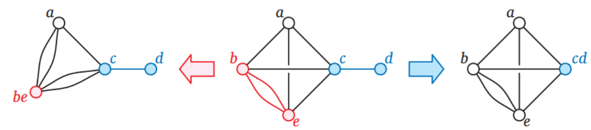
# Algorithms Final Cheat Sheet

RandomizedAlgorithms

## Max Flows and Min Cuts




### Blind Guess

**function** GuessMinCut($G$)
  **for** $i \leftarrow n, 2$ **do**
    pick a random edge $e$ in $G$
    $G \leftarrow G/e$
  **end for**
  **return** the only cut in $G$
**end function**

$P(n) = \frac{2}{n(n-1)}$

### Repeated Guessing

**function** KargerMinCut($G$)
  $mink \leftarrow \infty$
  **for** $i \leftarrow 1, N$ **do**
    $X \leftarrow$ GuessMinCut($G$)
    **if** $|X| < mink$ **then**
      $mink \leftarrow |X|$
      $minX \leftarrow X$
    **end if**
  **end for**
  **return** $minX$
**end function**

Set $N = c\binom{n}{2}\ln n$ for some constant $c$. $P(n) \geq 1 - \frac{1}{n^c}$. KargerMinCut computes the min cut of any $n$-node graph with high probability in $O(n^4 \log n)$ time.

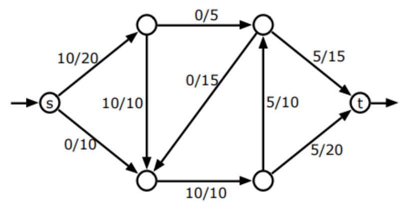
### Not-So-Blindly Guessing

**function** Contract($G, m$)
  **for** $i \leftarrow n, m$ **do**
    pick a random edge $e$ in $G$
    $G \leftarrow G/e$
  **end for**
**end function**
**function** BetterGuess($G$)
  **if** $G$ has more than 8 vertices **then**
    $G_1 \leftarrow$ Contract($G, n/\sqrt{2}+1$)
    $G_2 \leftarrow$ Contract($G, n/\sqrt{2}+1$)
    $X_1 \leftarrow$ BetterGuess($G_1$)
    $X_2 \leftarrow$ BetterGuess($G_2$)
    **return** $\min(X_1, X_2)$
  **else**
    use brute force
  **end if**
**end function**

$P(n) \geq 1/\log n$. The running time is $O(n^2 \log n)$.

### Flows

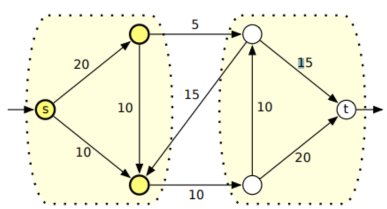
A *flow* is a function $f$ that satisfies the *conservation constraint* at every vertex $v$: the total flow *into* $v$ is equal to the total flow *out* of $v$.
A flow $f$ is *feasible* if $f(e) \leq c(e)$ for each edge $e$. A flow *saturates* edge $e$ if $f(e) = f(c)$, and *avoids* edge $e$ if $f(e) = 0$.




An $(s,t)$-flow with value 10. Each edge is labeled with its flow/capacity.

### Cuts

A *cut* is a partition of the vertices into disjoint subsets $S$ and $T$ - meaning $S \cup T = V$ and $S \cap T = \emptyset$ - where $s \in S$ and $t \in T$.
If we have a capacity function $c$, the *capacity* of a cut is the sum of the capacities of the edges that start in $S$ and end in $T$. The definition is asymmetric; edges that start in $T$ and end in $S$ are unimportant. The *min-cut problem* is to compute a cut whose capacity is as large as possible.

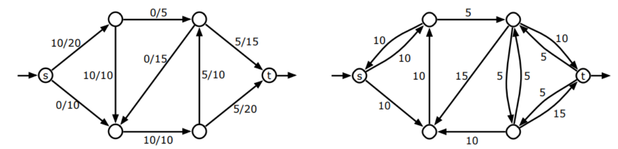



An $(s,t)$-cut with capacity 15. Each edge is labeled with its capacity.

**Theorem 1 (Maxflow Mincut Theorem)** *In any flow network, the value of the maximum flow is equal to the capacity of the minimum cut.*

### Residual Capacity

$$c_f(u \to v) = \begin{cases} c(u \to v) - f(u \to v) & \text{if } u \to v \in E \\ f(v \to u) & \text{if } v \to u \in E \\ 0 & \text{otherwise} \end{cases}$$



A flow $f$ in a weighted graph $G$ and the corresponding residual graph $G_f$.

### Augmenting Paths

Suppose there is a path $s = v_0 \to v_1 \to \cdots \to v_r$ in the residual graph $G_f$. This is an *augmenting path*. Let $F = \min_i c_f(v_i \to v_{i+1})$ denote the maximum amount of flow that we can push through the augmenting path in $G_f$. We can augment the flow into a new flow function $f'$:

$$f'(u \to v) = \begin{cases} f(u \to v) + F & \text{if } u \to v \in s \\ f(u \to v) - F & \text{if } v \to u \in s \\ f(u \to v) & \text{otherwise} \end{cases}$$

### Ford-Fulkerson

Starting with the zero flow, repeatedly augment the flow along *any* path from $s$ to $t$ in the residual graph, until there is no such path.

### Further Work

The fastest known maximum flow algorithm, announced by James Orlin in 2012, runs in $O(VE)$ time.

## Flow/Cut Applications

SATandCNFSAT  PvNP