Stochastic Enumeration Method for Counting Trees

Radislav Vaisman & Dirk P. Kroese

School of Mathematics and Physics
The University of Queensland Australia
http://acems.smp.uq.edu.au/

r.vaisman@uq.edu.au, kroese@maths.uq.edu.au

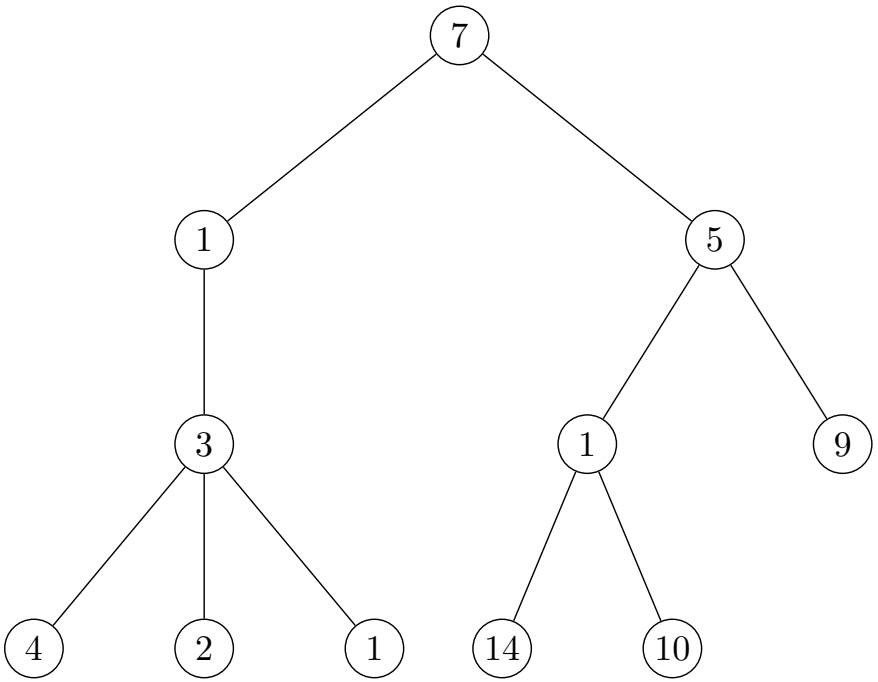
July 23, 2015

THE UNIVERSITY

Overview

1. The Tree Counting Problem — a Motivation
   - Is this hard?
   - Is this interesting?
   - Previous and Current Work
2. Knuth's estimator
   - Problem with Knuth's estimator
   - What can we do about this?
3. From Knuth to Stochastic Enumeration (SE) Algorithm
4. Some encouraging numerical results
5. Analysis and almost sure *Fully Polynomial Randomized Approximation Scheme* for random trees (Super–Critical Branching Process)
6. What next?

The Tree Counting Problem

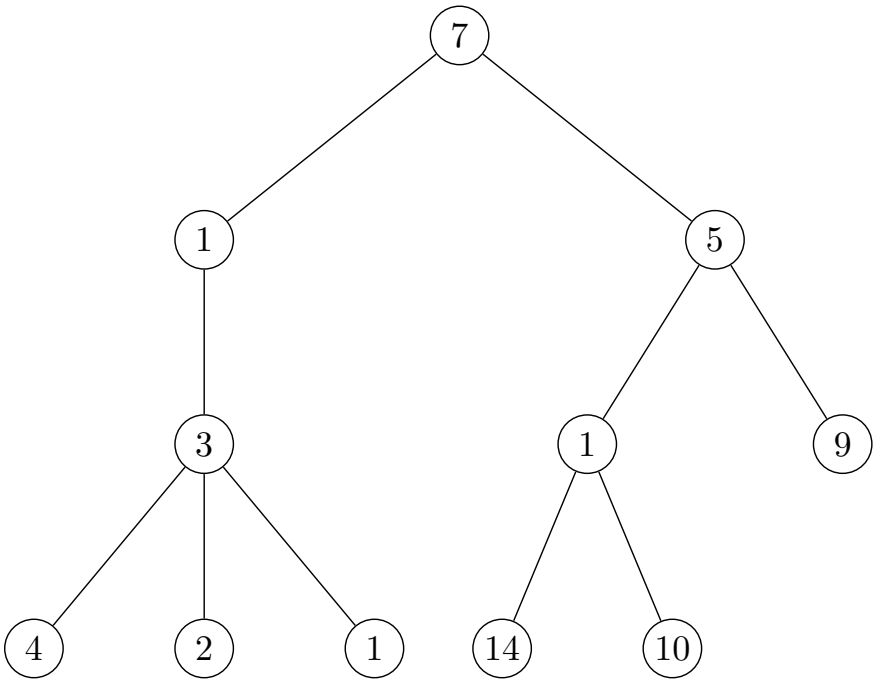


- Consider a rooted tree $T = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$.
- Which each node v is associated a cost $c(v) \in \mathbb{R}$, (it is also possible that $C(v)$ is a random variable).
- The main quantity of interest is the total cost of the tree,

$$\mathrm{Cost}(T) = \sum_{v \in \mathcal{V}} c(v), \text{ or for r.v:}$$

$$\mathrm{Cost}(T) = \mathbb{E}\left(\sum_{v \in \mathcal{V}} C(v) \right).$$

The Tree Counting Problem



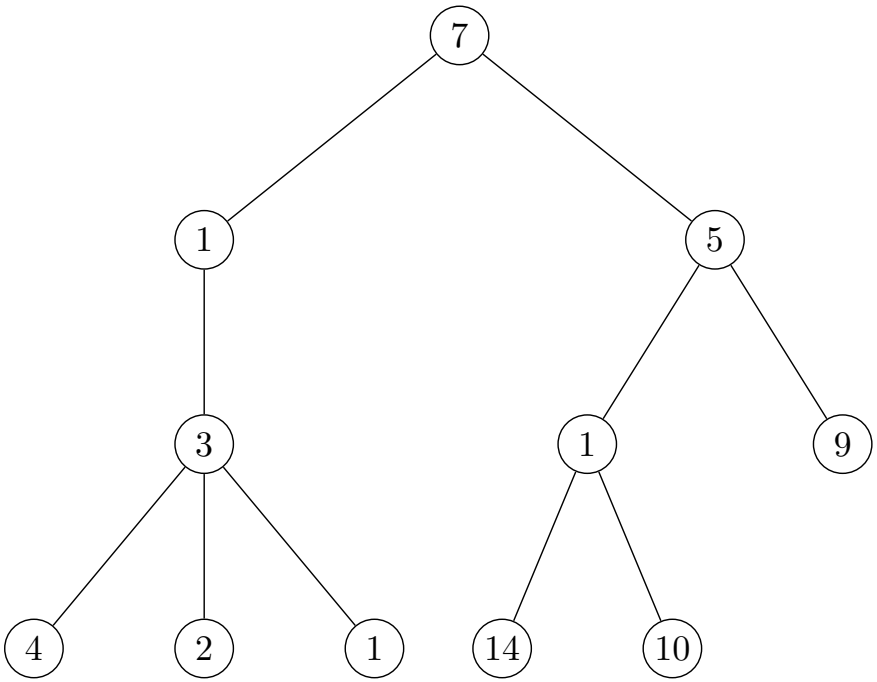
- Consider a rooted tree $T = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$.
- Which each node v is associated a cost $c(v) \in \mathbb{R}$, (it is also possible that $C(v)$ is a random variable).
- The main quantity of interest is the total cost of the tree,

$$\text{Cost}(T) = \sum_{v \in \mathcal{V}} c(v), \text{ or for r.v:}$$

$$\text{Cost}(T) = \mathbb{E}\left(\sum_{v \in \mathcal{V}} C(v) \right).$$

- Linear time solution? (BFS, DFS).

The Tree Counting Problem



- Consider a rooted tree $T = (\mathcal{V}, \mathcal{E})$ with node set $\mathcal{V}$ and edge set $\mathcal{E}$.
- Which each node v is associated a cost $c(v) \in \mathbb{R}$, (it is also possible that $C(v)$ is a random variable).
- The main quantity of interest is the total cost of the tree,

$$\text{Cost}(T) = \sum_{v \in \mathcal{V}} c(v), \text{ or for r.v:}$$

$$\text{Cost}(T) = \mathbb{E}\left(\sum_{v \in \mathcal{V}} C(v) \right).$$

- Linear time solution? (BFS, DFS).

What if the set $|\mathcal{V}|$ is large?

Is this hard?

The general problem of estimating the cost of a tree is at least #P, (Valiant, 1979). (Counting CNF formula solutions)

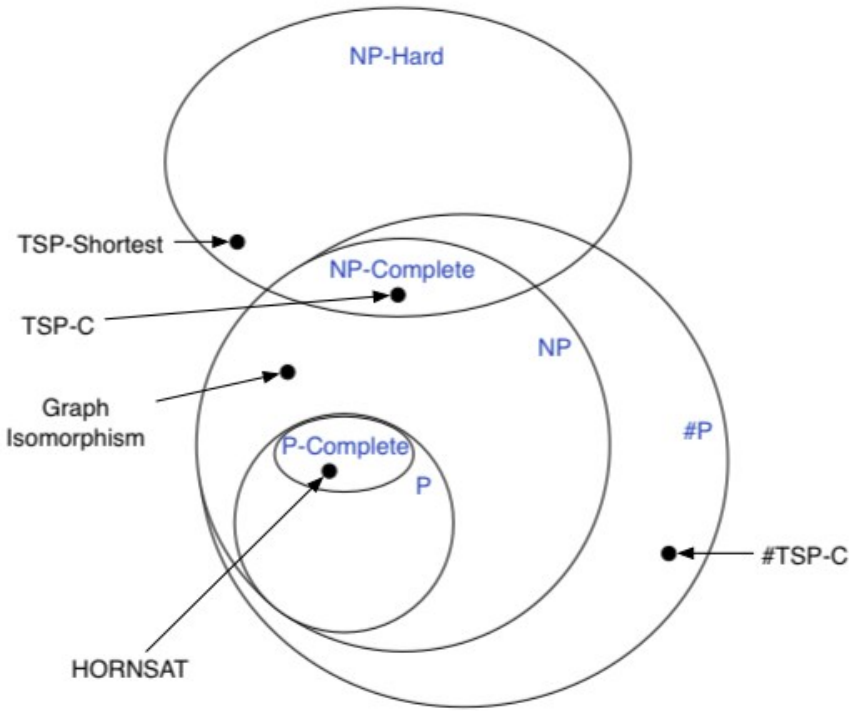


Figure: Complexity classes

Is this an interesting problem?

- From theoretical point of view.
  - Theoretical research of *complexity classes* — (#P Counting Problems).
  - New sampling strategies for stochastic simulation algorithms.
- In Practice.
  - Early estimates of the size of backtrack trees, (Knuth).
  - Model Counting — (Bayesian inference)
  - Network Reliability and sensitivity.
  - Hopefully much more...

Previous and current work

Previous work


Donald E. Knuth (1975). Estimating the Efficiency of Backtrack Programs. *Math. Comp.* 29.


Paul W. Purdom (1978). Tree Size by Partial Backtracking *SIAM J. Comput.* 7(4) 481-491.


Pang C. Chen (1992) Heuristic Sampling: A Method for Predicting the Performance of Tree Searching Programs. *SIAM J. Comput.* 21(2) 295-315.

Few additional attempts – based on Knuth's estimator.

Previous and current work

Previous work

- Donald E. Knuth (1975). Estimating the Efficiency of Backtrack Programs. *Math. Comp.* 29.
- Paul W. Purdom (1978). Tree Size by Partial Backtracking *SIAM J. Comput.* 7(4) 481-491.
- Pang C. Chen (1992) Heuristic Sampling: A Method for Predicting the Performance of Tree Searching Programs. *SIAM J. Comput.* 21(2) 295-315.

Few additional attempts – based on Knuth's estimator.

Current work

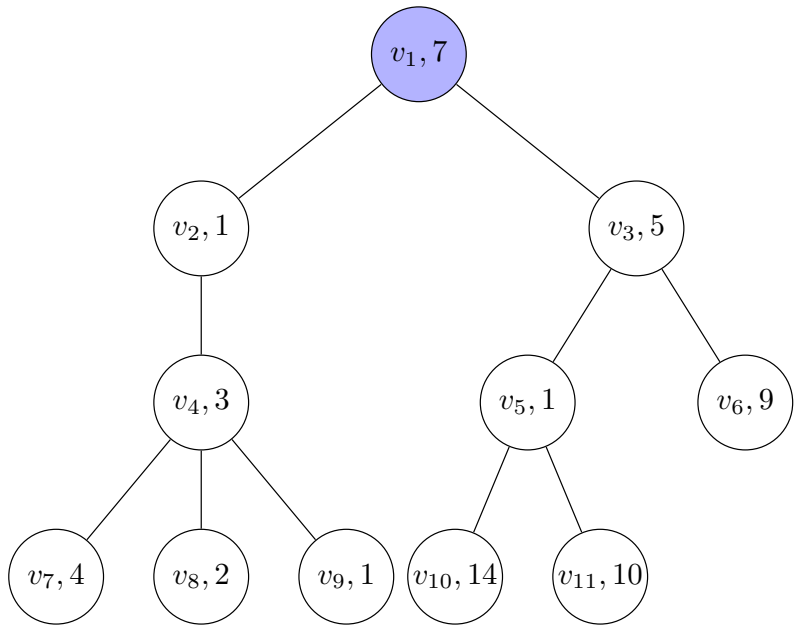
- R. Vaisman and D. P. Kroese, (2015). Stochastic Enumeration Method for Counting Trees. *Methodology & Computing in Applied Probability.*
- R. Vaisman, D. P. Kroese and I. B. Gertsbakh, (2015). Improved Sampling Plans for Combinatorial Invariants of Coherent Systems. *IEEE Transactions on Reliability.*

Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$k = 0,\ D = 1,\ X_0 = v_1,\ C = 7.$$

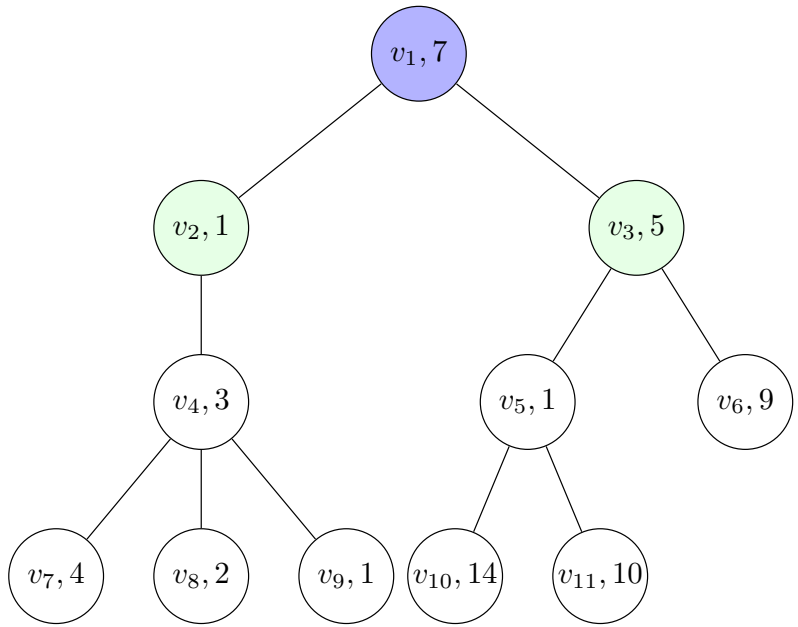


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$S(X_0) = \{v_2, v_3\}, \; D_0 = 2.$$

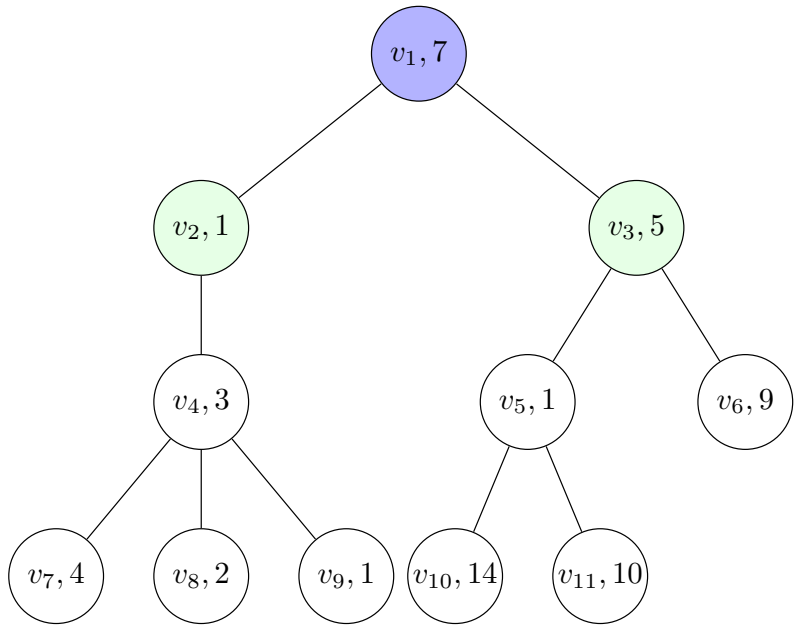


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$S(X_0) = \{v_2, v_3\}, \; D_0 = 2.$$

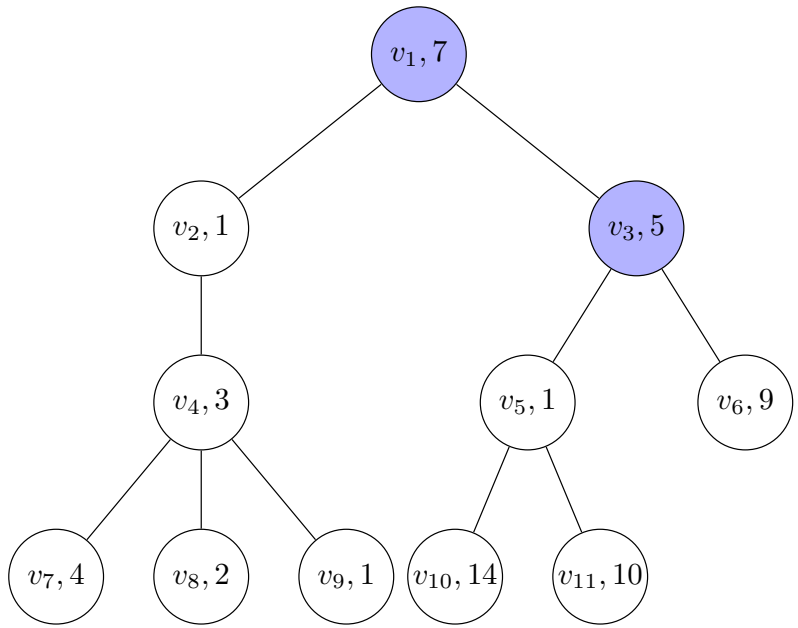


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$k = 1,\ X_1 = v_3,\ D = 1 \cdot D_0 = 2,$$
$$C = 7 + 5 \cdot 2 = 17.$$

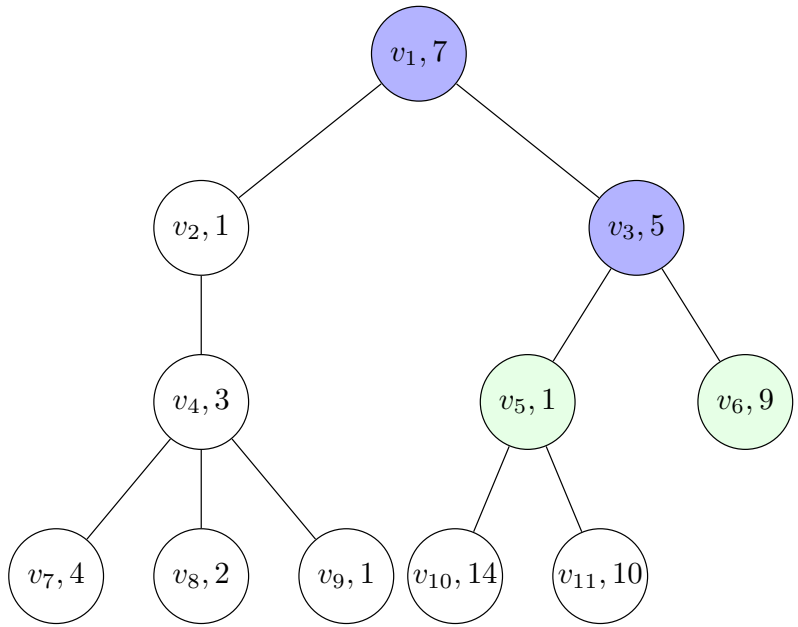


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$S(X_1) = \{v_5, v_6\}, \; D_1 = 2.$$

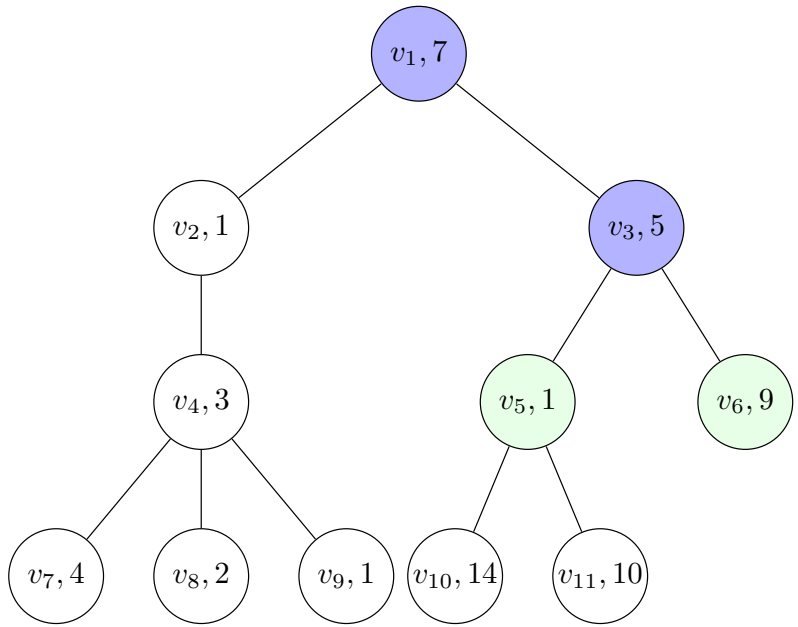


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$S(X_1) = \{v_5, v_6\}, \; D_1 = 2.$$

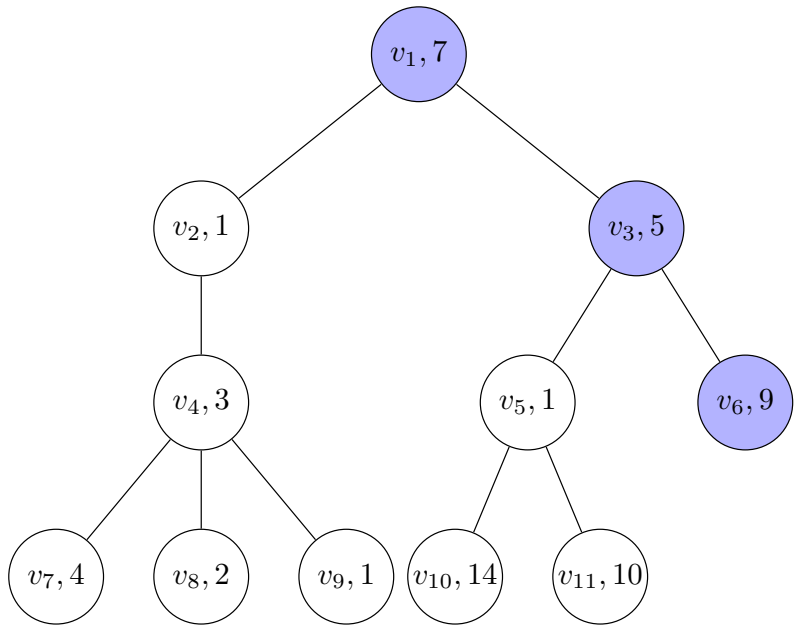


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$k = 2,\ X_2 = v_6,\ D = 2 \cdot D_1 = 4,$$
$$C = 7 + 5 \cdot 2 + 9 \cdot 4 = 53.$$

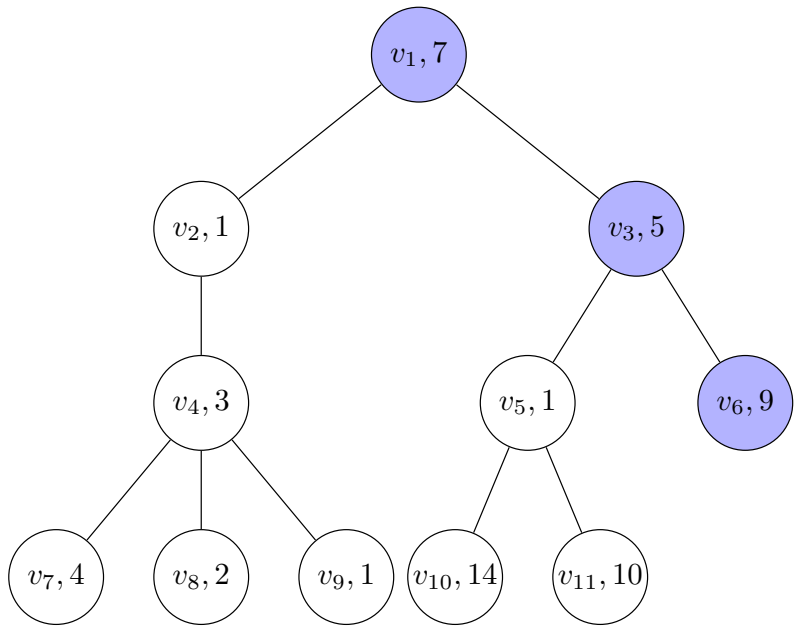


Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$X_2 = v_6,\ S(X_2) = \emptyset,\ D_2 = 0.$$



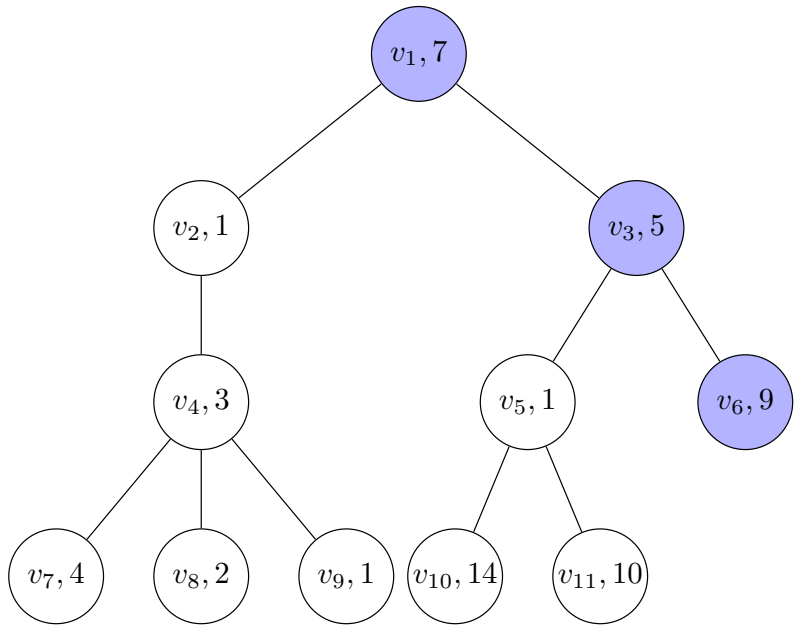
Knuth's estimator

Input: A tree T_v of height h, rooted at v.
Output: An unbiased estimator C of the total cost of tree T_v.

1. **(Initialization):** Set $k \leftarrow 0$, $D \leftarrow 1$, $X_0 = v$ and $C \leftarrow c(X_0)$. Here D is the product of all node degrees encountered in the tree.
2. **(Compute the successors):** Let $S(X_k)$ be the set of all successors of X_k and let D_k be the number of elements of $S(X_k)$. If $k = h$ or when $S(X_k)$ is empty, set $D_k = 0$.
3. **(Terminal position?):** If $D_k = 0$, the algorithm stops, returning C as an estimator of $\mathrm{Cost}(T_v)$.
4. **(Advance):** Choose an element $X_{k+1} \in S(X_k)$ at random, each element being equally likely. (Thus, each choice occurs with probability $1/D_k$.) Set $D \leftarrow D_k D$, then set $C \leftarrow C + c(X_{k+1})D$. Increase k by 1 and return to Step 2.

$$C = 53.$$

Reached terminal node. Note that $\mathrm{Cost}(T) = 57$.



Is this always work? (Rare-events)

Consider the "hair brush" tree T and suppose that the costs of all vertices are zero except for v_{n+1}, which has a cost of unity.

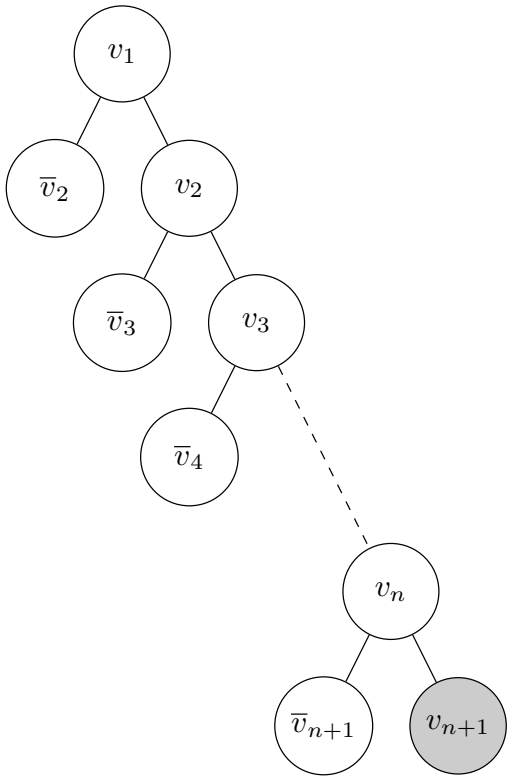


Figure: The hair brush tree.

The expectation and variance of the Knuth's estimator are

$$\mathbb{E}(C) = \frac{1}{2^n} \cdot 2^n \cdot 1 + \frac{2^n - 1}{2^n} \cdot D' \cdot 0 = 1,$$

and

$$\mathbb{E}\left(C^2\right) = \frac{1}{2^n} \cdot (2^n \cdot 1)^2 + \\ + \frac{2^n - 1}{2^n} \cdot \left(D' \cdot 0\right)^2 = 2^n \Rightarrow \\ \Rightarrow \operatorname{Var}(C) = 2^n - 1.$$

Is this always work? (Rare-events)

Consider the "hair brush" tree T and suppose that the costs of all vertices are zero except for v_{n+1}, which has a cost of unity.

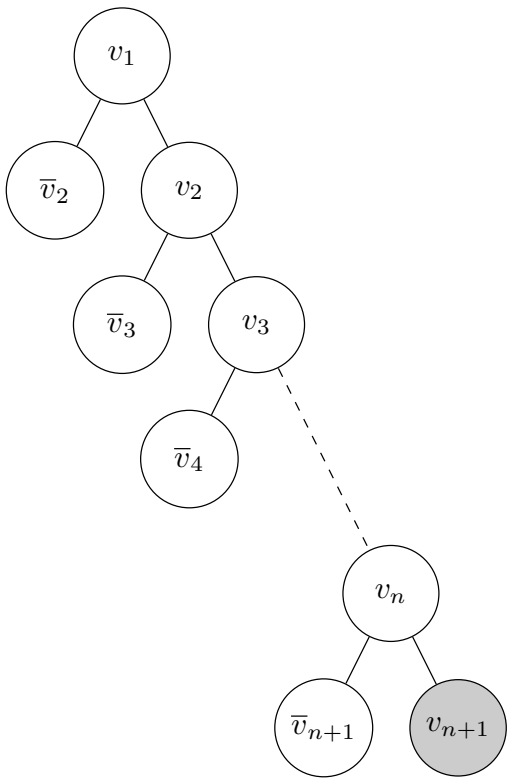


Figure: The hair brush tree.

The expectation and variance of the Knuth's estimator are

$$\mathbb{E}(C) = \frac{1}{2^n} \cdot 2^n \cdot 1 + \frac{2^n - 1}{2^n} \cdot D' \cdot 0 = 1,$$

and

$$\begin{aligned} \mathbb{E}\left(C^2\right) &= \frac{1}{2^n} \cdot (2^n \cdot 1)^2 + \\ &+ \frac{2^n - 1}{2^n} \cdot \left(D' \cdot 0\right)^2 = 2^n \Rightarrow \\ &\Rightarrow \operatorname{Var}(C) = 2^n - 1. \end{aligned}$$

$$\mathrm{CV}^2 = \frac{\operatorname{Var}(\mathbf{C})}{\mathbb{E}(\mathbf{C})^2} = \frac{\mathbf{2^n - 1}}{\mathbf{1^2}}$$

Is this always work? (Rare-events)

Consider the "hair brush" tree T and suppose that the costs of all vertices are zero except for v_{n+1}, which has a cost of unity.

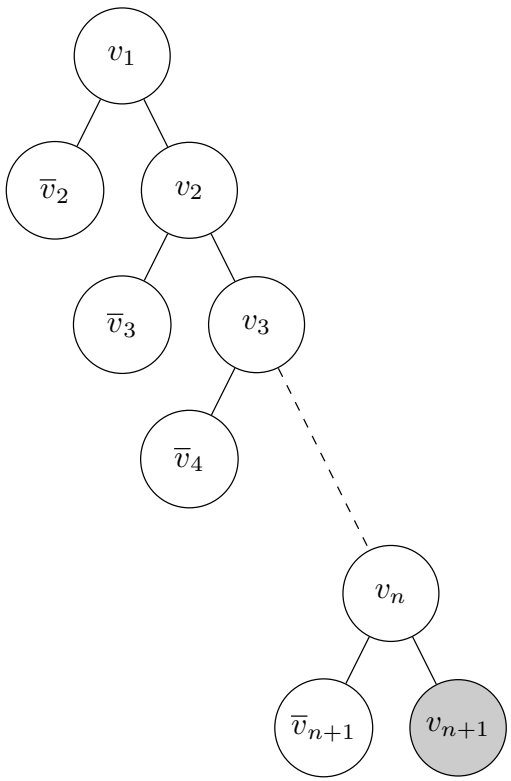


Figure: The hair brush tree.

The expectation and variance of the Knuth's estimator are

$$\mathbb{E}(C) = \frac{1}{2^n} \cdot 2^n \cdot 1 + \frac{2^n - 1}{2^n} \cdot D' \cdot 0 = 1,$$

and

$$\mathbb{E}\left(C^2\right) = \frac{1}{2^n} \cdot (2^n \cdot 1)^2 + \\ + \frac{2^n - 1}{2^n} \cdot \left(D' \cdot 0\right)^2 = 2^n \Rightarrow \\ \Rightarrow \operatorname{Var}(C) = 2^n - 1.$$

$$\mathrm{CV}^2 = \frac{\operatorname{Var}(\mathbf{C})}{\mathbb{E}(\mathbf{C})^2} = \frac{2^n - 1}{1^2}$$

What can we do?

The problem is the large variance.

What can we do?

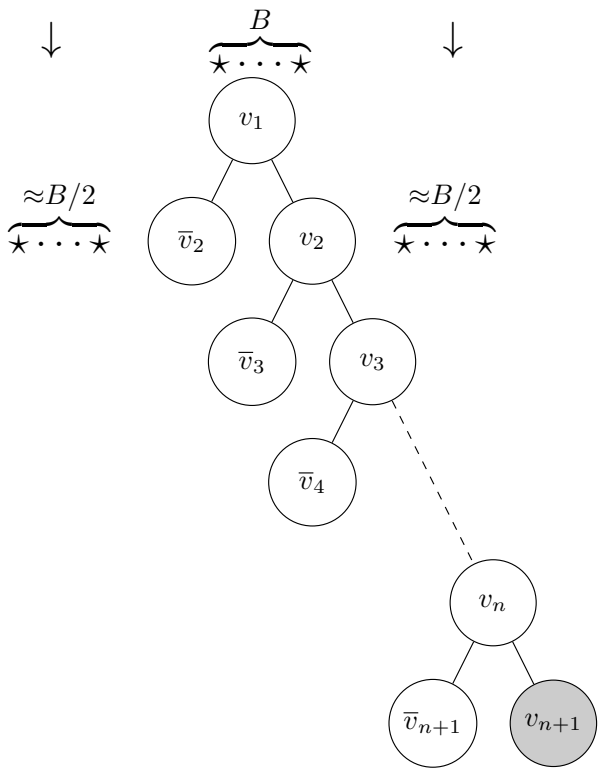
The problem is the large variance.

Variance reduction techniques.

- Common and antithetic random variables.
- Control variables.
- Conditional Monte Carlo.
- Stratified sampling.
- Importance Sampling.
- Multilevel Splitting.

To start with — Multilevel Splitting

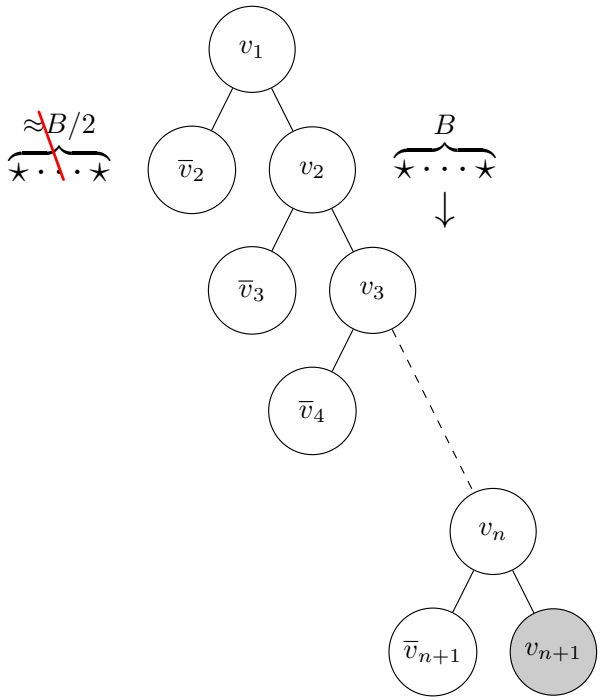
Consider (again) the "hair brush" tree T.



- Define some budget $B \geqslant 1$ of parallel random walks.
- Start from the root. The expected number of walks which reach the "good" vertex v_2 is $B/2$ — call them the "good" trajectories.

To start with — Multilevel Splitting

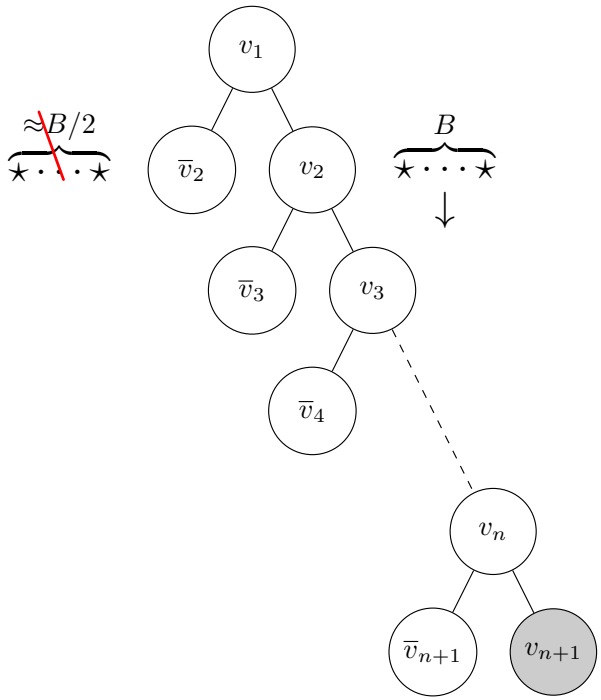
Consider (again) the "hair brush" tree T.



- Define some budget $B \geqslant 1$ of parallel random walks.
- Start from the root. The expected number of walks which reach the "good" vertex v_2 is $B/2$ — call them the "good" trajectories.
- Split the "good" trajectories such that there are B of them again and continue to the next tree level.

To start with — Multilevel Splitting

Consider (again) the "hair brush" tree T.



- Define some budget $B \geqslant 1$ of parallel random walks.
- Start from the root. The expected number of walks which reach the "good" vertex v_2 is $B/2$ — call them the "good" trajectories.
- Split the "good" trajectories such that there are B of them again and continue to the next tree level.
- Carefully choosing B (polynomial in $n!$), will allow us to reach the vertex of interest — v_{n+1} with reasonably high probability.

$$\mathbb{P}(\text{The process reaches the next level}) = 1 - 1/2^B.$$

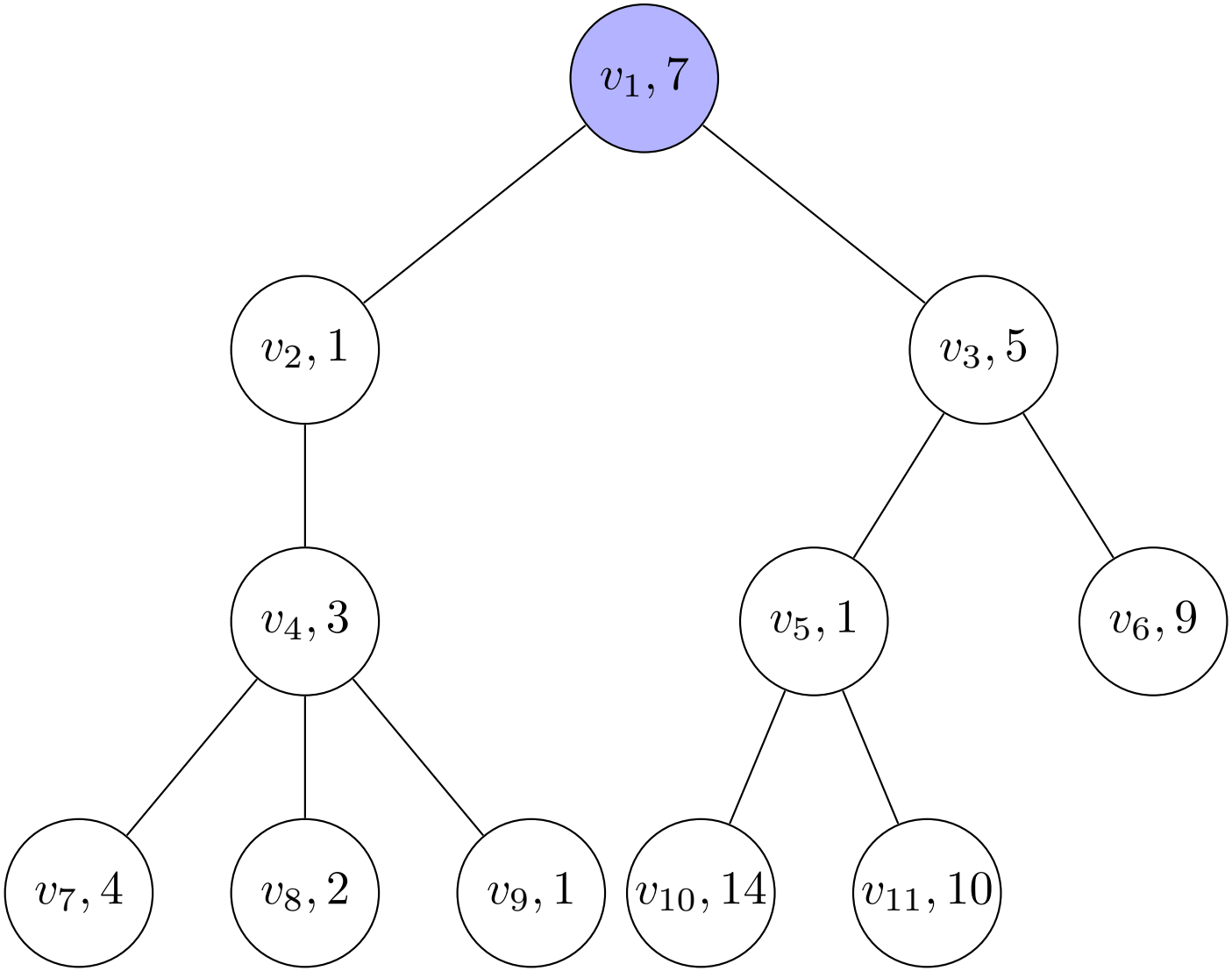
$$\mathbb{P}(\text{The process reaches the } v_{n+1} \text{ vertex}) = (1 - 1/2^B)^n.$$

$$B = \log_2(n) \Rightarrow \mathbb{P}(\text{The process reaches the } v_{n+1} \text{ vertex}) \to \mathrm{e}^{-1}, \quad \text{as } n \to \infty.$$

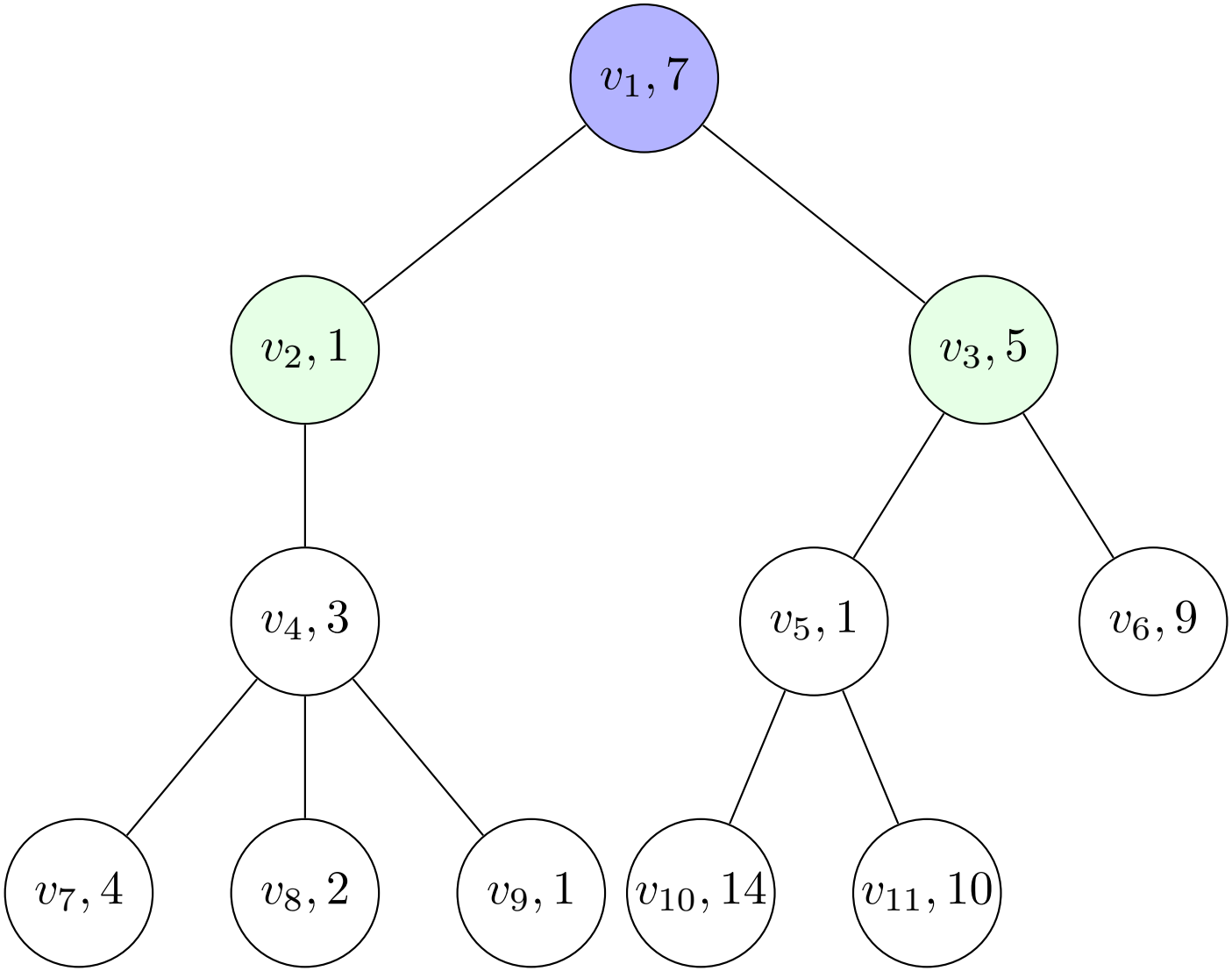
SE — the main idea

1. Define a budget $B \in \mathbb{N}$, and let B be the number of parallel random walks on the tree.
2. Using these B walks, run Knuth's Algorithm in parallel, (there are some technical issues!).
3. If some walks "die", split the remaining ones to continue with B walks as usual, (multilevel splitting).

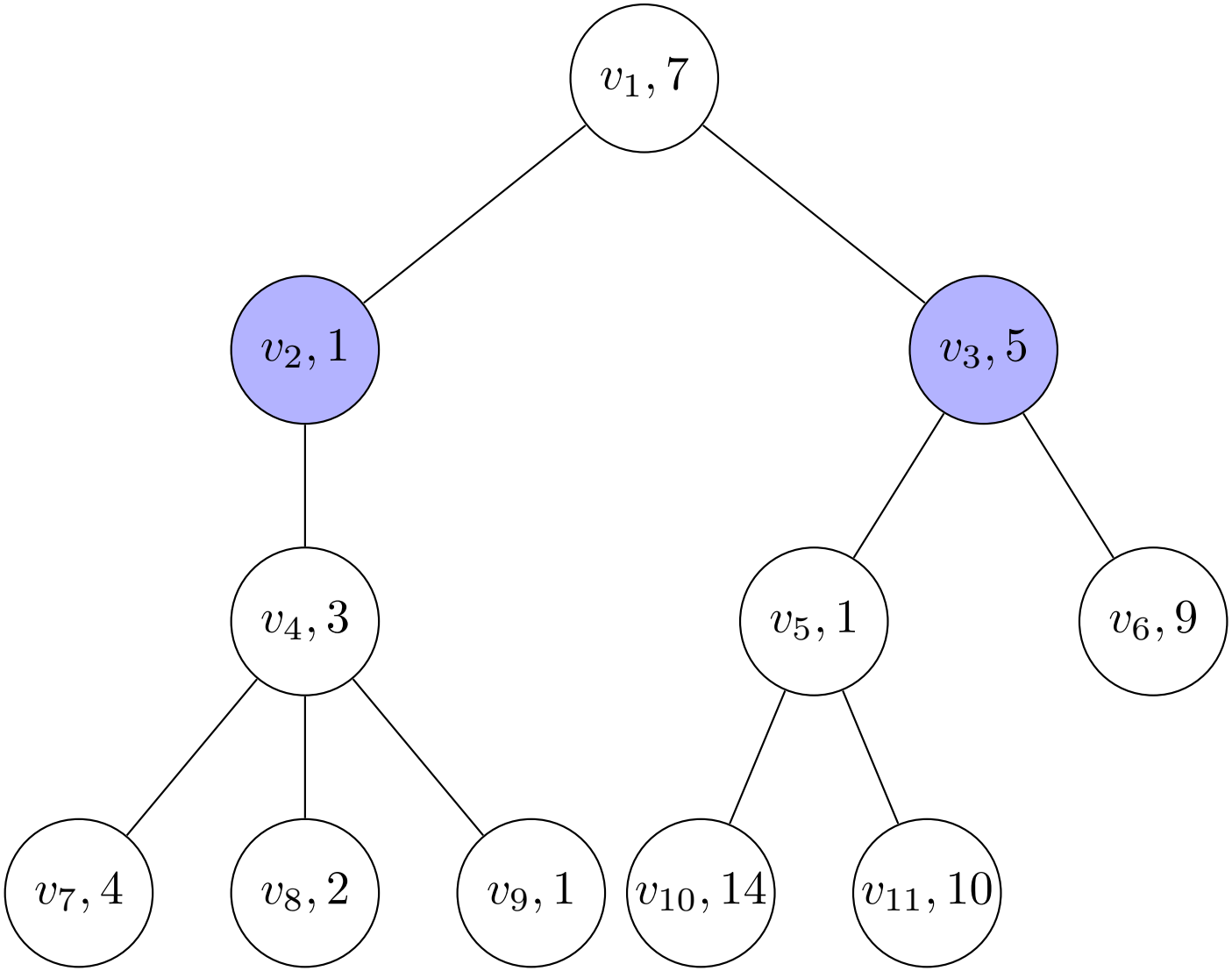
SE example with $B = 2$



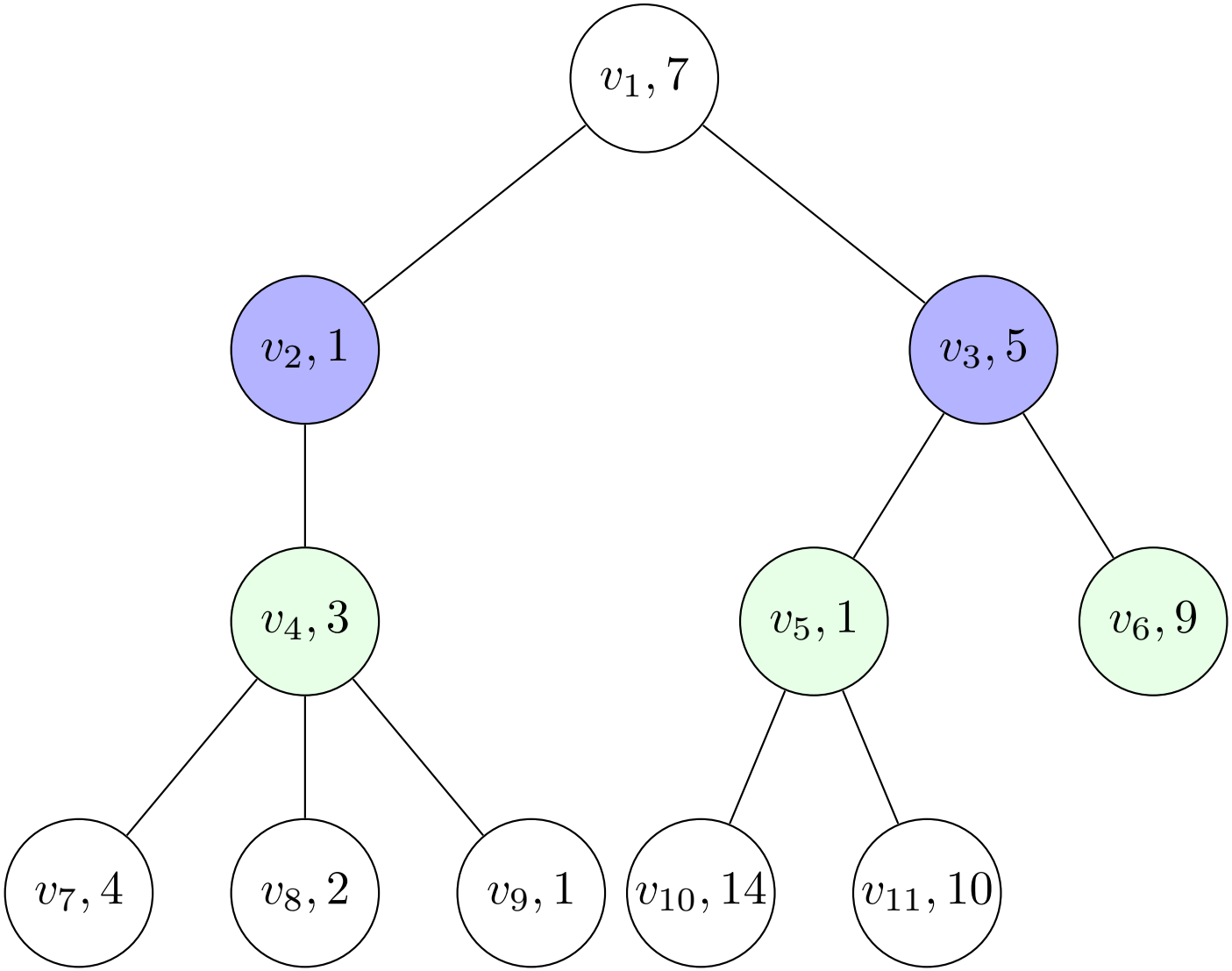
SE example with $B = 2$



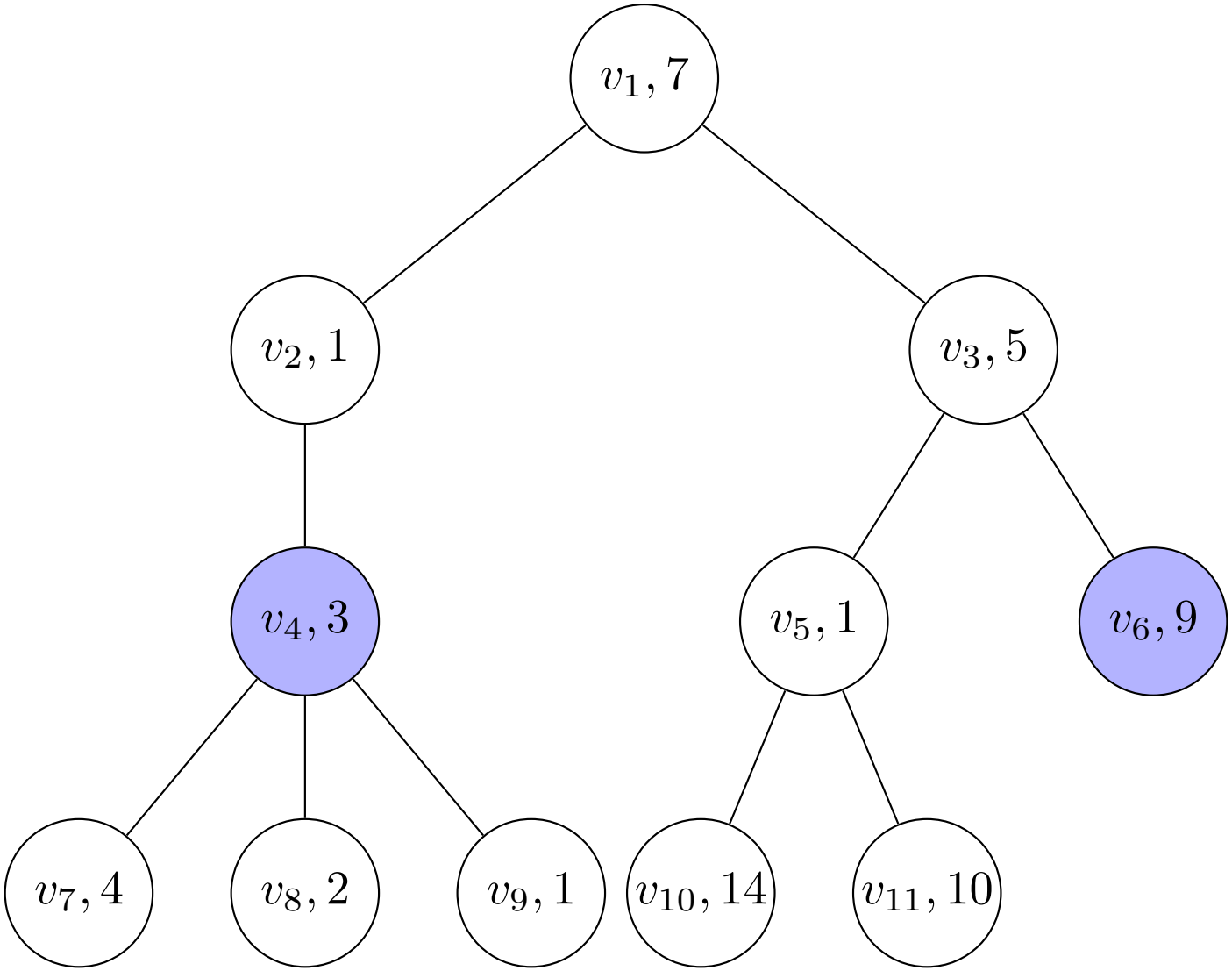
SE example with $B = 2$



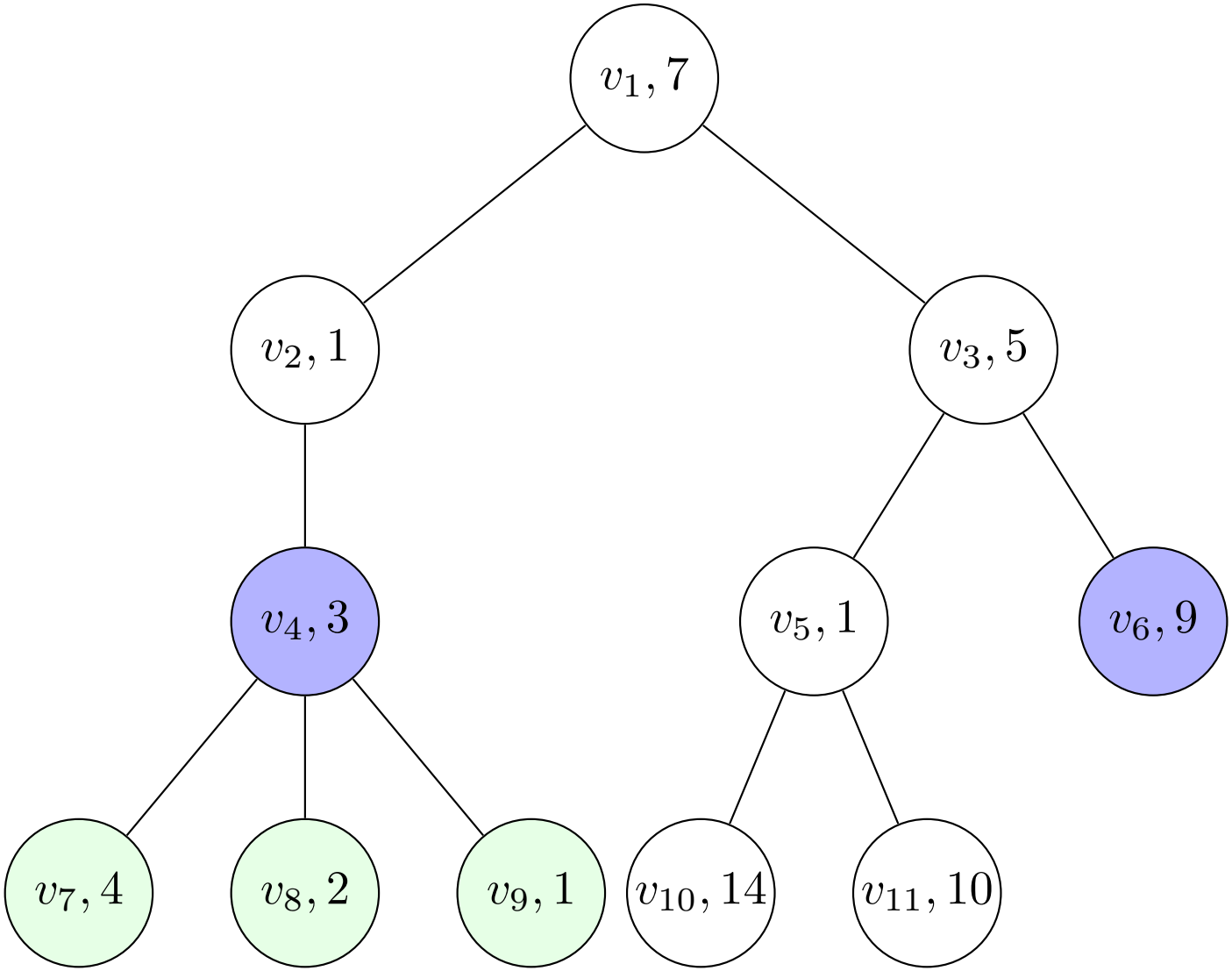
SE example with $B = 2$



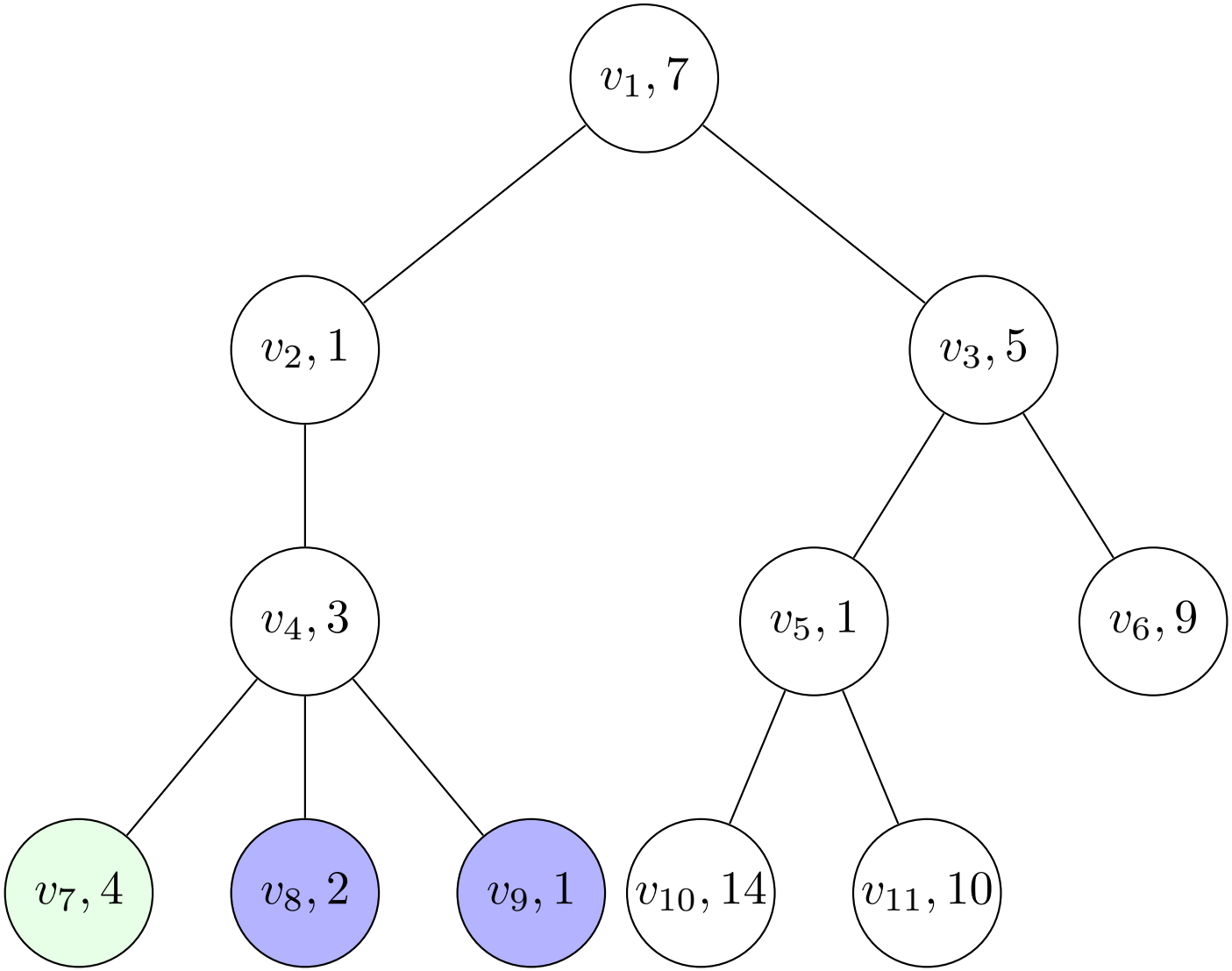
SE example with $B = 2$



SE example with $B = 2$



SE example with $B = 2$



SE Algorithm Variance for the hairbrush tree with $B = 2$

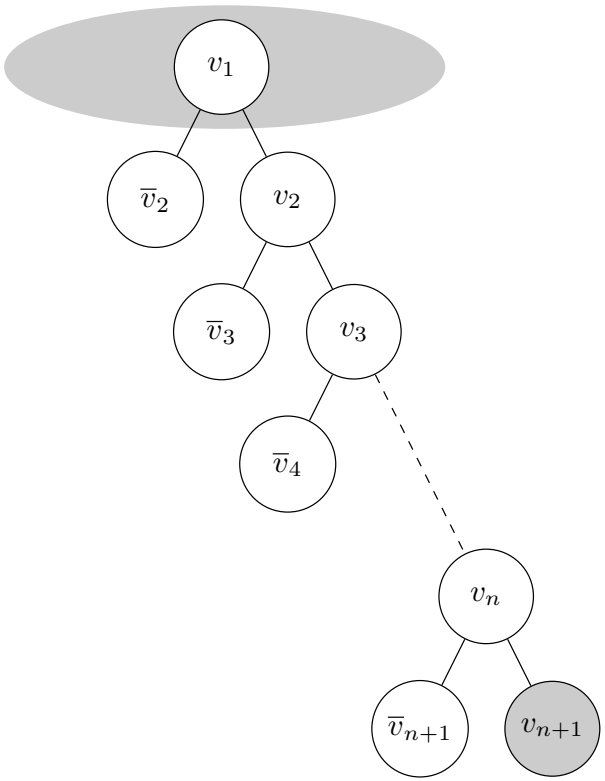


Figure: The hair brush tree.

SE Algorithm Variance for the hairbrush tree with $B = 2$

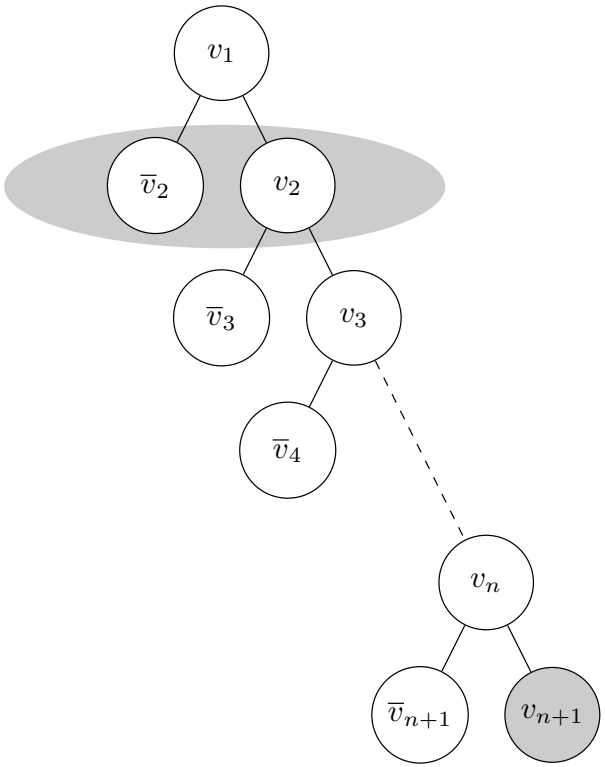


Figure: The hair brush tree.

SE Algorithm Variance for the hairbrush tree with $B = 2$

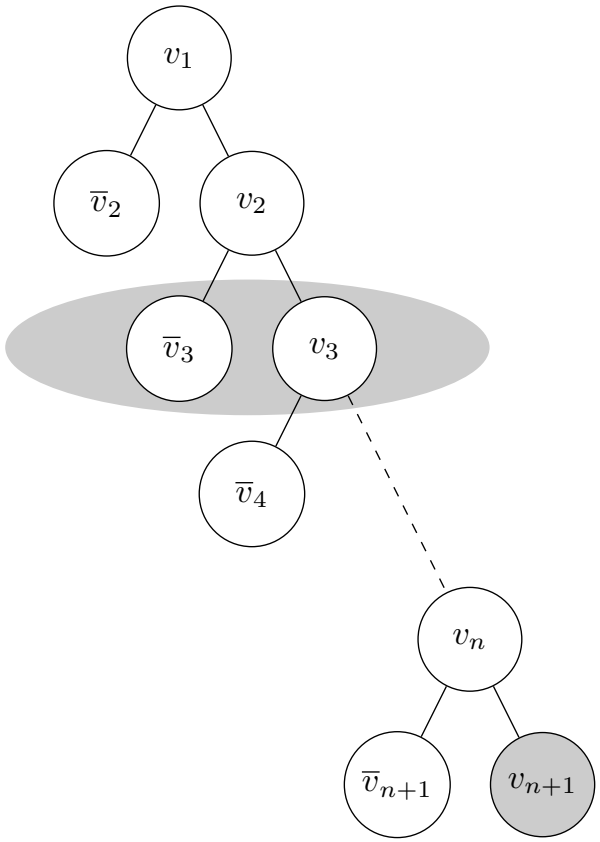


Figure: The hair brush tree.

SE Algorithm Variance for the hairbrush tree with $B = 2$

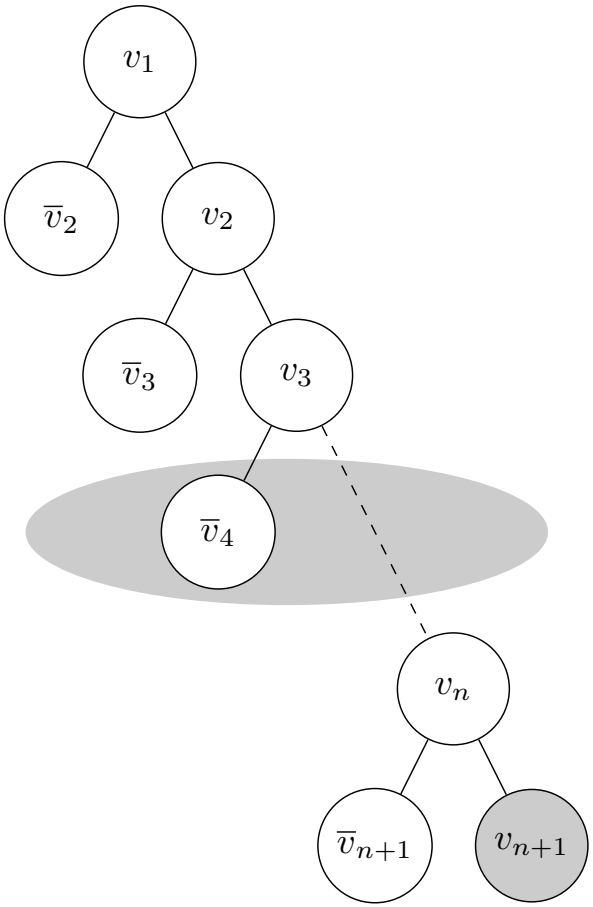


Figure: The hair brush tree.

SE Algorithm Variance for the hairbrush tree with $B = 2$

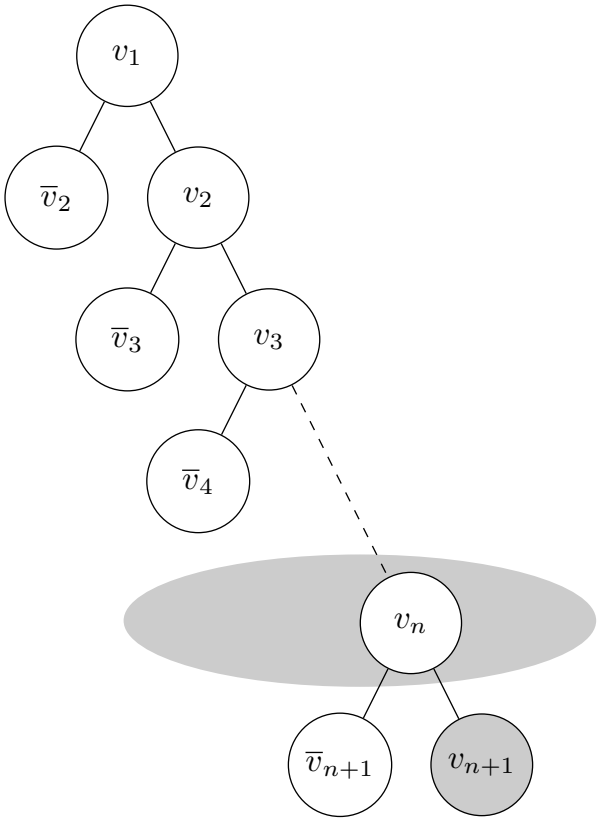


Figure: The hair brush tree.

SE Algorithm Variance for the hairbrush tree with $B = 2$

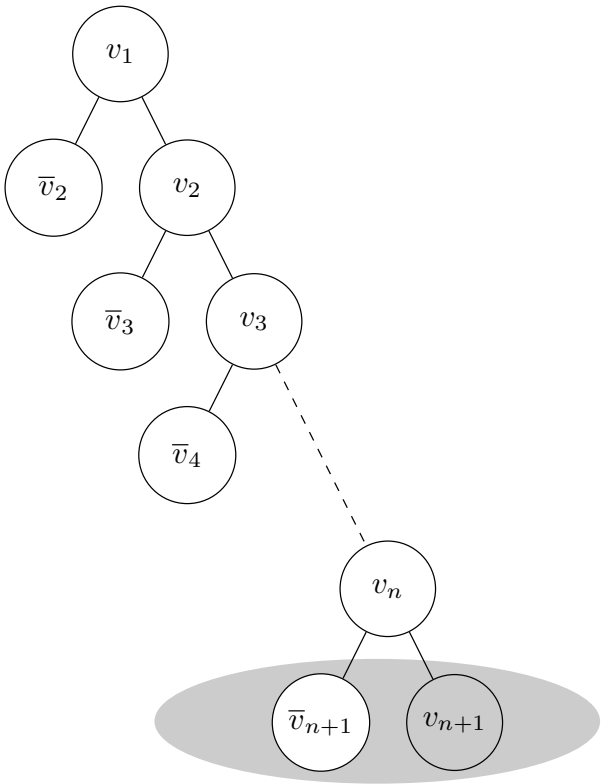


Figure: The hair brush tree.

The expectation and variance of the Knuth's estimator are

$$\mathbb{E}\left(C_{\mathrm{SE}}\right) = \underbrace{1}_{\mathbb{P}(\text{visit } v_{n+1})} \cdot \underbrace{2}_{D} \cdot \underbrace{\frac{1}{2}}_{\frac{c(\mathbf{x}_n)}{|\mathbf{x}_n|}} = 1,$$

and

$$\mathbb{E}\left(C_{\mathrm{SE}}^2\right) = 1 \cdot \left(2 \cdot \frac{1}{2}\right)^2 = 1 \;\Rightarrow$$

$$\Rightarrow \; \mathrm{Var}\left(C_{\mathrm{SE}}\right) = 0.$$

SE Algorithm Variance for the hairbrush tree with $B = 2$

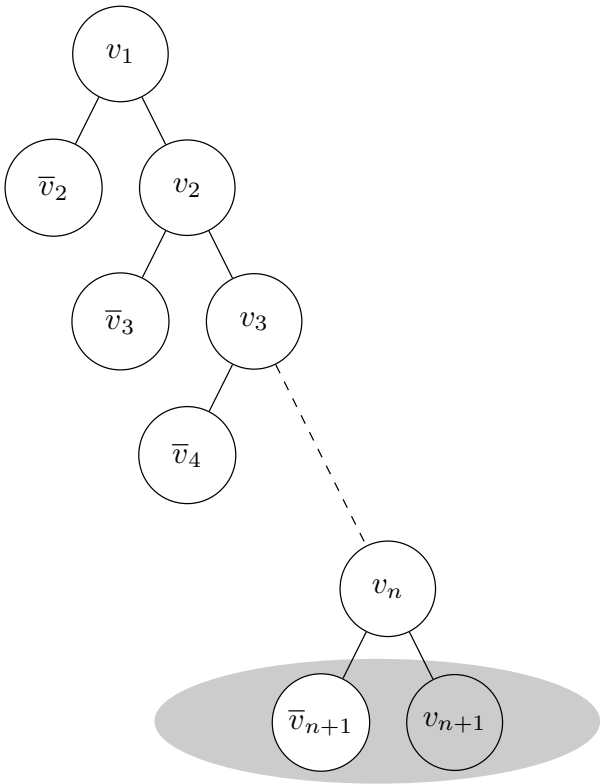


Figure: The hair brush tree.

The expectation and variance of the Knuth's estimator are

$$\mathbb{E}\left(C_{\mathrm{SE}}\right) = \underbrace{1}_{\mathbb{P}(\text{visit } v_{n+1})} \cdot \underbrace{2}_{D} \cdot \underbrace{\frac{1}{2}}_{\frac{c(\mathbf{x}_n)}{|\mathbf{x}_n|}} = 1,$$

and

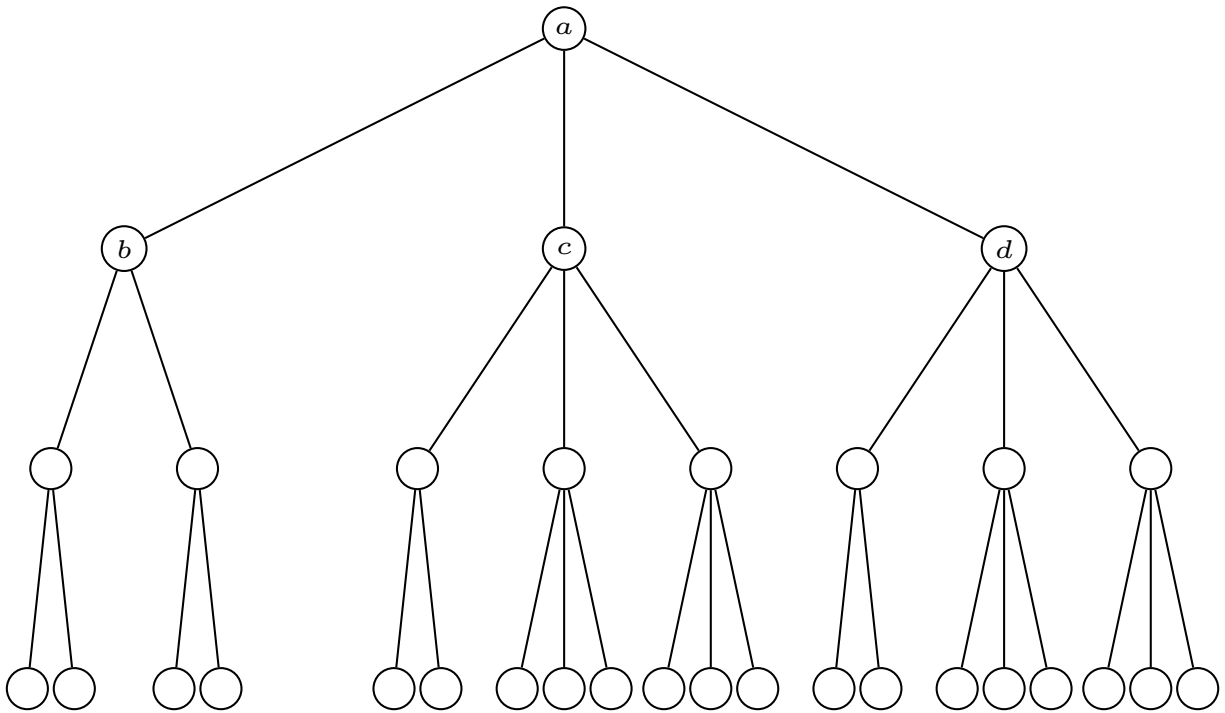
$$\mathbb{E}\left(C_{\mathrm{SE}}^2\right) = 1 \cdot \left(2 \cdot \frac{1}{2}\right)^2 = 1 \;\Rightarrow$$
$$\Rightarrow \; \mathrm{Var}\left(C_{\mathrm{SE}}\right) = 0.$$

$$\mathrm{CV}^2 = \frac{\mathrm{Var}\left(\mathbf{C}_{\mathrm{SE}}\right)}{\mathbb{E}\left(\mathbf{C}_{\mathrm{SE}}\right)^2} = \mathbf{0}.$$

Some encouraging numerical results (1)

Consider the following, very structured tree of height h. We define $c(v) = 1$ for all $v \in \mathcal{V}$.

- The root has 3 children.
- The leftmost child becomes the root of full binary tree and the rest of the children will continue the root behavior recursively.



Some encouraging numerical results (2)

- For Knuth's algorithm, the following holds:

$$\mathrm{CV}^2 \geqslant \frac{1.4^{h-1}}{16(h+1)^2}.$$

- Nevertheless, SE performance with $B = h$ is quite satisfactory.

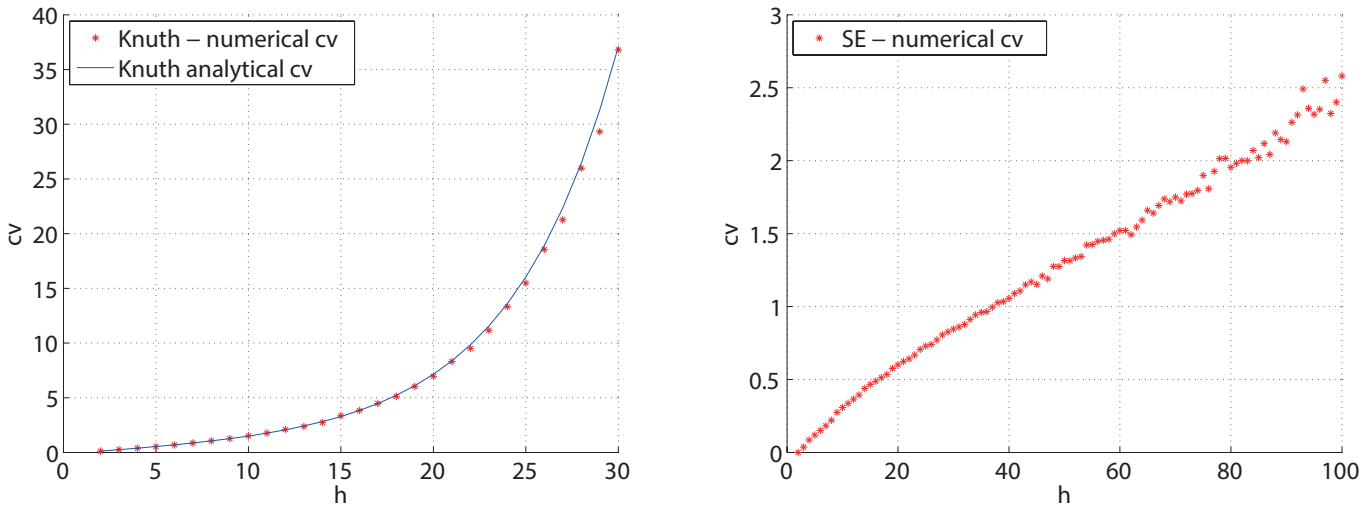


Figure: The performance of Knuth's Algorithm and the SE Algorithm on counting recursive trees of different heights. Left panel: Knuth. Right panel: SE.

Random trees

Definition (Family of random trees)

Consider a probability vector $\mathbf{p} = (p_0, \ldots, p_k)$ that corresponds to the probability of a vertex to have $0, \ldots, k$ successors respectively. Define a family of random trees $\mathcal{F}_{\mathbf{p}}^{h}$ as all possible trees of hight at most h that are generated using $\mathbf{p}$ up to the level h.

- The family $\mathcal{F}_{\mathbf{p}}^{h}$ is fully characterized by the probability vector $\mathbf{p}$ and the parameter h.
- The tree generation corresponds to a branching process.

Random trees

Definition (Family of random trees)

Consider a probability vector $\mathbf{p} = (p_0, \ldots, p_k)$ that corresponds to the probability of a vertex to have $0, \ldots, k$ successors respectively. Define a family of random trees $\mathcal{F}_{\mathbf{p}}^{h}$ as all possible trees of hight at most h that are generated using $\mathbf{p}$ up to the level h.

- The family $\mathcal{F}_{\mathbf{p}}^{h}$ is fully characterized by the probability vector $\mathbf{p}$ and the parameter h.
- The tree generation corresponds to a branching process.

Objective

Let $T = (\mathcal{V}, \mathcal{E})$ be a random tree from $\mathcal{F}_{\mathbf{p}}^{h}$. By assigning the cost $c(v) = 1$ for all $v \in \mathcal{V}$, the cost of the tree — $\mathrm{Cost}(T)$ is equal to $|\mathcal{V}|$. Our objective is to analyse the behavior of Knuth's and SE's estimators under this setting.

Super–critical branching process

- Consider a random tree rooted at v_0 and let R_m be the total number of children (population size) at level (generation) m and denote by M_m the total progeny at generation m. Define

$$\mu = \mathbb{E}(R_1) = \sum_{0 \leqslant j \leqslant k} j p_j \quad \text{and} \quad \sigma^2 = \mathrm{Var}(R_1) = \left(\sum_{0 \leqslant j \leqslant k} j^2 p_j \right) - \mu^2.$$

Super–critical branching process

- Consider a random tree rooted at v_0 and let R_m be the total number of children (population size) at level (generation) m and denote by M_m the total progeny at generation m. Define

$$\mu = \mathbb{E}(R_1) = \sum_{0 \leqslant j \leqslant k} j p_j \quad \text{and} \quad \sigma^2 = \mathrm{Var}(R_1) = \left(\sum_{0 \leqslant j \leqslant k} j^2 p_j \right) - \mu^2.$$

- It holds that

$$\mathbb{E}(M_m) = \mathbb{E}\left(1 + \sum_{1 \leqslant j \leqslant m} R_t\right) = \frac{1 - \mu^{m+1}}{1 - \mu}.$$

Random trees — expected performance

Theorem (Knuth v.s SE)

For a random tree $T^{(h)}$ the following holds.

1. *Lower bound on Knuth's expected variance satisfies:*

$$\mathbb{E}\left(\mathrm{Var}\left(C\left(T^{(h)}\right) \;\middle|\; T^{(h)}\right)\right) \geqslant \left(\sigma^2+\mu^2-\mu\right)\frac{1-\left(\sigma^2+\mu^2\right)^h}{1-\left(\sigma^2+\mu^2\right)}.$$

2. *For*

$$B \geqslant \max\left\{\left\lceil\frac{hk^2\ln\left(2h(\sigma^2+\mu^2)\frac{\sigma^2\mu}{(\mu-1)^3}\right)}{2(\mu-1)^2}\right\rceil,\ \left\lceil\frac{h\sigma^2}{\mu^2}\right\rceil\right\},$$

the upper bound on SE's expected variance satisfies:

$$\mathbb{E}\left(\mathrm{Var}\left(C_{\mathrm{SE}}\left(T^{(h)}\right) \;\middle|\; T^{(h)}\right)\right) \leqslant B^2 h\mathrm{e}\mu^{2h}\left(\frac{\sigma^2\mu}{(\mu-1)^3}+1\right).$$

The SE Algorithm introduces an expected variance reduction that is *approximately* equal to

$$\left(1+\frac{\sigma^2}{\mu^2}\right)^h.$$

How about the performance in practice?

We expect that variance reduction is governed by $\left(1 + \frac{\sigma^2}{\mu^2}\right)^h$ term.

We choose $\mathbf{p} = (0.3, 0.4, 0.1, 0.2)$, and $h = 60$. The true number of nodes is 1976527.

Knuth's performance is very bad.

$$\mathbf{p} = (0.3, 0.4, 0.1, 0.2) \Rightarrow \mu = 1.2, \sigma^2 = 2.6 \Rightarrow \left(1 + \frac{2.6}{1.2^2}\right)^{60} \approx 7.61 \cdot 10^{26}.$$

How about the performance in practice?

We expect that variance reduction is governed by $\left(1+\frac{\sigma^2}{\mu^2}\right)^h$ term.

We choose $\mathbf{p} = (0.3, 0.4, 0.1, 0.2)$, and $h = 60$. The true number of nodes is 1976527.

Knuth's performance is very bad.

$$\mathbf{p} = (0.3, 0.4, 0.1, 0.2) \;\Rightarrow\; \mu = 1.2, \sigma^2 = 2.6 \;\Rightarrow\; \left(1+\frac{2.6}{1.2^2}\right)^{60} \approx 7.61 \cdot 10^{26}.$$

Table: Knuth's Algorithm.

Run	$\widehat{C}$	$\widehat{\mathrm{RE}}$
1	3.06×10^3	6.09×10^{-1}
2	1.44×10^4	9.52×10^{-1}
3	1.05×10^3	2.40×10^{-1}
4	7.08×10^3	7.93×10^{-1}
5	3.01×10^3	6.08×10^{-1}
6	4.36×10^4	8.41×10^{-1}
7	3.26×10^3	5.36×10^{-1}
8	3.01×10^3	4.14×10^{-1}
9	1.51×10^3	2.65×10^{-1}
10	1.06×10^3	3.06×10^{-1}
Average	8.10×10^3	5.56×10^{-1}

Table: SE Algorithm

Run	$\widehat{C_{\mathrm{SE}}}$	$\widehat{\mathrm{RE}}$
1	2.04×10^6	5.03×10^{-2}
2	1.83×10^6	5.56×10^{-2}
3	1.99×10^6	7.18×10^{-2}
4	2.02×10^6	5.80×10^{-2}
5	1.90×10^6	5.97×10^{-2}
6	1.95×10^6	5.70×10^{-2}
7	2.03×10^6	6.38×10^{-2}
8	1.83×10^6	5.25×10^{-2}
9	2.14×10^6	6.88×10^{-2}
10	1.97×10^6	5.97×10^{-2}
Average	1.97×10^6	5.97×10^{-2}

Fully Polynomial Randomized Approximation Scheme

- A *randomized approximation scheme* for $\mathrm{Cost}(T)$ is a non-deterministic algorithm which, when given an input tree T and a real number $\varepsilon \in (0, 1)$, outputs a random variable $\mathcal{K}$ such that

$$\mathbb{P}\left((1-\varepsilon)\mathrm{Cost}(T) \leqslant \mathcal{K} \leqslant (1+\varepsilon)\mathrm{Cost}(T)\right) \geqslant \frac{3}{4}.$$

- Such a scheme is said to be fully polynomial if its execution time is bounded by some polynomial in the tree height and ε^{-1}.

Fully Polynomial Randomized Approximation Scheme

- A *randomized approximation scheme* for $\mathrm{Cost}(T)$ is a non-deterministic algorithm which, when given an input tree T and a real number $\varepsilon \in (0,1)$, outputs a random variable $\mathcal{K}$ such that

$$\mathbb{P}\left((1-\varepsilon)\mathrm{Cost}(T) \leqslant \mathcal{K} \leqslant (1+\varepsilon)\mathrm{Cost}(T)\right) \geqslant \frac{3}{4}.$$

- Such a scheme is said to be fully polynomial if its execution time is bounded by some polynomial in the tree height and ε^{-1}.

To prove FPRAS, it is generally enough to prove that the CV is bounded in a polynomial in problem input.

Random trees — FPRAS

Theorem (Almost sure FPRAS)

Let $\mathcal{F}^h_{\mathbf{p}'}$ be a family of random trees such that for $T \in \mathcal{F}^h_{\mathbf{p}'}$

$$\lim_{h\to\infty} \mathbb{P}\left(\mathrm{Cost}(T) < \frac{1}{\mathrm{P}(h)}\nu_h\right) = 0,$$

where $\mathrm{P}(h) > 0$ is some polynomial function in h and $\nu_h = \frac{1-\mu^{h+1}}{1-\mu}$ is the expected number of nodes. In other words, for most instances, (almost surely), the actual number of nodes is not much smaller than the expectation. Then, under the above condition, and provided that

$$\mu > 1 + \delta \quad \text{for any } \delta > 0,$$

the SE algorithm is FPRAS for most of the instances in $T \in \mathcal{F}^h_{\mathbf{p}'}$, that is,

$$\mathrm{CV}^2 = \frac{\mathrm{Var}\left(C_{\mathrm{SE}}(T) \mid T\right)}{\left(\mathbb{E}\left(C_{\mathrm{SE}}(T) \mid T\right)\right)^2}$$

is bounded by a polynomial in h with high probability.

What next?

- **(Hard)** — finding more classes of trees that can be efficiently handled by SE, (that is, show proven performance guarantees like for the random tree case).
- **(Not very hard)** — Adaptation of SE for estimation of general expression:

$$\mathbb{E}(S(\mathbf{x})).$$

- **(Easy)** — Extending different Sequential Monte Carlo algorithms with SE mechanism (splitting).
- **(???)** — Approximate Bayesian inference.
- **(???)** — Adaptation of SE for optimization.
- **(???)** — Introducing Importance Sampling to SE estimator.

Thank you