Lecture 14

Tuesday, July 4, 2017 8:38 PM

Running time

Assumption. Capacities are integers between 1 and C.

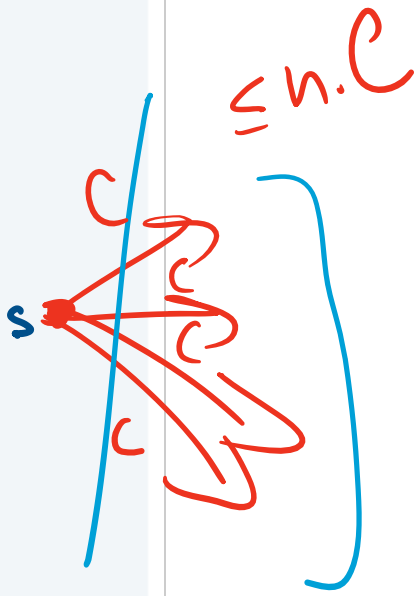
Integrality invariant. Throughout the algorithm, the flow values $f(e)$ and the residual capacities $c_f(e)$ are integers.

Theorem. The algorithm terminates in at most $val(f^*) \leq nC$ iterations.
Pf. Each augmentation increases the value by at least 1. ▪

Corollary. The running time of Ford-Fulkerson is $O(mnC)$.
Corollary. If $C = 1$, the running time of Ford-Fulkerson is $O(mn)$.

Integrality theorem. Then exists a max-flow f^* for which every flow value $f^*(e)$ is an integer.
Pf. Since algorithm terminates, theorem follows from invariant. ▪

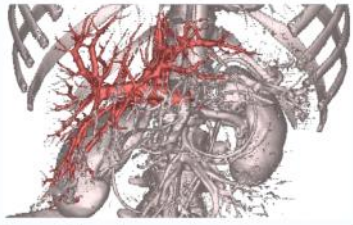
37



Max-flow and min-cut applications

Max-flow and min-cut are widely applicable problem-solving model.

- Data mining.
- Open-pit mining.
- Bipartite matching.
- Network reliability.
- Baseball elimination.
- Image segmentation.
- Network connectivity.
- Distributed computing.
- Security of statistical data.
- Egalitarian stable matching.
- Network intrusion detection.
- Multi-camera scene reconstruction.
- Sensor placement for homeland security.
- Many, many, more.



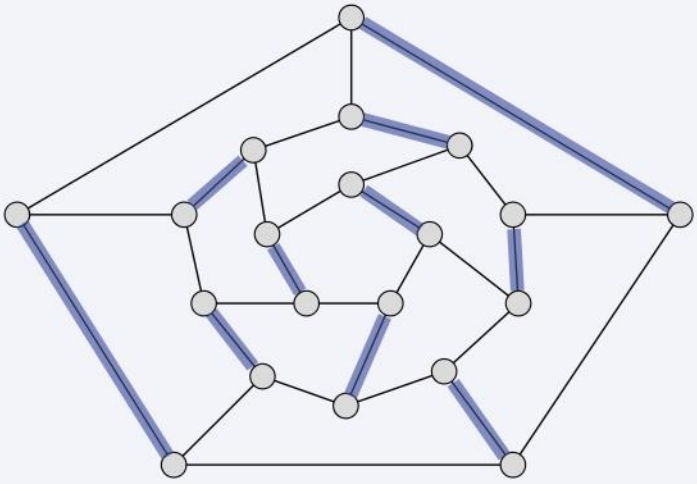
liver and hepatic vascularization segmentation

3

Matching

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

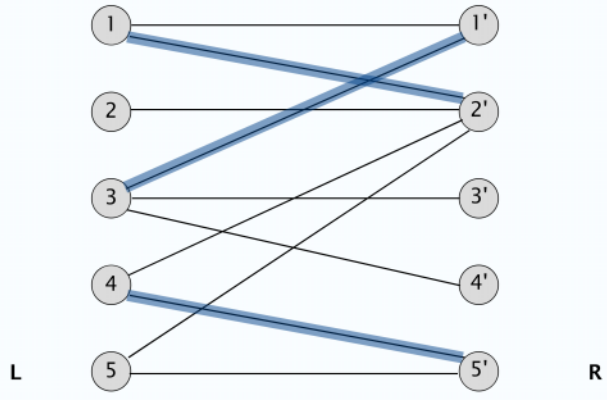
Max matching. Given a graph, find a max cardinality matching.



Bipartite matching

Def. 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 to one in R.

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

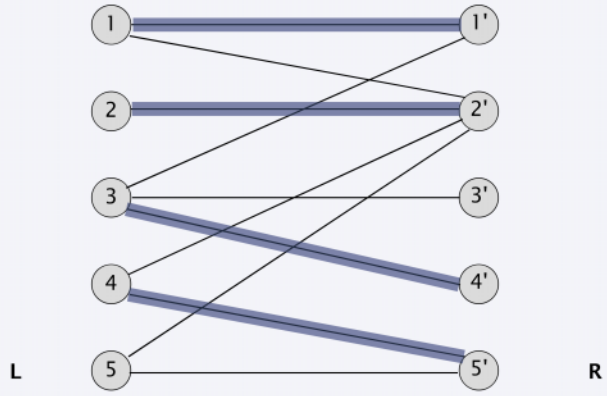


matching: 1-2', 3-1', 4-5'

Bipartite matching

Def. 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 to one in R.

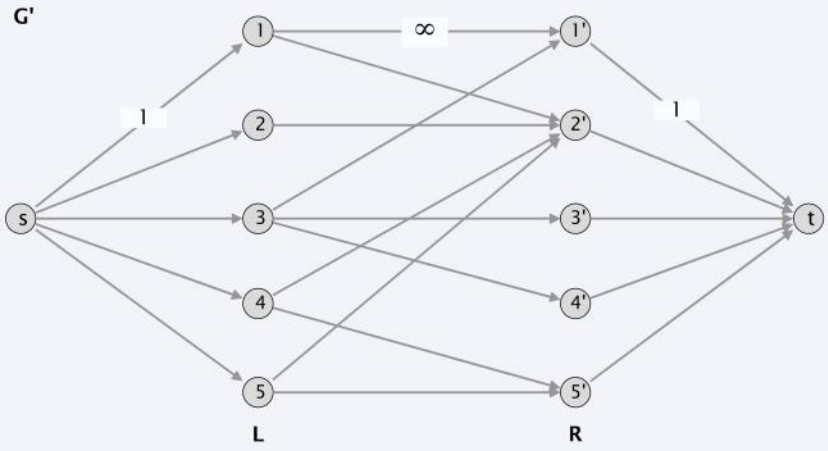
Bipartite matching. Given a bipartite graph $G = (L \cup R, E)$, find a max cardinality matching.



matching: 1-1', 2-2', 3-4', 4-5'

Bipartite matching: max flow 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 source s, and unit capacity edges from s to each node in L.
- Add sink t, and unit capacity edges from each node in R to t.

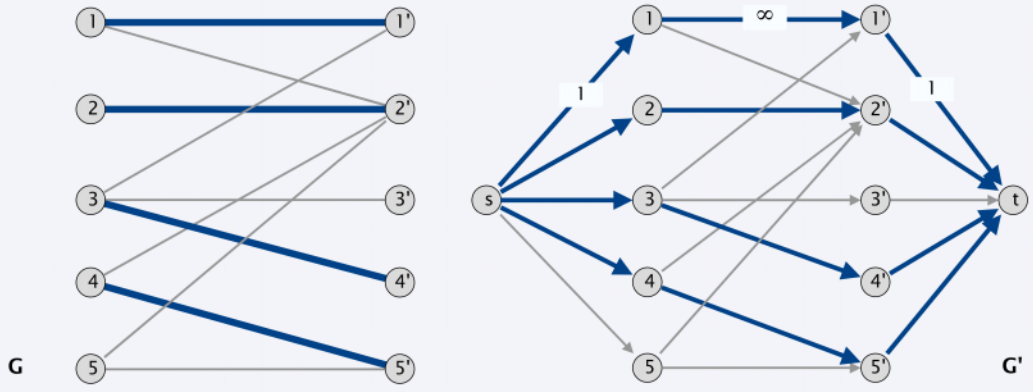


Max flow formulation: proof of correctness

Theorem. Max cardinality of a matching in G = value of max flow in G'.

Pf. ≤

- Given a max matching M of cardinality k.
- Consider flow f that sends 1 unit along each of k paths.
- f is a flow, and has value k. ▪



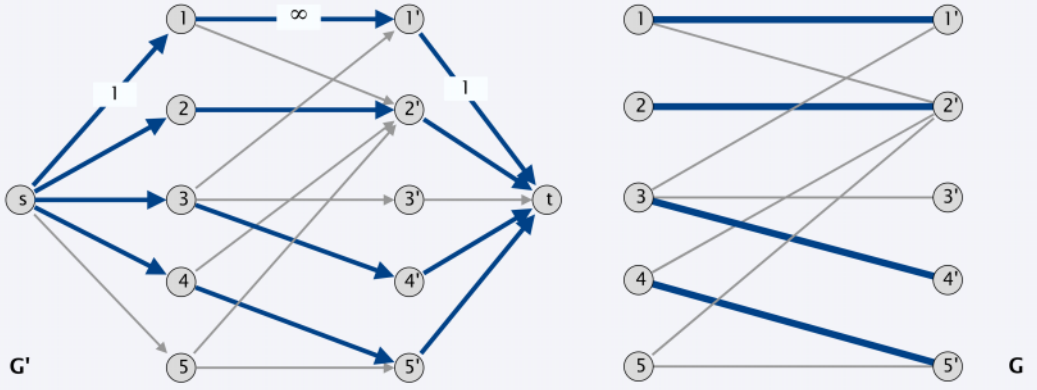
9

Max flow formulation: proof of correctness

Theorem. Max cardinality of a matching in G = value of max flow in G'.

Pf. ≥

- Let f be a max flow in G' of value k.
- Integrality theorem ⇒ k is integral and can assume f is 0-1.
- 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
  - $|M| = k$: consider cut $(L \cup s, R \cup t)$ ▪



10

Perfect matching in a bipartite graph

Def. 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 $|L| = |R|$.
- What other conditions are necessary?
- What conditions are sufficient?

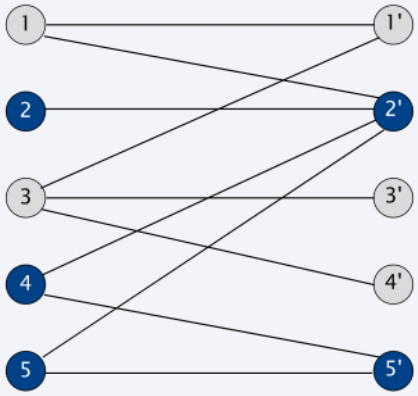
Perfect matching in a bipartite graph

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 $S \subseteq L$.

Pf. Each node in S has to be matched to a different node in $N(S)$. ▪

S = { 2, 4, 5 }
N(S) = { 2', 5' }



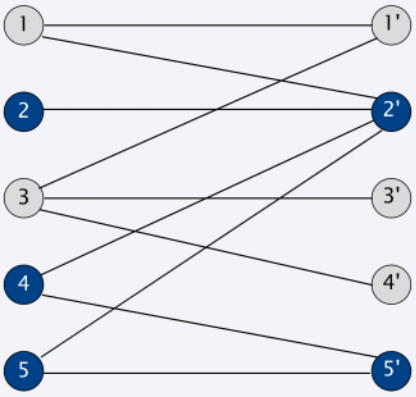
no perfect matching

Hall's theorem

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

Pf. $\Rightarrow$ This was the previous observation.

S = { 2, 4, 5 }
N(S) = { 2', 5' }

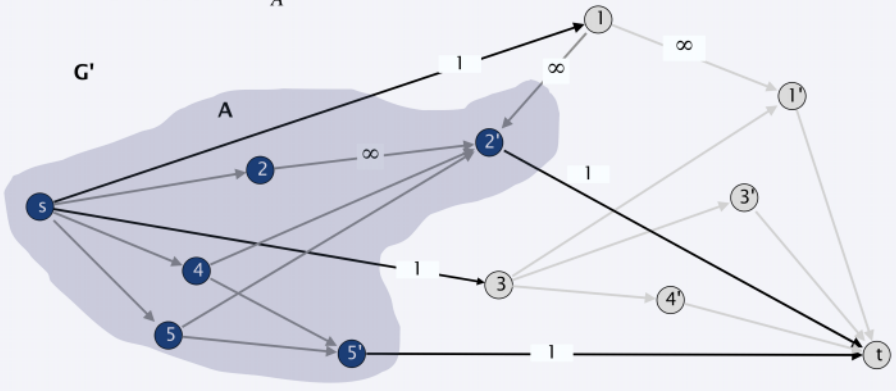


no perfect matching

Proof of Hall's theorem

Pf. $\Leftarrow$ Suppose G does not have a perfect matching.

- Formulate as a max flow problem and let (A, B) be min cut in G'.
- By max-flow min-cut theorem, $cap(A, B) < |L|$.
- Define $L_A = L \cap A$, $L_B = L \cap B$, $R_A = R \cap A$.
- $cap(A, B) = |L_B| + |R_A|$.
- Since min cut can't use ∞ edges: $N(L_A) \subseteq R_A$.
- $|N(L_A)| \leq |R_A| = cap(A, B) - |L_B| < |L| - |L_B| = |L_A|$.
- Choose $S = L_A$. ▪

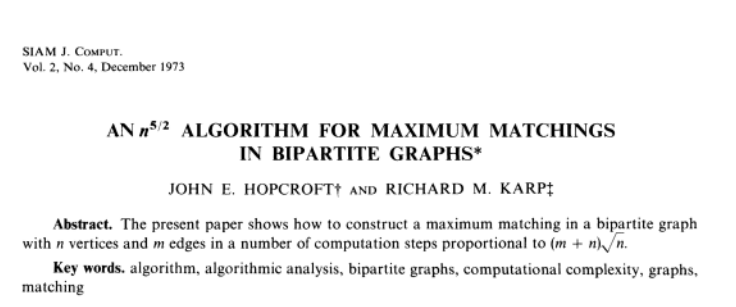


L_A = {2, 4, 5}
L_B = {1, 3}
R_A = {2', 5'}
$N(L_A)$ = {2', 5'}

Bipartite matching running time

Theorem. The Ford-Fulkerson algorithm solves the bipartite matching problem in $O(m\,n)$ time.

Theorem. [Hopcroft-Karp 1973] The bipartite matching problem can be solved in $O(m\,n^{1/2})$ time.



SIAM J. COMPUT.
Vol. 2, No. 4, December 1973

AN $n^{5/2}$ ALGORITHM FOR MAXIMUM MATCHINGS IN BIPARTITE GRAPHS*

JOHN E. HOPCROFT† AND RICHARD M. KARP‡

Abstract. The present paper shows how to construct a maximum matching in a bipartite graph with n vertices and m edges in a number of computation steps proportional to $(m + n)\sqrt{n}$.

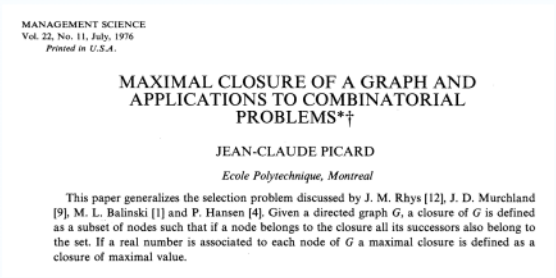
Key words. algorithm, algorithmic analysis, bipartite graphs, computational complexity, graphs, matching

Project selection (maximum weight closure problem)

Projects with prerequisites.

- Set of possible projects P: project v has associated revenue p_v. (can be positive or negative)
- Set of prerequisites E: if $(v, w) \in E$, cannot do project v unless also do project w.
- A subset of projects $A \subseteq P$ is feasible if the prerequisite of every project in A also belongs to A.

Project selection problem. Given a set of projects P and prerequisites E, choose a feasible subset of projects to maximize revenue.



MANAGEMENT SCIENCE
Vol. 22, No. 11, July, 1976
Printed in U.S.A.

MAXIMAL CLOSURE OF A GRAPH AND APPLICATIONS TO COMBINATORIAL PROBLEMS*†

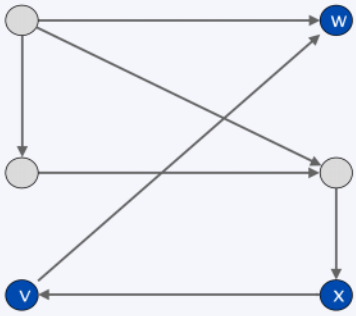
JEAN-CLAUDE PICARD

Ecole Polytechnique, Montreal

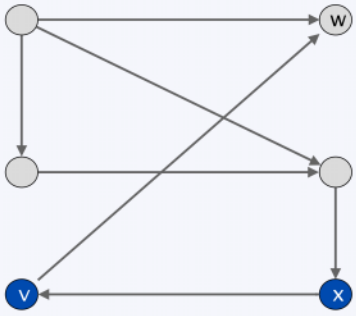
This paper generalizes the selection problem discussed by J. M. Rhys [12], J. D. Murchland [9], M. L. Balinski [1] and P. Hansen [4]. Given a directed graph G, a closure of G is defined as a subset of nodes such that if a node belongs to the closure all its successors also belong to the set. If a real number is associated to each node of G a maximal closure is defined as a closure of maximal value.

Project selection: prerequisite graph

Prerequisite graph. Add edge (v, w) if can't do v without also doing w.



{ v, w, x } is feasible

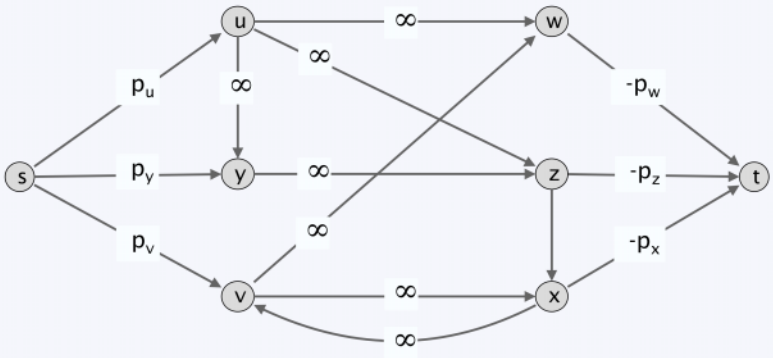


{ v, x } is infeasible

Project selection: min-cut formulation

Min-cut formulation.

- Assign capacity ∞ to all prerequisite edge.
- Add edge (s, v) with capacity p_v if $p_v > 0$.
- Add edge (v, t) with capacity $-p_v$ if $p_v < 0$.
- For notational convenience, define $p_s = p_t = 0$.

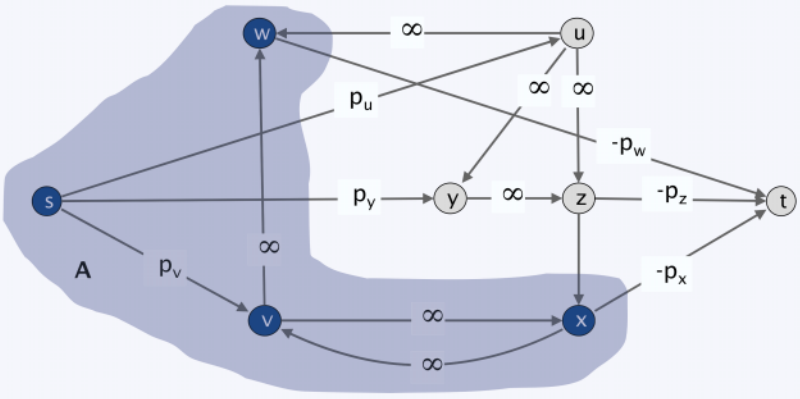


Project selection: min-cut formulation

Claim. (A, B) is min cut iff $A - \{ s \}$ is optimal set of projects.

- Infinite capacity edges ensure $A - \{ s \}$ is feasible.
- Max revenue because: $cap(A, B) = \sum_{v \in B:\, p_v > 0} p_v + \sum_{v \in A:\, p_v < 0} (-p_v)$

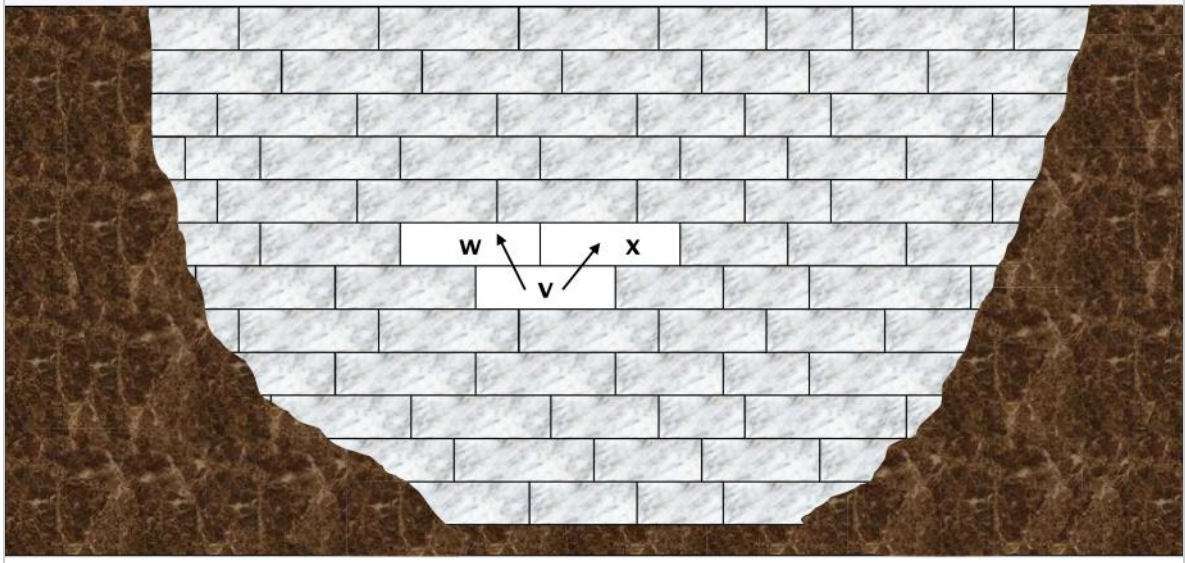
$$= \underbrace{\sum_{v:\, p_v > 0} p_v}_{\text{constant}} - \sum_{v \in A} p_v$$



Open-pit mining

Open-pit mining. (studied since early 1960s)

- Blocks of earth are extracted from surface to retrieve ore.
- Each block v has net value p_v = value of ore − processing cost.
- Can't remove block v before w or x.



Bad case for Ford-Fulkerson

Q. Is generic Ford-Fulkerson algorithm poly-time in input size?

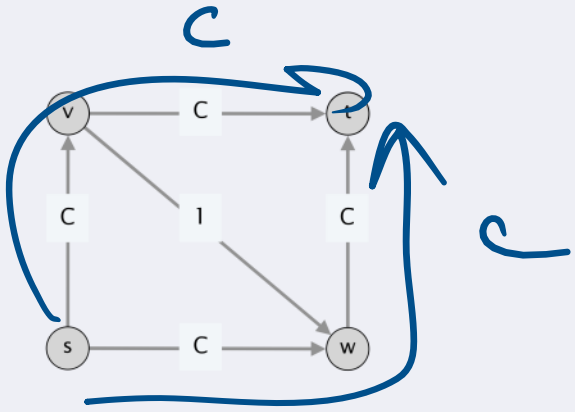
m, n, and log C

$O(m \cdot n \cdot C)$

A. No. If max capacity is C, then algorithm can take $\geq C$ iterations.

- $s \to v \to w \to t$
- $s \to w \to v \to t$
- $s \to v \to w \to t$
- $s \to w \to v \to t$
- ...
- $s \to v \to w \to t$
- $s \to w \to v \to t$

each augmenting path sends only 1 unit of flow (# augmenting paths = 2C)



val(f) = 1 2

Choosing good augmenting paths

Use care when selecting augmenting paths.

- Some choices lead to exponential algorithms.
- Clever choices lead to polynomial algorithms.
- If capacities are irrational, algorithm not guaranteed to terminate!

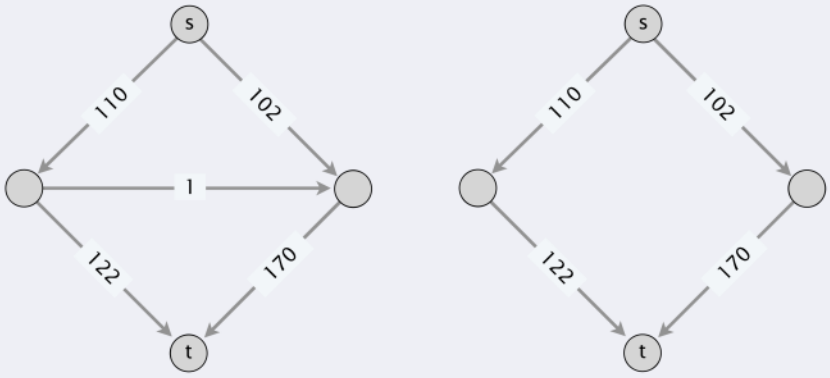
Goal. Choose augmenting paths so that:

- Can find augmenting paths efficiently.
- Few iterations.

Capacity-scaling algorithm

Intuition. Choose augmenting path with highest bottleneck capacity: it increases flow by max possible amount in given iteration.

- Don't worry about finding exact highest bottleneck path.
- Maintain scaling parameter Δ.
- Let $G_f(\Delta)$ be the subgraph of the residual graph consisting only of arcs with capacity $\geq \Delta$.



G_f $G_f(\Delta)$, $\Delta = 100$

Capacity-scaling algorithm

CAPACITY-SCALING(G, s, t, c)

FOREACH edge $e \in E : f(e) \leftarrow 0$.

$\Delta \leftarrow$ largest power of 2 $\leq C$.

WHILE ($\Delta \geq 1$)

  $G_f(\Delta) \leftarrow \Delta$-residual graph.

  WHILE (there exists an augmenting path P in $G_f(\Delta)$)

    $f \leftarrow$ AUGMENT (f, c, P).

    Update $G_f(\Delta)$.

  $\Delta \leftarrow \Delta / 2$.

RETURN f.

C = max edge capacity in G

≤ O(m) iterations

m = # edges in G

Overall: $O(m \cdot \log_2 C)$ augmentations.
Runtime : $O(m^2 \cdot \log_2 C)$.

Capacity-scaling algorithm: proof of correctness

Assumption. All edge capacities are integers between 1 and C.

Integrality invariant. All flow and residual capacity values are integral.

Theorem. If capacity-scaling algorithm terminates, then f is a max-flow.

Pf.

- By integrality invariant, when $\Delta = 1 \Rightarrow G_f(\Delta) = G_f$.
- Upon termination of $\Delta = 1$ phase, there are no augmenting paths. ▪

Capacity-scaling algorithm: analysis of running time

Lemma 1. The outer while loop repeats $1 + \lceil \log_2 C \rceil$ times.

Pf. Initially $C/2 < \Delta \leq C$; Δ decreases by a factor of 2 in each iteration. ▪

Lemma 2. Let f be the flow at the end of a Δ-scaling phase. Then, the value of the max-flow $\leq val(f) + m\,\Delta$. ← proof on next slide

Lemma 3. There are at most $2m$ augmentations per scaling phase.

Pf.

- Let f be the flow at the end of the previous scaling phase.
- LEMMA 2 $\Rightarrow$ $val(f^*) \leq val(f) + 2\,m\,\Delta$.
- Each augmentation in a Δ-phase increases $val(f)$ by at least Δ. ▪

Theorem. The scaling max-flow algorithm finds a max flow in $O(m \log C)$ augmentations. It can be implemented to run in $O(m^2 \log C)$ time.

Pf. Follows from LEMMA 1 and LEMMA 3. ▪

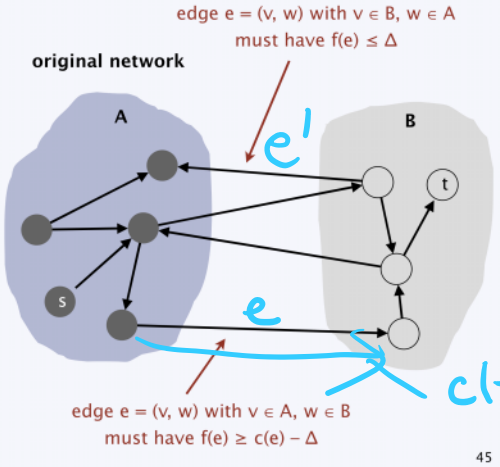
Capacity-scaling algorithm: analysis of running time

Lemma 2. Let f be the flow at the end of a Δ-scaling phase. Then, the value of the max-flow $\leq val(f) + m\,\Delta$.

Pf.

- We show there exists a cut (A, B) such that $cap(A, B) \leq val(f) + m\,\Delta$.
- Choose A to be the set of nodes reachable from s in $G_f(\Delta)$.
- By definition of cut A, $s \in A$.
- By definition of flow f, $t \notin A$.

$$
\begin{aligned}
val(f) &= \sum_{e \text{ out of } A} f(e) - \sum_{e \text{ in to } A} f(e) \\
&\geq \sum_{e \text{ out of } A} (c(e) - \Delta) - \sum_{e \text{ in to } A} \Delta \\
&= \sum_{e \text{ out of } A} c(e) - \sum_{e \text{ out of } A} \Delta - \sum_{e \text{ in to } A} \Delta \\
&\geq cap(A, B) - m\Delta \quad \blacksquare
\end{aligned}
$$

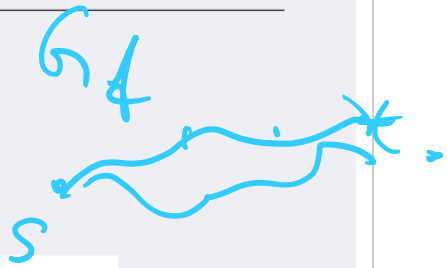


Shortest augmenting path

Q. Which augmenting path?

A. The one with the fewest number of edges.

can find via BFS



SHORTEST-AUGMENTING-PATH(G, s, t, c)

FOREACH $e \in E : f(e) \leftarrow 0$.
  $G_f \leftarrow$ residual graph.
  WHILE (there exists an augmenting path in G_f)
    $P \leftarrow$ BREADTH-FIRST-SEARCH (G_f, s, t).
    $f \leftarrow$ AUGMENT (f, c, P).
    Update G_f.
RETURN f.

Shortest augmenting path: overview of analysis

L1. Throughout the algorithm, length of the shortest path never decreases.

L2. After at most m shortest path augmentations, the length of the shortest augmenting path strictly increases.

Theorem. The shortest augmenting path algorithm runs in $O(m^2 n)$ time.

Pf.

- $O(m + n)$ time to find shortest augmenting path via BFS.
- $O(m)$ augmentations for paths of length k.
- If there is an augmenting path, there is a simple one.

  $\Rightarrow$ $1 \le k < n$

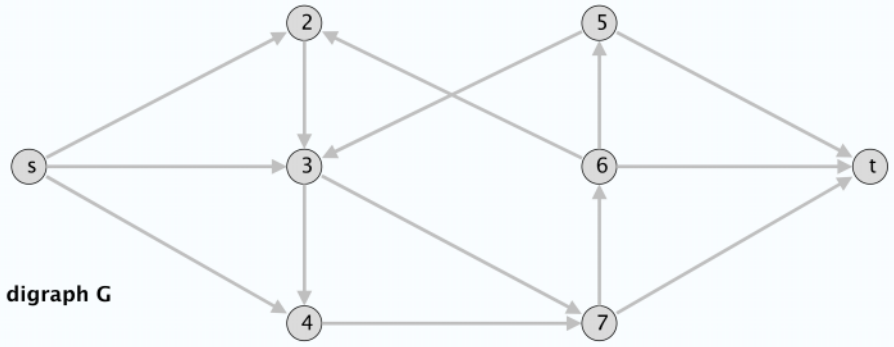
  $\Rightarrow$ $O(m\, n)$ augmentations. ▪

We will skip the proof.

Edge-disjoint paths

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

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



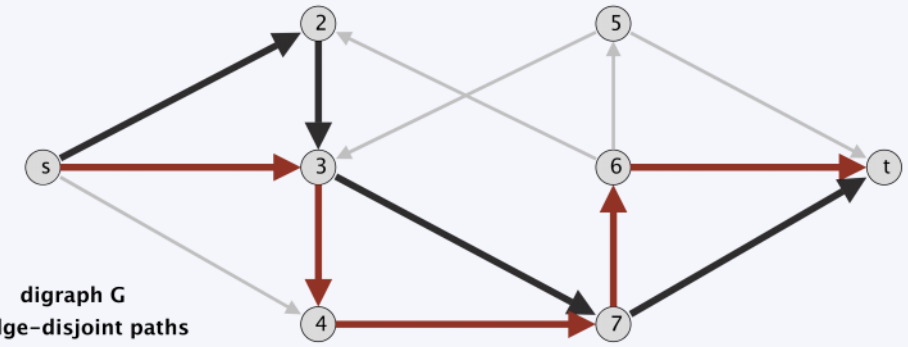
digraph G

Edge-disjoint paths

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

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

Ex. Communication networks.



digraph G
2 edge-disjoint paths

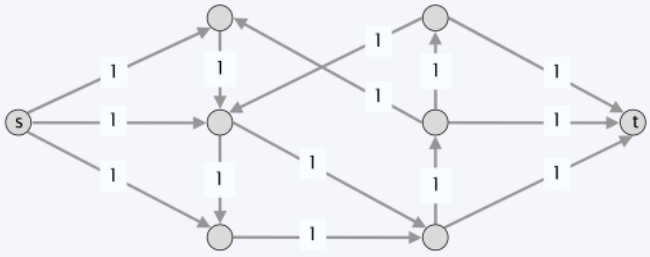
Edge-disjoint paths

Max flow formulation. Assign unit capacity to every edge.

Theorem. Max number edge-disjoint $s \rightarrow t$ paths equals value of max flow.

Pf. $\leq$

- Suppose there are k edge-disjoint $s \rightarrow t$ paths $P_1, \ldots, P_k$.
- Set $f(e) = 1$ if e participates in some path P_j; else set $f(e) = 0$.
- Since paths are edge-disjoint, f is a flow of value k. ▪



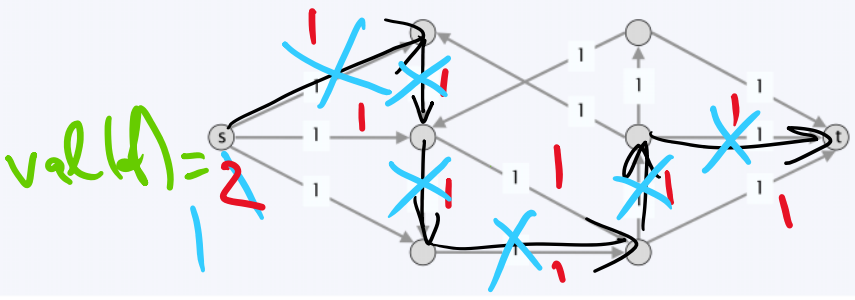
Edge-disjoint paths

Max flow formulation. Assign unit capacity to every edge.

Theorem. Max number edge-disjoint $s \rightarrow t$ paths equals value of max flow.

Pf. $\geq$

- Suppose max flow value is k.
- Integrality theorem $\Rightarrow$ there exists 0-1 flow f of value k.
- Consider edge (s, u) with $f(s, u) = 1$.
  - by 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. ▪

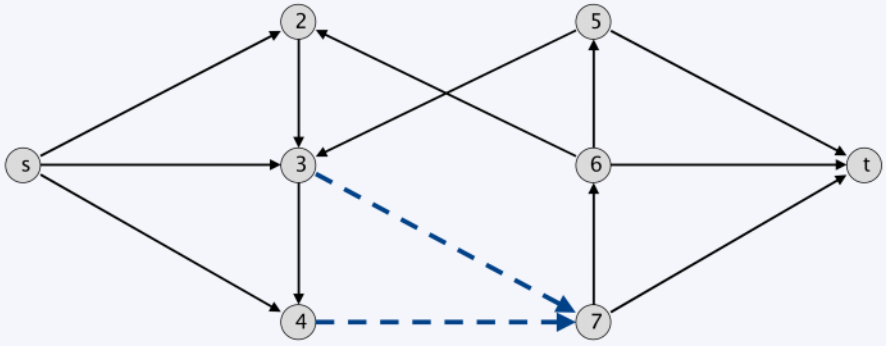


can eliminate cycles to get simple paths in O(mn) time if desired (flow decomposition)

Network connectivity

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

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



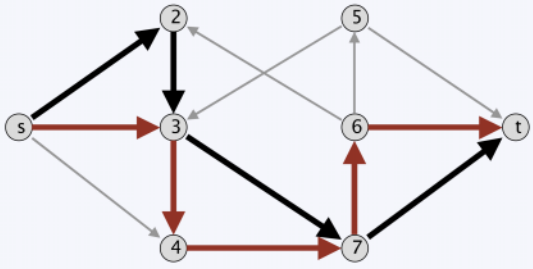
25

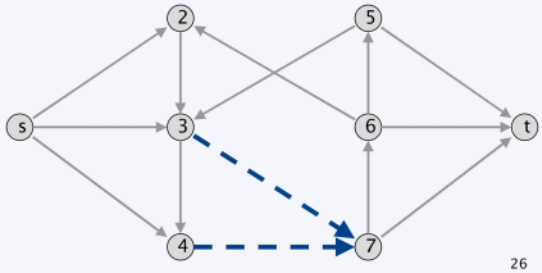
Menger's theorem

Theorem. [Menger 1927] The max number of edge-disjoint $s \rightarrow t$ paths is equal to the min number of edges whose removal disconnects t from s.

Pf. $\leq$

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





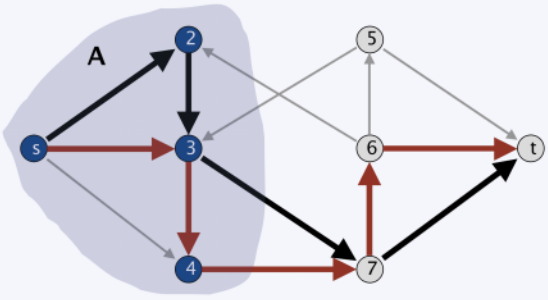
26

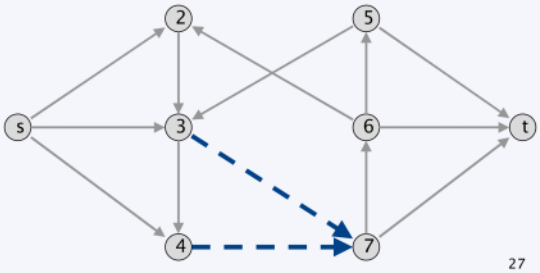
Menger's theorem

Theorem. [Menger 1927] The max number of edge-disjoint $s \rightarrow t$ paths equals the min number of edges whose removal disconnects t from s.

Pf. $\geq$

- Suppose max number of edge-disjoint 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. ▪





27