Applications of the Maxflow Problem

7.5 Bipartite Matching

Bipartite Matching

- Input: undirected, bipartite graph $G = (L \cup R, E)$.
- $M \subseteq E$ is a matching if each node appears in at most one edge in $M$.
- Max matching: find a max cardinality matching.

How to solve using maxflow?
- Need a source and sink
- Graph needs to be directed
- Capacities?
- How to interpret the flow?

Source + sink:

Capacities:
Max flow solution.
- Create directed graph $G' = (L \cup R \cup \{s, t\}, E')$.
- 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 $t$.

Theorem. Max cardinality matching in $G = \text{value of max flow in } G'$.
Proof.
- Let $f$ be a max flow in $G'$ of value $k$.
- Consider $M$ = set of edges from $L$ to $R$ with $f(e) = 1$.
  - each node in $L$ and $R$ participated in at most one edge in $M$, and therefore represent a matching with $|M| = k$.

Running time?
- $O(m \text{val}(f^*)) = O(mn)$. 

Bipartite Matching: Proof of Correctness

Reductions
We took problem A (bipartite matching) and showed that it can be solved using an instance of problem B (maxflow).
This is an example of reducing problem A to problem B.
Reduction: mapping from an instance of A to an instance of B, such that the solution to the instance of B we can construct the solution of the instance of A.
If $A$ reduces to $B$, which one is harder?
Reductions

Reduce, Reuse, Recycle...

If we have a solution to one problem and we can use this solution to solve another problem, we do not need to write a new program, we can reuse the existing code, and reduce the new problem (change its input (and output)), so it can use the existing code to solve it.

Example: We have a max heap, but we need a min heap. How can we use the max heap to perform min heap operations, without changing one bit of the max heap code? (Assume the heap holds integers)

- Insert(x): InsertMaxHeap(-x);
- Extract(): x = ExtractMaxHeap(); return -x;

7.6 Disjoint Paths

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

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

Example: Communication networks.

Max flow formulation: assign unit capacity to every edge.

Theorem. Max number edge-disjoint \( s \)-\( t \) paths equals max flow value.

Proof.

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

Edge Disjoint Paths

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

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

Example: Communication networks.

Max flow formulation: assign unit capacity to every edge.

Theorem. Max number edge-disjoint \( s \)-\( t \) paths equals max flow value.

Proof.

- 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 \( k \) edge-disjoint paths.
Network Connectivity

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

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

Edge Disjoint Paths and Network Connectivity

**Theorem.** (Menger 1927) The max number of edge-disjoint \( s \)-\( t \) paths is equal to the min number of edges whose removal disconnects \( t \) from \( s \).

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

\[ \square \]

Circulation with Demands

Circulation with demands:
- Directed graph \( G = (V, E) \)
- Edge capacities \( c(e), e \in E \)
- Node supply and demands \( d(v), v \in V \)

Def. A circulation is a function that satisfies:
- For each \( e \in E \):
  \[ f(e) \leq c(e) \] (capacity)
- For each \( v \in V \):
  \[ \sum_{e \in \delta^+(v)} f(e) - \sum_{e \in \delta^-(v)} f(e) = d(v) \] (conservation)

Circulation problem: given \( (V, E, c, d) \), does there exist a circulation?

Edge Disjoint Paths and Network Connectivity

**Theorem.** (Menger 1927) The max number of edge-disjoint \( s \)-\( t \) paths is equal to the min number of edges whose removal disconnects \( t \) from \( s \).

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

\[ \square \]

7.7 Extensions to Max Flow

Necessary condition: \( \sum d(v) = \sum d'(v) \)

**Proof.** Sum conservation constraints for every demand node \( v \).
Max flow formulation?

- Add new source \( s \) and sink \( t \).
- For each \( v \) with \( d(v) > 0 \), add edge \( (v, t) \) with capacity \( d(v) \).
- For each \( v \) with \( d(v) < 0 \), add edge \( (s, v) \) with capacity \( -d(v) \).
- Claim: \( G \) has circulation if \( G' \) has max flow of value \( D \).

Reductions

We have seen the following problems that can be reduced to maxflow/mincut:

- Bipartite matching
- Number of disjoint paths
- Circulation with demands
- Image segmentation

There are many more!

Reduction as a problem solving tool: If you have an algorithm for a very general problem, that gives you a tool for solving lots of other problems.