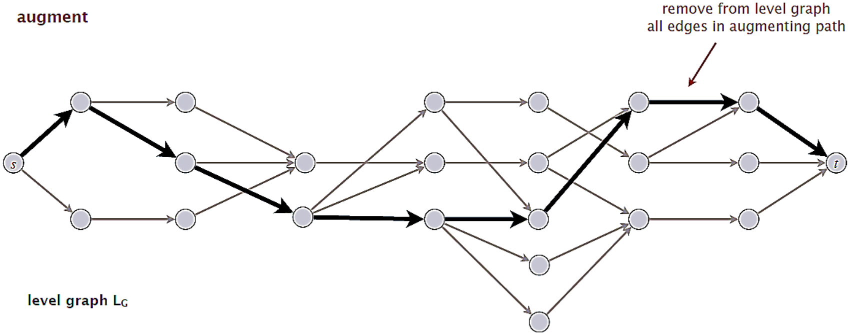


# Simple Unit-Capacity Networks



Phase of normal augmentations.

- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.

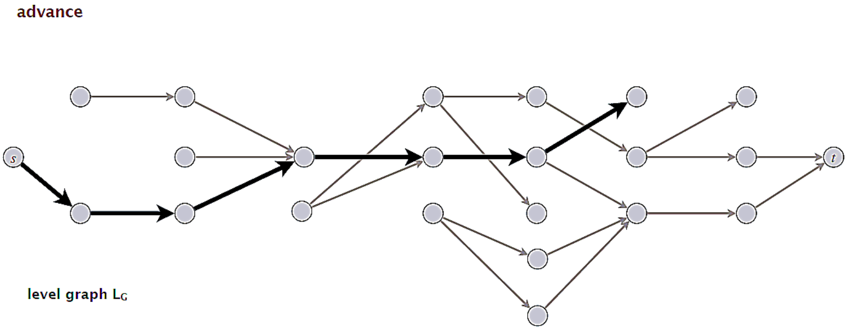


# Simple Unit-Capacity Networks

SHANGHAI JIAO TONG UNIVERSITY

Phase of normal augmentations.

- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.

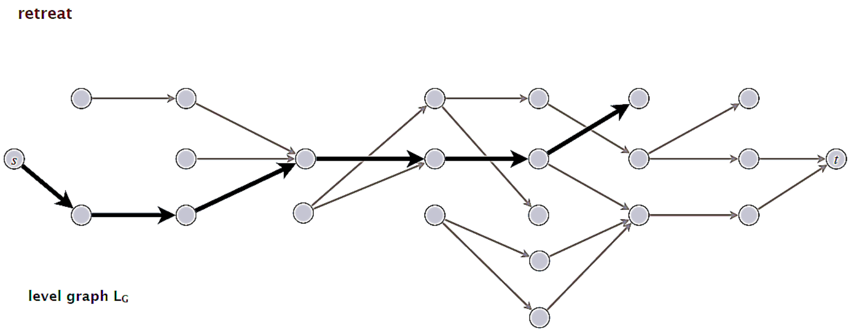


## Simple Unit-Capacity Networks



Phase of normal augmentations.

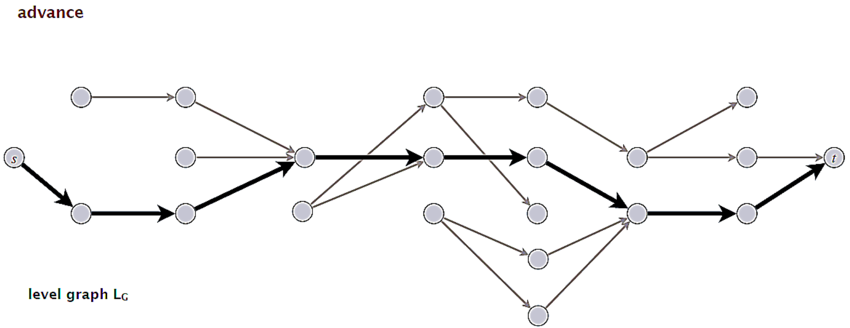
- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.



# Simple Unit-Capacity Networks

Phase of normal augmentations.

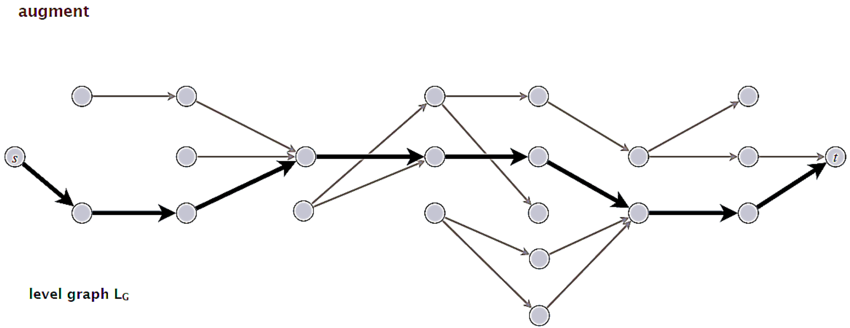
- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.



# Simple Unit-Capacity Networks

Phase of normal augmentations.

- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.

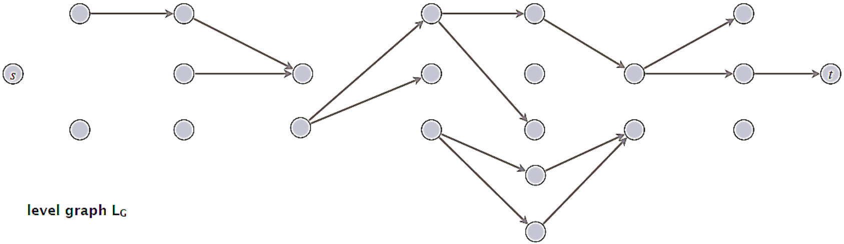


# Simple Unit-Capacity Networks

Shanghai Jiao Tong University

Phase of normal augmentations.

- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.




level graph $L_G$

## Simple Unit-Capacity Networks



Phase of normal augmentations.

- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.

## Simple Unit-Capacity Networks



Phase of normal augmentations.

- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.

**Lemma 1**

A phase of normal augmentations takes $O(|E|)$ time.

## Simple Unit-Capacity Networks



Phase of normal augmentations.

- Construct level graph $L_G$.
- Start at $s$, advance along an edge in $L_G$ until reach $t$ or get stuck.
- If reach $t$, augment flow; update $L_G$; and restart from $s$.
- If get stuck, delete node from $L_G$ and go to previous node.

**Lemma 1**

A phase of normal augmentations takes $O(|E|)$ time.

*Proof.*

- $O(|E|)$ to create level graph $L_G$.
- $O(1)$ per edge (each edge involved in at most one advance, retreat, and augmentation).
- $O(1)$ per node (each node deleted at most once)

**Lemma 2**

After $|V|^{1/2}$ phases, $\mathrm{val}(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$.

**Lemma 2**

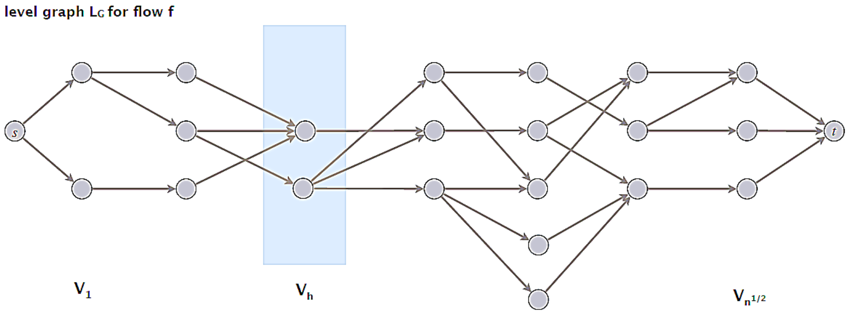
After $|V|^{1/2}$ phases, $\mathrm{val}(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$.

*Proof.*

**Lemma 2**

After $|V|^{1/2}$ phases, $\text{val}(f) \geq \text{val}(f^*) - |V|^{1/2}$.

*Proof.*



**Lemma 2**

After $|V|^{1/2}$ phases, $\mathrm{val}(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$.

*Proof.*

After $|V|^{1/2}$ phases, length of shortest augmenting path is $> |V|^{1/2}$.

**Lemma 2**

After $|V|^{1/2}$ phases, $\mathrm{val}(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$.

*Proof.*

After $|V|^{1/2}$ phases, length of shortest augmenting path is $> |V|^{1/2}$.

Thus, level graph has $\geq |V|^{1/2}$ levels (not including levels for $s$ or $t$)

**Lemma 2**

After $|V|^{1/2}$ phases, $\mathrm{val}(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$.

*Proof.*

After $|V|^{1/2}$ phases, length of shortest augmenting path is $> |V|^{1/2}$.

Thus, level graph has $\geq |V|^{1/2}$ levels (not including levels for $s$ or $t$)

Let $1 \leq h \leq |V|^{1/2}$ be a level with min number of nodes $\Rightarrow |V_h| \leq |V|^{1/2}$.

**Lemma 2**

After $|V|^{1/2}$ phases, $\mathrm{val}(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$.

*Proof.*

After $|V|^{1/2}$ phases, length of shortest augmenting path is $> |V|^{1/2}$.

Thus, level graph has $\geq |V|^{1/2}$ levels (not including levels for $s$ or $t$)

Let $1 \leq h \leq |V|^{1/2}$ be a level with min number of nodes $\Rightarrow |V_h| \leq |V|^{1/2}$.

Let $A = \{v : \ell(v) < h\} \cup \{v : \ell(v) = h \text{ and } v \text{ has } \leq 1 \text{ outgoing residual edge}\}$ .

**Lemma 2**

After $|V|^{1/2}$ phases, $\mathrm{val}(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$.

*Proof.*

After $|V|^{1/2}$ phases, length of shortest augmenting path is $> |V|^{1/2}$.

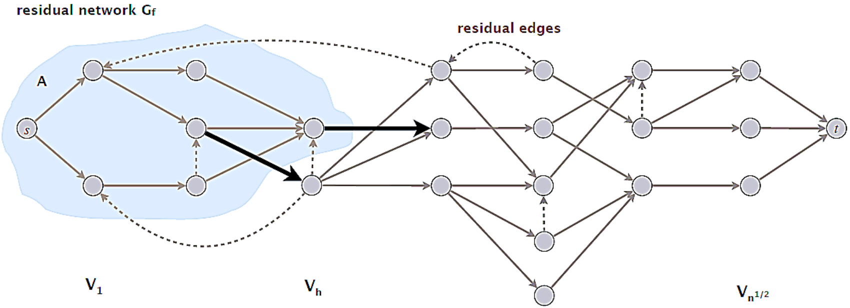
Thus, level graph has $\geq |V|^{1/2}$ levels (not including levels for $s$ or $t$)

Let $1 \leq h \leq |V|^{1/2}$ be a level with min number of nodes $\Rightarrow |V_h| \leq |V|^{1/2}$.

Let $A = \{v : \ell(v) < h\} \cup \{v : \ell(v) = h \text{ and } v \text{ has } \leq 1 \text{ outgoing residual edge}\}$ .

$\mathrm{cap}_f(A, B) \leq |V_h| \leq |V|^{1/2} \Rightarrow val(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$

# Computational Geometry



## Review



**Theorem (Even–Tarjan 1975)**

*In simple unit-capacity networks, Dinitz' algorithm computes a maximum flow in $O(|E||V|^{1/2})$ time.*

*Proof.*

- Lemma 1. Each phase take $O(|E|)$ time.
- Lemma 2. After $|V|^{1/2}$ phase, $\mathrm{val}(f) \geq \mathrm{val}\left(f^*\right) - |V|^{1/2}$
- Lemma 3. After $\leq |V|^{1/2}$ additional augmentations.

## Review



**Theorem (Even–Tarjan 1975)**

*In simple unit-capacity networks, Dinitz' algorithm computes a maximum flow in $O(|E||V|^{1/2})$ time.*

*Proof.*

- Lemma 1. Each phase take $O(|E|)$ time.
- Lemma 2. After $|V|^{1/2}$ phase, $\mathrm{val}(f) \geq \mathrm{val}(f^*) - |V|^{1/2}$
- Lemma 3. After $\leq |V|^{1/2}$ additional augmentations.

**Corollary**

*Dinitz' algorithm computes maximum-cardinality bipartite matching in $O(|E||V|^{1/2})$ time.*

# Disjoint Paths

## Edge-Disjoint Paths



**Definition**

Two paths are edge-disjoint if they have no edge in common.

# Edge-Disjoint Paths



**Definition**

Two paths are edge-disjoint if they have no edge in common.

Edge-disjoint paths problem. Given a digraph $G = (V, E)$ and two nodes $s$ and $t$, find the max number of edge-disjoint $s \rightsquigarrow t$ paths.

**Definition**

Two paths are edge-disjoint if they have no edge in common.

Edge-disjoint paths problem. Given a digraph $G = (V, E)$ and two nodes $s$ and $t$, find the max number of edge-disjoint $s \rightsquigarrow t$ paths.

Max-flow formulation. Assign unit capacity to every edge.

## Edge-Disjoint Paths



**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \leadsto t$ paths in $G$ and integral flows of value $k$ in $G'$.*

# Edge-Disjoint Paths

SHANGHAI JIAO TONG UNIVERSITY

**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Rightarrow$

**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Rightarrow$

- Let $P_1, \ldots, P_k$ be $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$.

## Edge-Disjoint Paths



**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Rightarrow$

- Let $P_1, \ldots, P_k$ be $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$.
- Set $f(e) = \begin{cases} 1 & \text{edge } e \text{ participates in some path } P_j \\ 0 & \text{otherwise} \end{cases}$

## Edge-Disjoint Paths



**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Rightarrow$

- Let $P_1, \ldots, P_k$ be $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$.
- Set $f(e) = \begin{cases} 1 & \text{edge } e \text{ participates in some path } P_j \\ 0 & \text{otherwise} \end{cases}$
- Since paths are edge-disjoint, $f$ is a flow of value $k$.

## Edge-Disjoint Paths



**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Leftarrow$

# Edge-Disjoint Paths

SHANGHAI JIAO TONG UNIVERSITY

**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

*Proof.* $\Leftarrow$

- Let $f$ be an integral flow in $G'$ of value $k$.
- Consider edge $(s, u)$ with $f(s, u) = 1$.
  - by flow conservation, there exists an edge $(u, v)$ with $f(u, v) = 1$.
  - continue until reach $t$, always choosing a new edge
- Produces $k$ (not necessarily simple) edge-disjoint paths.

## Edge-Disjoint Paths



**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

# Edge-Disjoint Paths

SHANGHAI JIAO TONG UNIVERSITY

**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

**Corollary**

*Can solve edge-disjoint paths problem via max-flow formulation.*

## Edge-Disjoint Paths



**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

**Corollary**

*Can solve edge-disjoint paths problem via max-flow formulation.*

*Proof.*

# Edge-Disjoint Paths



**Theorem**

*1-1 correspondence between $k$ edge-disjoint $s \rightsquigarrow t$ paths in $G$ and integral flows of value $k$ in $G'$.*

**Corollary**

*Can solve edge-disjoint paths problem via max-flow formulation.*

*Proof.*

- Integrality theorem $\Rightarrow$ there exists a max flow $f^*$ in $G'$ that is integral.
- 1-1 correspondence $\Rightarrow$ $f^*$ corresponds to max number of edge-disjoint $s \rightsquigarrow t$ paths in $G$.

## Network Connectivity



**Definition**

A set of edges $F \subseteq E$ disconnects $t$ from $s$ if every $s \rightsquigarrow t$ path uses at least one edge in $F$.

## Network Connectivity

**Definition**

A set of edges $F \subseteq E$ disconnects $t$ from $s$ if every $s \rightsquigarrow t$ path uses at least one edge in $F$.

Network connectivity. Given a digraph $G = (V, E)$ and two nodes $s$ and $t$, find minimal number of edges whose removal disconnects $t$ from $s$.

## Menger's Theorem



**Theorem (Menger 1927)**

*The max number of edge-disjoint $s \rightsquigarrow t$ paths equals the min number of edges whose removal disconnects $t$ from $s$.*

# Menger's Theorem

**Theorem (Menger 1927)**

*The max number of edge-disjoint $s \rightsquigarrow t$ paths equals the min number of edges whose removal disconnects $t$ from $s$.*

*Proof.* $\leq$

## Menger's Theorem



**Theorem (Menger 1927)**

*The max number of edge-disjoint $s \rightsquigarrow t$ paths equals the min number of edges whose removal disconnects $t$ from $s$.*

*Proof.* $\leq$

- Suppose the removal of $F \subseteq E$ disconnects $t$ from $s$, and $|F| = k$.
- Every $s \rightsquigarrow t$ path uses at least one edge in $F$.
- Hence, the number of edge-disjoint paths is $\leq k$.

## Menger's Theorem



**Theorem (Menger 1927)**

*The max number of edge-disjoint $s \rightsquigarrow t$ paths equals the min number of edges whose removal disconnects $t$ from $s$.*

*Proof.* $\geq$

## Menger's Theorem



**Theorem (Menger 1927)**

*The max number of edge-disjoint $s \rightsquigarrow t$ paths equals the min number of edges whose removal disconnects $t$ from $s$.*

*Proof.* $\geq$

- Suppose max number of edge-disjoint $s \rightsquigarrow t$ paths is $k$.
- Then value of max flow = $k$.
- Max-flow min-cut theorem $\Rightarrow$ there exists a cut $(A, B)$ of capacity $k$.
- Let $F$ be set of edges going from $A$ to $B$.
- $|F| = k$ and disconnects $t$ from $s$.

# Referred Materials

## Referred Materials

- Content of this lecture comes from Section 7.5 in [KT05].