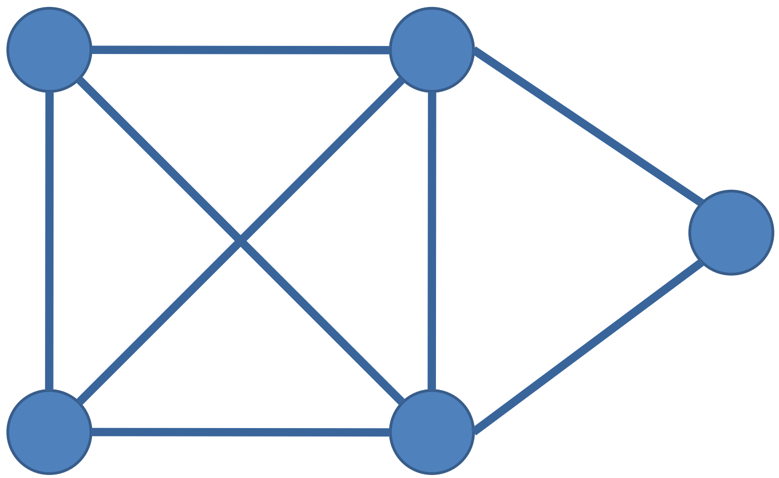
Randomized Algorithm

Karger's Min-Cut Algorithm

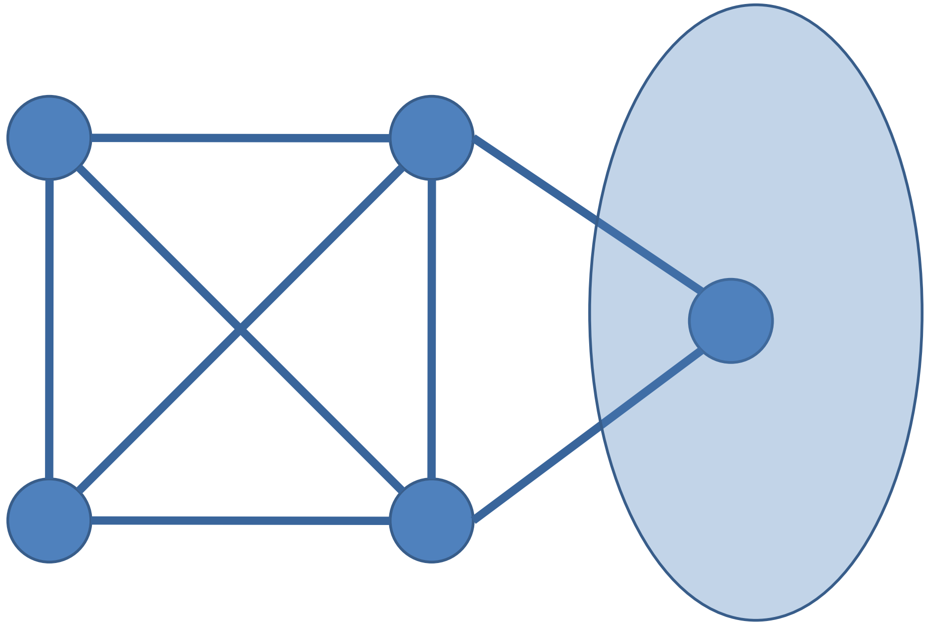
Karger's Min-Cut Algorithm

- Problem: Given a graph, find the cut of minimum size in the graph.
  - Cut: Partition of vertices into two non-empty sets. The edges between vertices in different sets are called cut edges. The size of the cut is the number of such edges.



Karger’s Min-Cut Algorithm

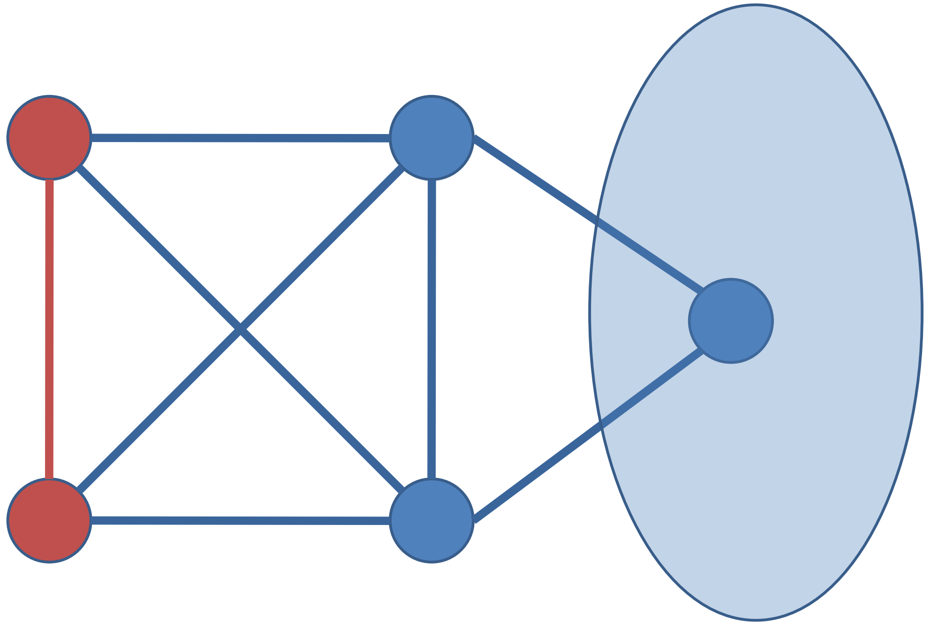
- Problem: Given a graph, find the cut of minimum size in the graph.
  - Cut: Partition of vertices into two non-empty sets. The edges between vertices in different sets are called cut edges. The size of the cut is the number of such edges.



Cut of minimum size

Karger's Min-Cut Algorithm

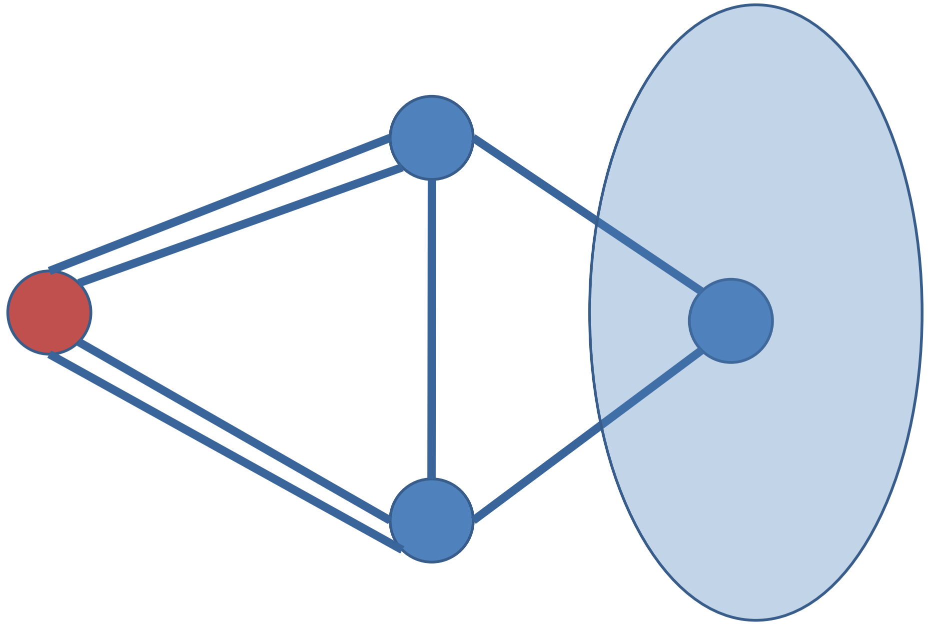
- *Collapse(e)*: Collapsing vertices across an edge **e=(u,v)**.
  - Merge vertices **u** and **v** and remove self loops. There may be multiple edges between same vertices (multi-graph).



Cut of minimum size

Karger's Min-Cut Algorithm

- *Collapse(e)*: Collapsing vertices across an edge **e=(u,v)**.
  - Merge vertices **u** and **v** and remove self loops. There may be multiple edges between same vertices (multi-graph).



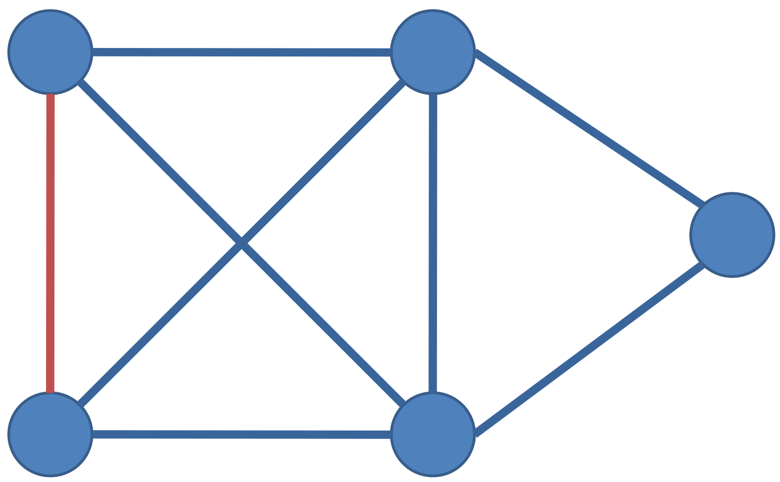
Cut of minimum size

Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times   *//For probability amplification.*
    - While **|V|>2**
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G $\leftarrow$ G'
    - The edges across the remaining two vertices are the candidate cut edges.
  - Output the best answer from the **P** trials.

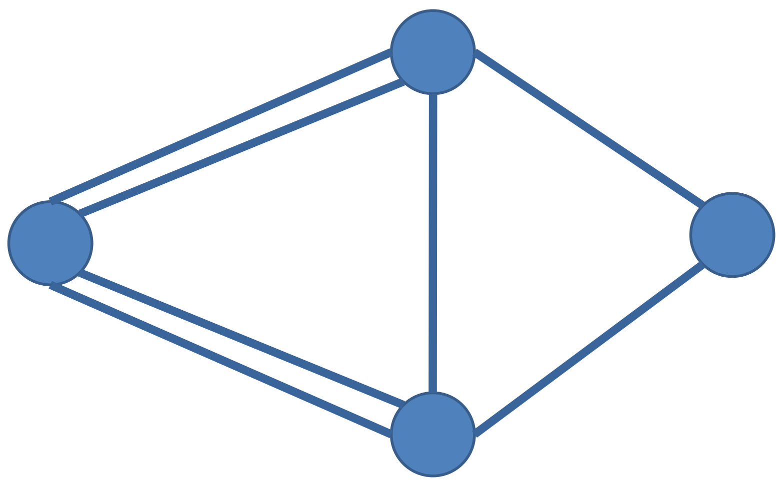
Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times   *//For probability amplification.*
    - While **|V|>2**
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.



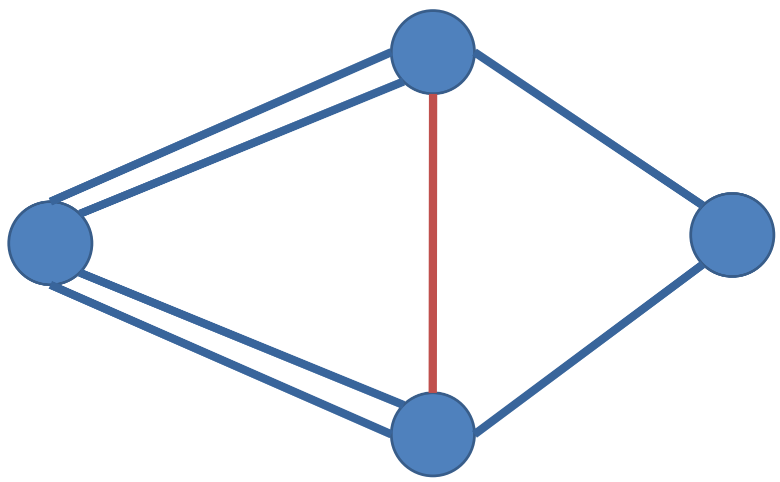
Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times   *//For probability amplification.*
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G $\leftarrow$ G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.



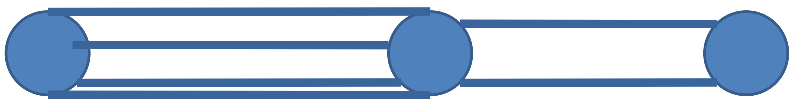
Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times   //*For probability amplification*.
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.



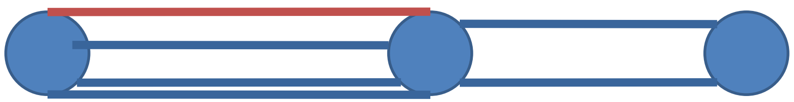
Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times *//For probability amplification.*
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.



Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times   *//For probability amplification.*
    - While **|V|>2**
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.



Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times   *//For probability amplification.*
    - While **|V|>2**
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.

The algorithm does well if the edges across a min-cut is never picked for collapsing.

Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times *//For probability amplification.*
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.

- Let **C** denote the edges in a minimum size cut in **G**.
- What is the probability that a randomly chosen edge belongs to **C**?

Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times   *//For probability amplification.*
    - While $|V|>2$
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.

- Let **C** denote the edges in a minimum size cut in **G**.
- What is the probability that a randomly chosen edge belongs to **C**?
  - Lemma 1: $\mathbf{Pr[e \in C] \leq 2/n}$.
    - Proof: Since $|\mathbf{E}| \geq \mathbf{n}|\mathbf{C}|/2$.

Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times *//For probability amplification.*
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.

- Let **C** denote the edges in a minimum size cut in **G**.
- <u>Lemma 1</u>: $\mathbf{Pr[e \in C] \leq 2/n}$.
- <u>Lemma 2</u>: If $\mathbf{e \notin C}$, then size of min-cut of **G'** is the same as the size of min-cut of **G** after performing Collapse(e).

Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times *//For probability amplification.*
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.

- Let **C** denote the edges in a minimum size cut in **G**.
- <u>Lemma 1</u>: $\mathbf{Pr[e \in C] \leq 2/n}$.
- <u>Lemma 2</u>: If $\mathbf{e \notin C}$, then size of min-cut of **G'** is the same as the size of min-cut of **G** after performing Collapse(e).
- Let $\mathbf{D_i}$ denote the event that none of the edges in **C** have been used in the first **i** iterations.
- $\mathbf{Pr[D_{n-2}] = \prod Pr[D_{i+1} \mid D_i] \geq (1-2/n)(1-2/n-1)\ldots(1/3)}$

Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times *//For probability amplification.*
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.

- Let **C** denote the edges in a minimum size cut in **G**.
- Lemma 1: $\mathbf{Pr[e \in C] \leq 2/n}$.
- Lemma 2: If $\mathbf{e \notin C}$, then size of min-cut of **G'** is the same as the size of min-cut of **G** after performing Collapse(e).
- Let $\mathbf{D_i}$ denote the event that none of the edges in **C** have been used in the first **i** iterations.
- $$\mathbf{Pr[D_{n-2}] = \prod Pr[D_{i+1} | D_i] \geq (1\text{-}2/n)(1\text{-}2/n\text{-}1)\ldots(1/3)}$$
  $$\mathbf{= 2/(n(n-1))}$$
  $$\mathbf{= \Omega(1/n^2)}$$

Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times //*For probability amplification.*
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.

- $\mathbf{Pr[D_{n-2}] = \prod Pr[D_{i+1} \mid D_i] \geq (1-2/n)(1-2/n-1)\ldots(1/3)}$
  $\mathbf{= 2/(n(n-1))}$
  $\mathbf{= \Omega(1/n^2)}$
- Let $\mathbf{P = n^2(\ln n)}$.
- What is the probability that the algorithm returns the correct answer?

Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times *//For probability amplification.*
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G $\leftarrow$ G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.

- $\mathbf{Pr[D_{n-2}] = \prod Pr[D_{i+1} \mid D_i] \geq (1\text{-}2/n)(1\text{-}2/n\text{-}1)\ldots(1/3)}$
  $\mathbf{= 2/(n(n\text{-}1))}$
  $\mathbf{= \Omega(1/n^2)}$
- Let $\mathbf{P = n^2(\ln n)}$.
- What is the probability that the algorithm returns the correct answer?
  - Pr[Algorithm returns min-cut] $\geq \mathbf{1 - (1 - 2/n^2)^{n\text{^}2\ln n}}$
    $\geq \mathbf{1 - 1/n}$

Karger's Min-Cut Algorithm

- Min-cut algorithm:
  - Repeat **P** times //*For probability amplification.*
    - While |**V**|>2
      - Pick a random edge **e** in the multi-graph **G** and perform **Collapse(e)** to obtain **G'**.
      - G ← G'
    - The edges across the remaining two vertices are the cut edges.
  - Output the best answer from the **P** trials.

- Let $\mathbf{P = n^2(\ln n)}$.
- Pr[Algorithm returns min-cut] $\geq \mathbf{1 - 1/n}$.
- <u>Running time</u>: $\mathbf{P.(n\text{-}2).n = O(n^4 \log n)}$.
- <u>Question</u>: Is it possible to improve the running time?
  - <u>Observation</u>: The probability of a cut **C** surviving becomes smaller as the graph shrinks in size. So, "there is a need to repeat for smaller graphs not for larger ones".

Karger's Min-Cut Algorithm

- Suppose we keep collapsing vertices until the number of vertices in the graph is $\mathbf{n/2}$.
- What is the probability that a min-cut **C** has survived?
  - Pr[**C** survives] $\geq \mathbf{(1\text{-}2/n)\ldots(1\text{-}2/(n/2+1))} \geq$ ¼.
- Starting from **G** we run the iterative collapse procedure four times independently to obtain graphs $\mathbf{G_1}$, $\mathbf{G_2}$, $\mathbf{G_3}$, and $\mathbf{G_4}$ that have $\mathbf{n/2}$ vertices. Repeat this step in a tree like fashion.
- Running time of above algorithm:
  $\mathbf{T(n) = 4T(n/2) + O(n^2) \Rightarrow T(n) = O(n^2 \log n)}$
- <u>Success probability</u>:

Karger's Min-Cut Algorithm

- Suppose we keep collapsing vertices until the number of vertices in the graph is **n/2**.
- What is the probability that a min-cut **C** has survived?
  - Pr[**C** survives] $\geq$ **(1-2/n)…(1-2/(n/2+1))** $\geq$ ¼.
- Starting from **G** we run the iterative collapse procedure four times independently to obtain graphs $\mathbf{G_1}$, $\mathbf{G_2}$, $\mathbf{G_3}$, and $\mathbf{G_4}$ that have **n/2** vertices. Repeat this step in a tree like fashion.
- Running time of above algorithm:
  $\mathbf{T(n) = 4T(n/2) + O(n^2) \Rightarrow T(n) = O(n^2 \log n)}$
- Success probability: Let **P(n)** denote the probability that a fixed min-cut **C** survives.
  - $\mathbf{P(n) \geq 1 - (1 - \frac{1}{4}.P(n/2))^4}$
  - $\mathbf{P(n) = \Omega(1/\log n)}$

Karger's Min-Cut Algorithm

- Suppose we keep collapsing vertices until the number of vertices in the graph is **n/2**.
- What is the probability that a min-cut **C** has survived?
  - Pr[**C** survives] ≥ **(1-2/n)...(1-2/(n/2+1))** ≥ ¼.
- Starting from **G** we run the iterative collapse procedure four times independently to obtain graphs $\mathbf{G_1}$, $\mathbf{G_2}$, $\mathbf{G_3}$, and $\mathbf{G_4}$ that have **n/2** vertices. Repeat this step in a tree like fashion.
- Running time of above algorithm:
  $\mathbf{T(n) = 4T(n/2) + O(n^2)}$ ➔ $\mathbf{T(n) = O(n^2 \log n)}$
- <u>Success probability</u>: Let P(n) denote the probability that a fixed min-cut C survives.
  - $\mathbf{P(n) \geq 1 - (1 - \tfrac{1}{4}.P(n/2))^4}$
  - $\mathbf{P(n) = \Omega(1/\log n)}$
- So, we repeat the algorithm $\mathbf{O(\log^2 n)}$ times to obtain **C** w.h.p.
- <u>Overall running time</u>: $\mathbf{O(n^2 \log^3 n)}$.

Types of randomized algorithms

Monte Carlo Vs Las Vegas

Types of randomized algorithms

- We talked about two randomized algorithms:
  1. Randomized quick-sort
  2. Karger's min-cut algorithm

Types of randomized algorithms

- We talked about two randomized algorithms:
  1. Randomized quick-sort
     - The algorithm always produces correct answer.
  2. Karger's min-cut algorithm
     - The algorithm produces an incorrect answer with some small probability.

Types of randomized algorithms

- We talked about two randomized algorithms:
  1. Randomized quick-sort
     - The algorithm always produces correct answer.
     - Such randomized algorithms are called **Las Vegas** algorithms.
  2. Karger's min-cut algorithm
     - The algorithm produces an incorrect answer with some small probability.
     - Such randomized algorithms are called **Monte Carlo** algorithms.

Types of randomized algorithms

- We talked about two randomized algorithms:
  1. Randomized quick-sort
     - The algorithm always produces correct answer.
     - Such randomized algorithms are called **Las Vegas** algorithms.
  2. Karger's min-cut algorithm
     - The algorithm produces an incorrect answer with some small probability.
     - Such randomized algorithms are called **Monte Carlo** algorithms.
- <u>Theorem (Monte-Carlo to Las Vegas)</u>: Given a Monte-Carlo algorithm for solving a problem that runs in expected time **T(n)** and has a success probability of **γ(n)**. Further, given a solution, there is a deterministic algorithm can verify the correctness of the solution in time **t(n)**. Then there is a Las-Vegas algorithm for the problem that runs in expected time ________________.

Types of randomized algorithms

- We talked about two randomized algorithms:
  1. Randomized quick-sort
     - The algorithm always produces correct answer.
     - Such randomized algorithms are called **Las Vegas** algorithms.
  2. Karger's min-cut algorithm
     - The algorithm produces an incorrect answer with some small probability.
     - Such randomized algorithms are called **Monte Carlo** algorithms.
- <u>Theorem (Monte-Carlo to Las Vegas)</u>: Given a Monte-Carlo algorithm for solving a problem that runs in expected time $\mathbf{T(n)}$ and has a success probability of $\boldsymbol{\gamma(n)}$. Further, given a solution, there is a deterministic algorithm can verify the correctness of the solution in time $\mathbf{t(n)}$. Then there is a Las-Vegas algorithm for the problem that runs in expected time $\mathbf{(T(n) + t(n))/\boldsymbol{\gamma}(n)}$.