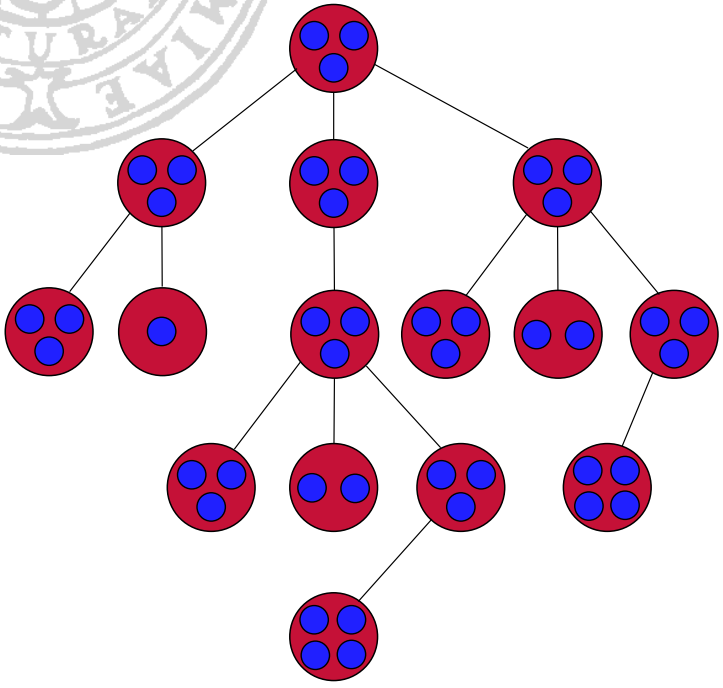
# Split trees–A unifying model for many important random trees of logarithmic height

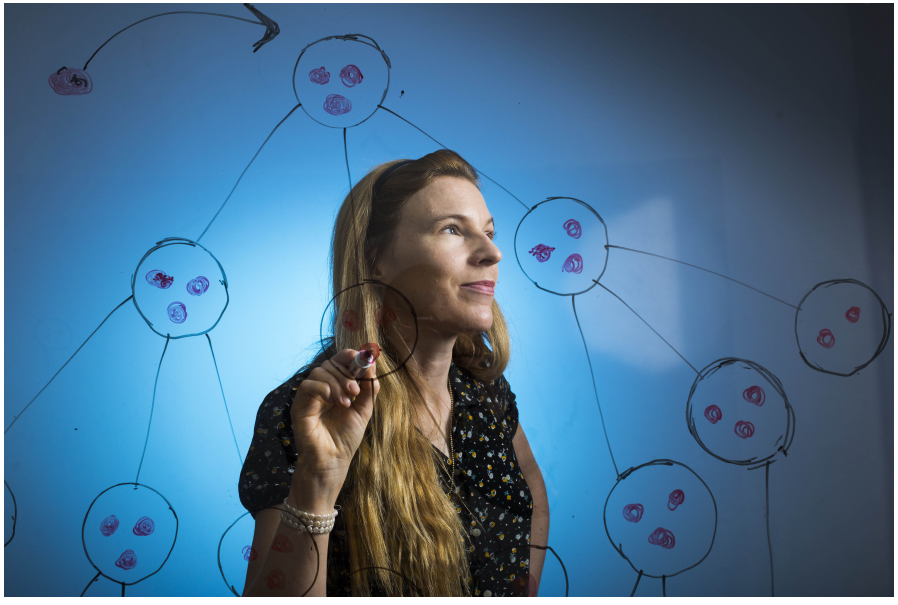
**Cecilia Holmgren**

**Department of Mathematics, Uppsala University, Sweden**

***NORCOM 2025, Reykjavik, Iceland***

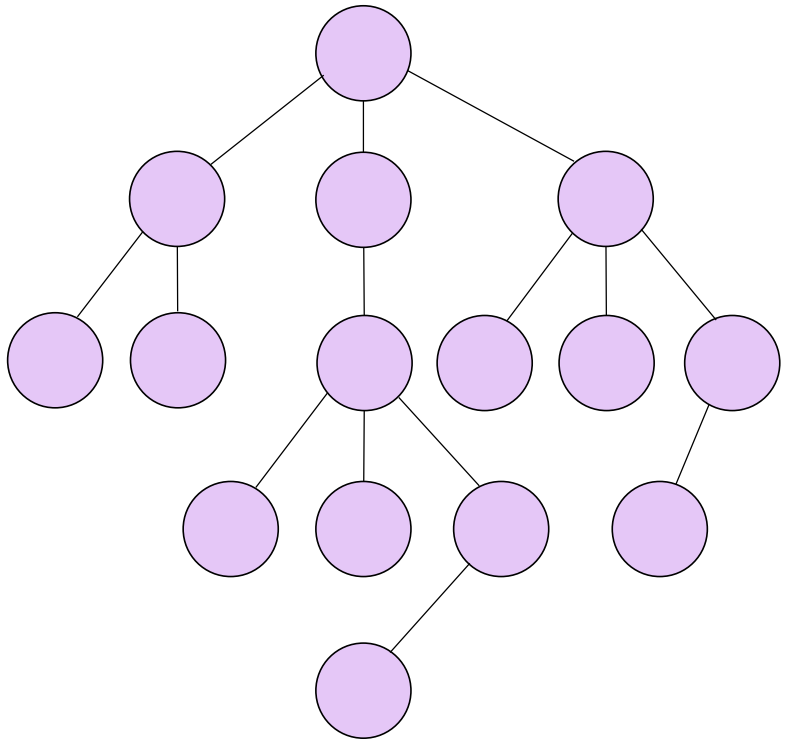
**June 17, 2025**





UPPSALA UNIVERSITET

# Rooted Trees

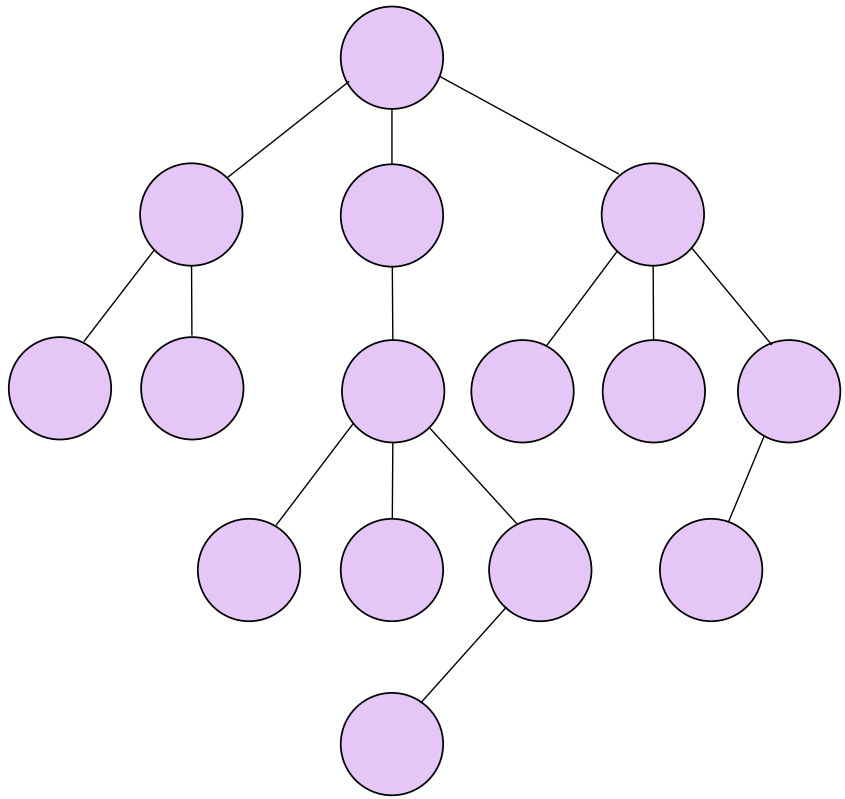


A *rooted tree T* is a connected graph such that:

- One node *r* is defined as the *root*
- There is a unique path of edges from each node *v* to the root *r*
- Thus, there are no cycles in the graph

UPPSALA UNIVERSITET

# Rooted Trees

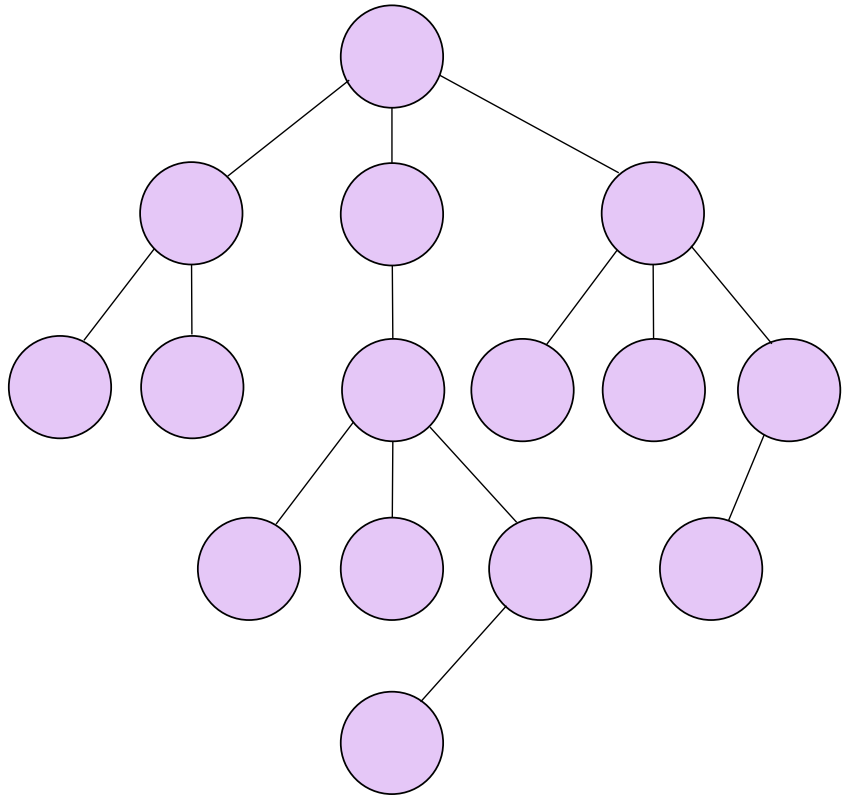


Thus, it is natural to use the following notations:

- The *children*, the *descendants* and the *ancestors* to a node $v$ in $T$ are defined in the same way as for a family tree

UPPSALA UNIVERSITET

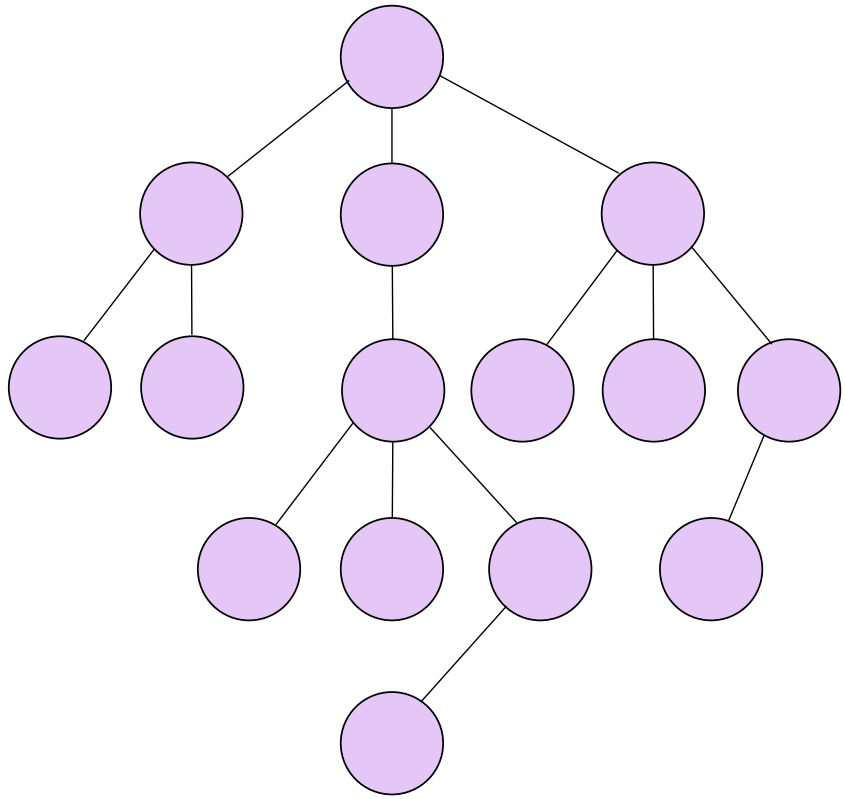
# Rooted Trees



- The *depth* $d(v)$ of a node $v$ in $T$ is the distance of $v$ to the root $r$
- The *height* is the maximal depth of the tree, i.e., $\max_{v \in T} d(v)$
- Nodes without any children are called *leaves*

UPPSALA UNIVERSITET

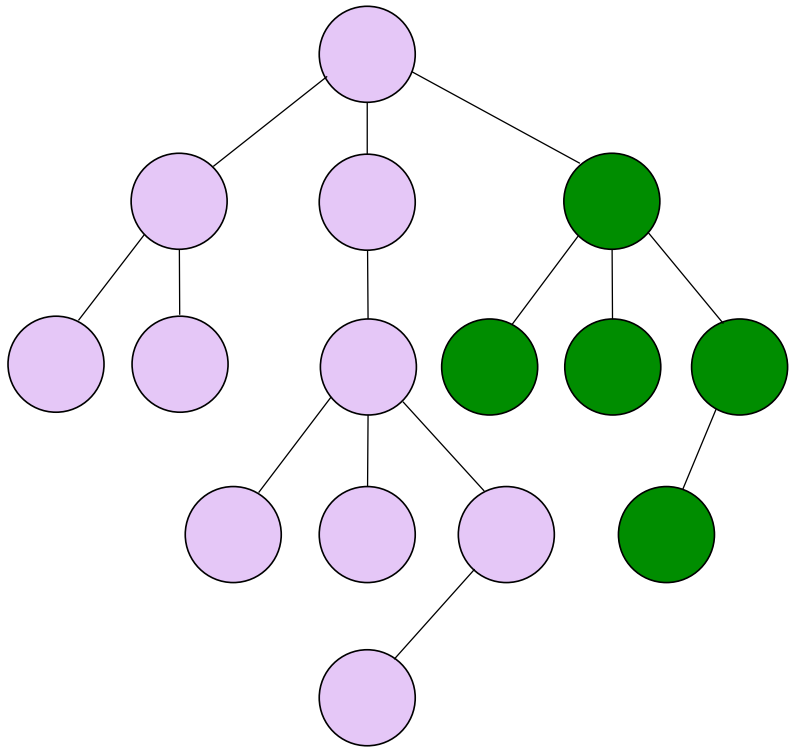
# Rooted Trees



- The number of nodes $n$ is often defined as the size of the tree
- Trees of logarithmic height, or equivalently $\mathcal{O}(\log n)$ trees, are classes of trees where the height for large $n$ is bounded by $C \log n$ for some constant $C$

UPPSALA UNIVERSITET

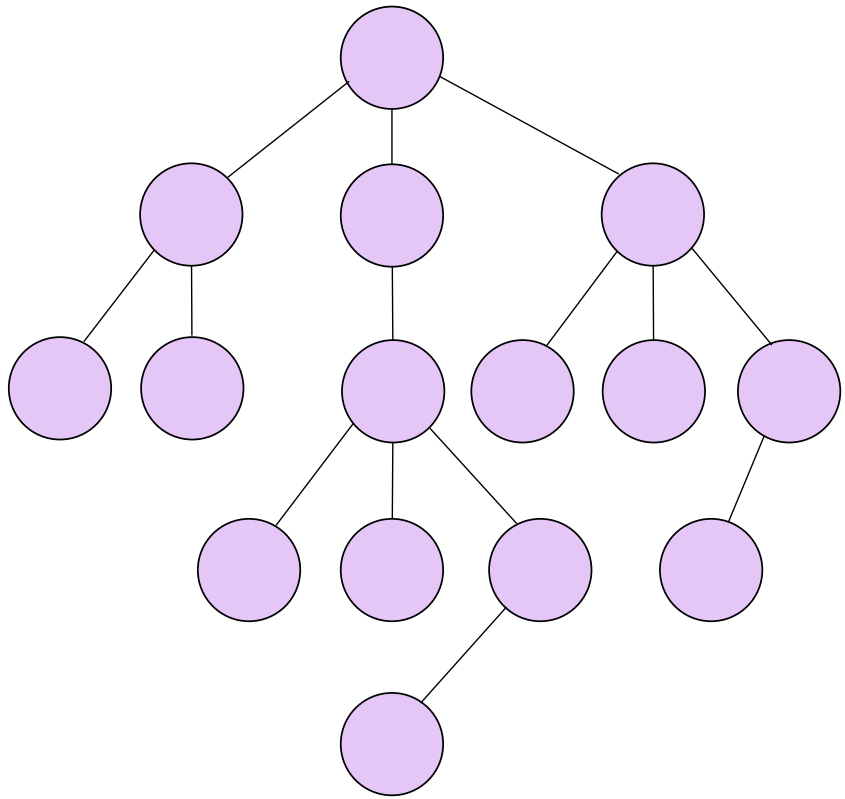
# Subtrees



- A *subtree* $T_v$ in a tree $T$ is a smaller tree inside $T$ that has as its root the node $v$ and contains all descendants of $v$
- The *subtree size* $n_v$ is then defined as the total number of nodes in the subtree $T_v$

UPPSALA UNIVERSITET

# Random Trees

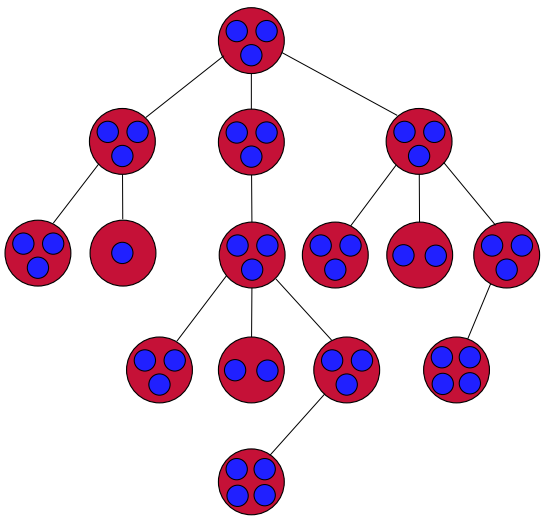


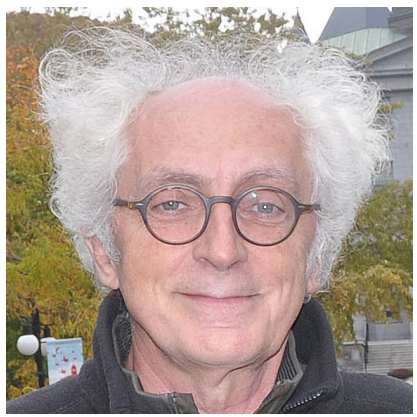
- A random tree is a tree that is generated from some probability procedure

UPPSALA UNIVERSITET

# Split Trees-A unifying model for many random trees of logarithmic height

- **Split trees** were introduced by Devroye (1998) as a novel approach for unifying many important random trees of logarithmic height. They are interesting not least because of their usefulness as models of **sorting algorithms** in computer science; for instance the well-known **Quicksort** algorithm (introduced by Hoare [1960]) can be depicted as a **binary search tree** (which is one example of a split tree)





*Cecilia Holmgren*

UPPSALA UNIVERSITET

# Split Trees-A unifying model for many random trees of logarithmic height

- In 2012, I introduced renewal theory as a novel approach for studying split trees[1]. This approach has proved to be highly useful for investigating such trees and has allowed me to show several general results valid for all split trees (instead of studying them one by one with different methods)
- In my presentation, I will introduce split trees and illustrate some of our results for this large class of random trees, e.g. on the size, total path length, number of cuttings and number of inversions as well as on the size of the giant and other components after bond percolation

[1] **Holmgren C.**, Novel characteristic of split trees by use of renewal theory. **Electronic Journal of Probability**, 2012.

UPPSALA UNIVERSITET

# Examples of Split Trees

- The class of split trees includes many important random trees of logarithmic height, e.g., **binary search trees**, **m-ary search trees**, **quadtrees**, **median of $(2k+1)$-trees**, **simplex trees** and **tries**

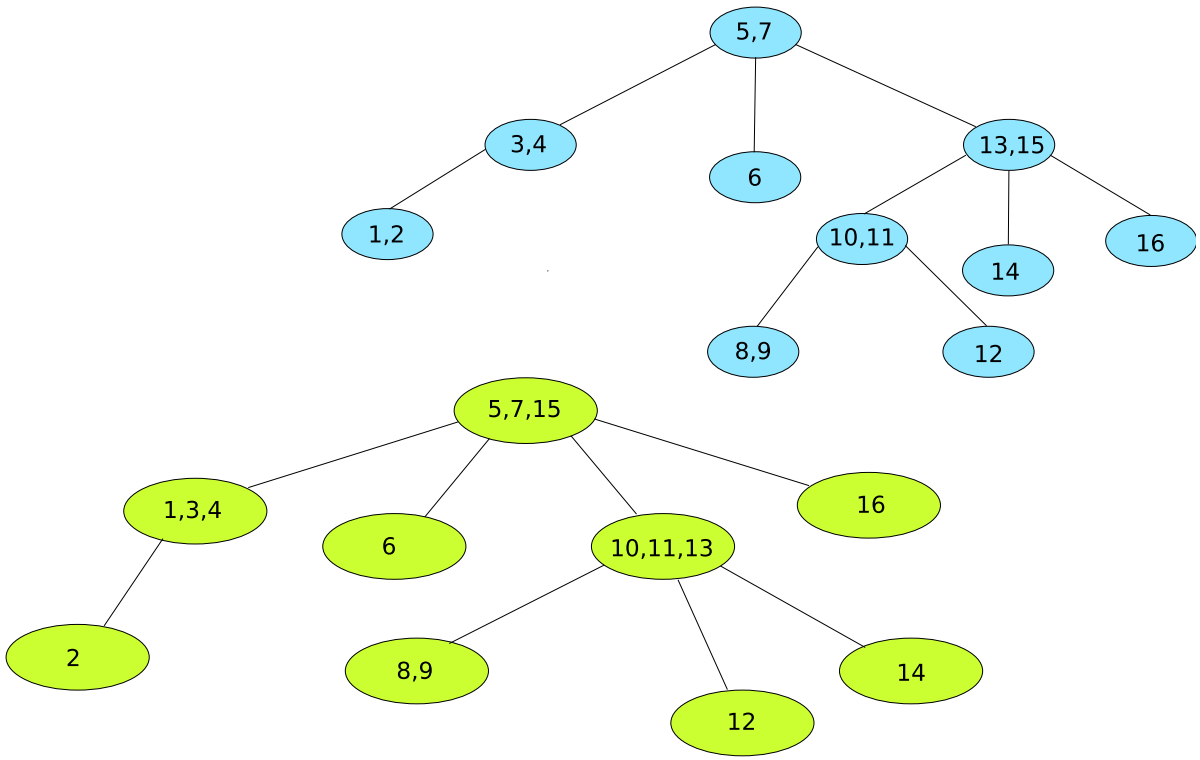
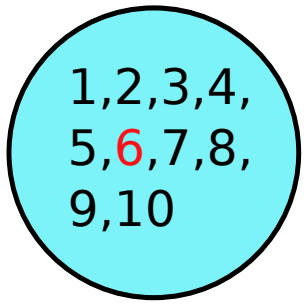



Figure: *A 3-ary and a 4-ary search tree.*

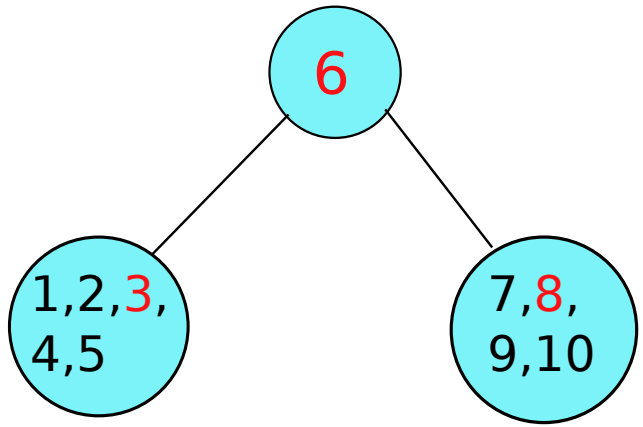
UPPSALA
UNIVERSITET

# The Binary Search Tree is an Example of a Split Tree



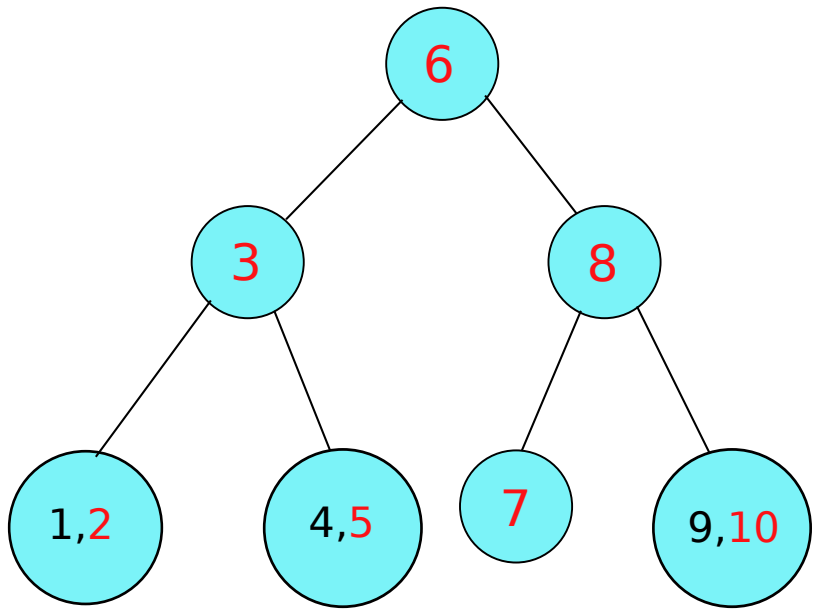
UPPSALA UNIVERSITET

# The Binary Search Tree is an Example of a Split Tree



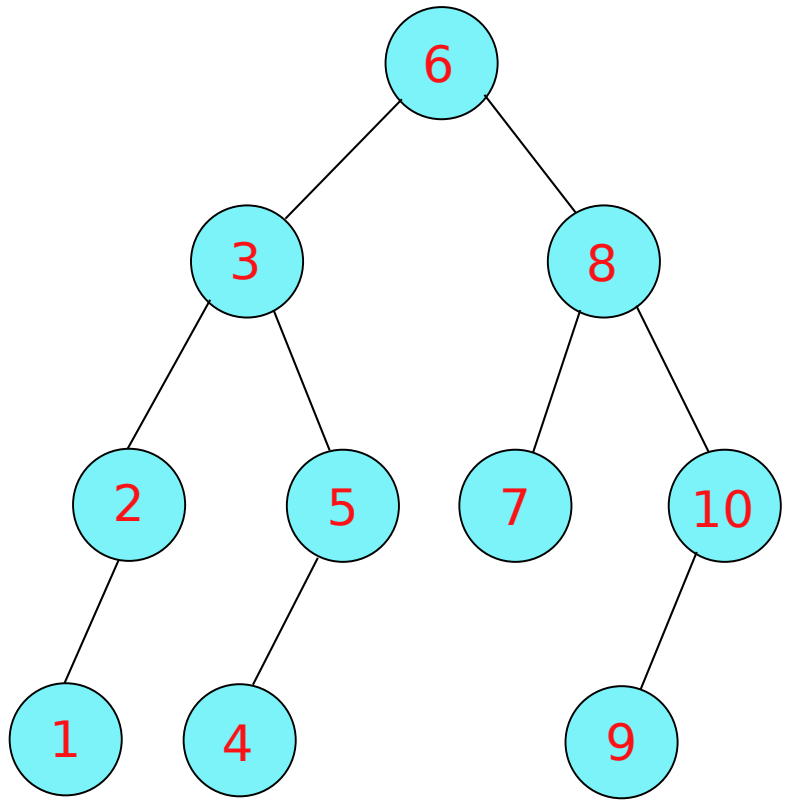
UPPSALA UNIVERSITET

# The Binary Search Tree is an Example of a Split Tree



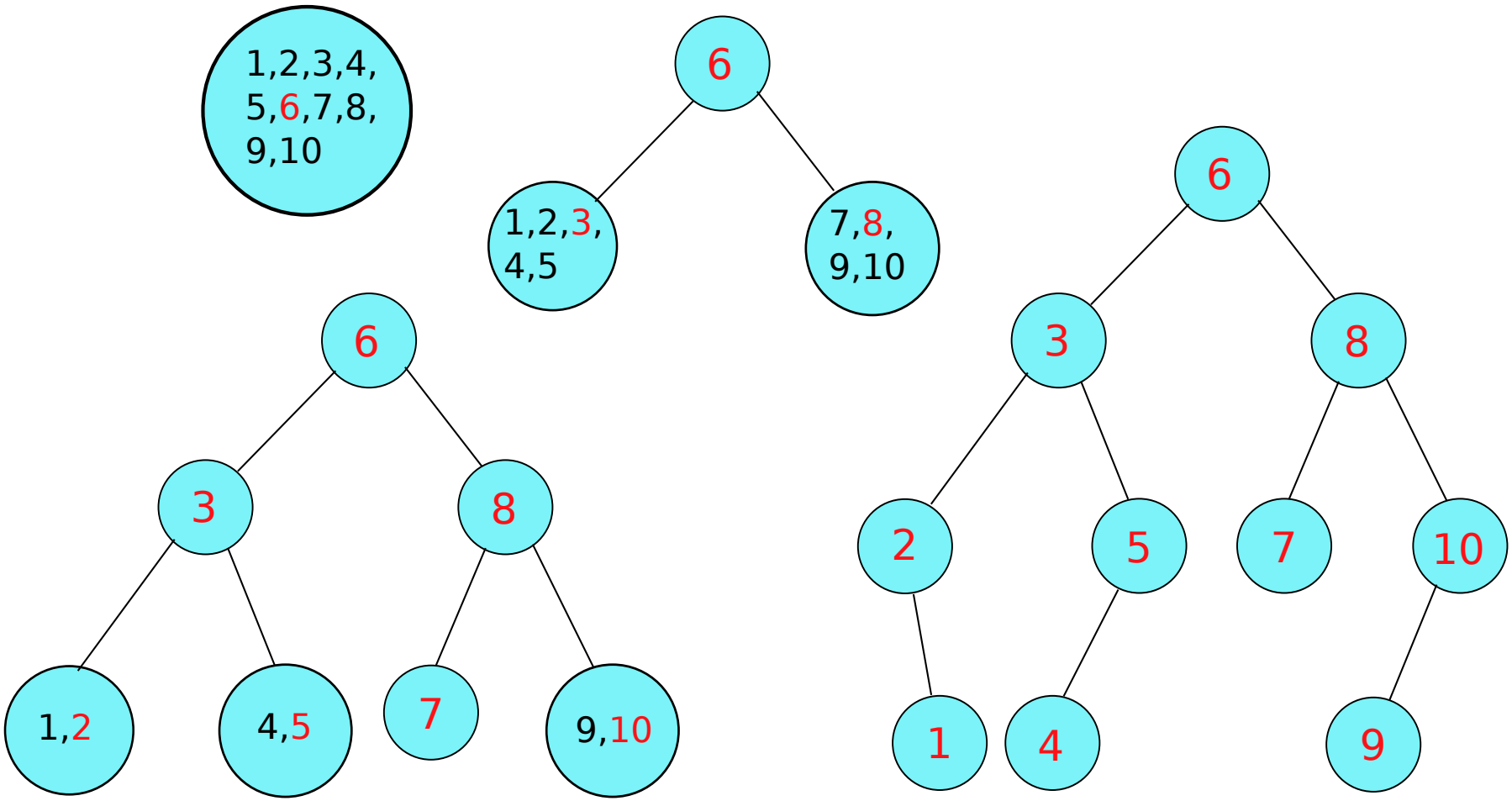
UPPSALA UNIVERSITET

# The Binary Search Tree is an Example of a Split Tree



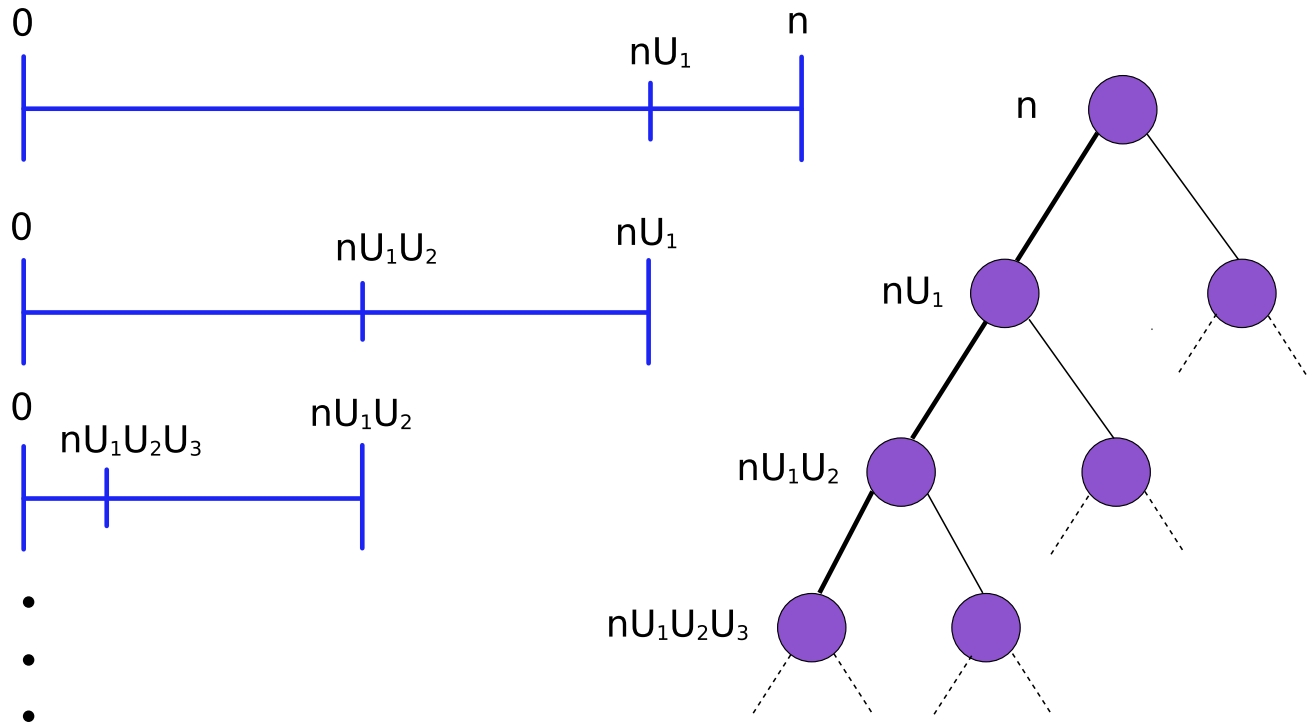
UPPSALA UNIVERSITET

# The Binary Search Tree is an Example of a Split Tree



UPPSALA UNIVERSITET

# The Binary Search Tree (continued)

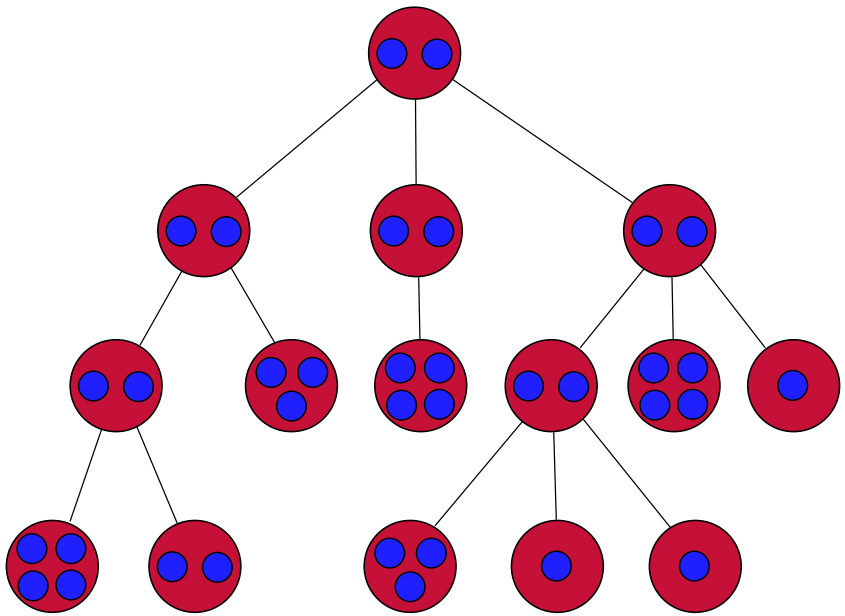



- Since the rank of the root's key is equally likely to be $\{1, 2, \ldots, n\}$, the size of its left subtree is distributed as the integer part of $nU$, where $U$ is a uniform $U(0, 1)$ random variable

UPPSALA UNIVERSITET

# Important Parameters for Split Trees

(Devroye 1998)



- Branch factor b
- Cardinality n
- Node capacity s>0
- Internal node capacity $s_0 \leq s$
- An independent copy of the random split vector $\mathcal{V} = (V_1, V_2, \ldots, V_b)$ is attached to each node.

Figure: A split tree with $n = 35$, $b = 3$, $s = 4$ and $s_0 = 2$. The split vector $\mathcal{V}$ is the most important parameter and its components are probabilities

The binary search tree has $b = 2$, $s = s_0 = 1$ and $\mathcal{V} = (U, 1 - U)$, where $U$ and $1 - U$ are $U(0, 1)$ random variables.


# Algorithmic Construction of Split Trees

- We consider an infinite $b$-ary tree, i.e., an infinite tree where each node has exactly $b$ children
- We view each node $u$ in such an infinite $b$-ary tree as a bucket which can hold at most $s$ balls
- Each node $u$ is assigned an independent copy $\mathcal{V}_u$ of the random split vector $\mathcal{V} = (V_1, \ldots, V_b)$ of probabilities that sum to 1. Thus, different nodes can get different vectors, but all the vectors $\mathcal{V}_u$ have the same distribution. For example in the binary search tree the $V_i$:s are distributed as uniform $U(0,1)$ random variables, but in other split trees the $V_i$:s could have other distributions

UPPSALA UNIVERSITET

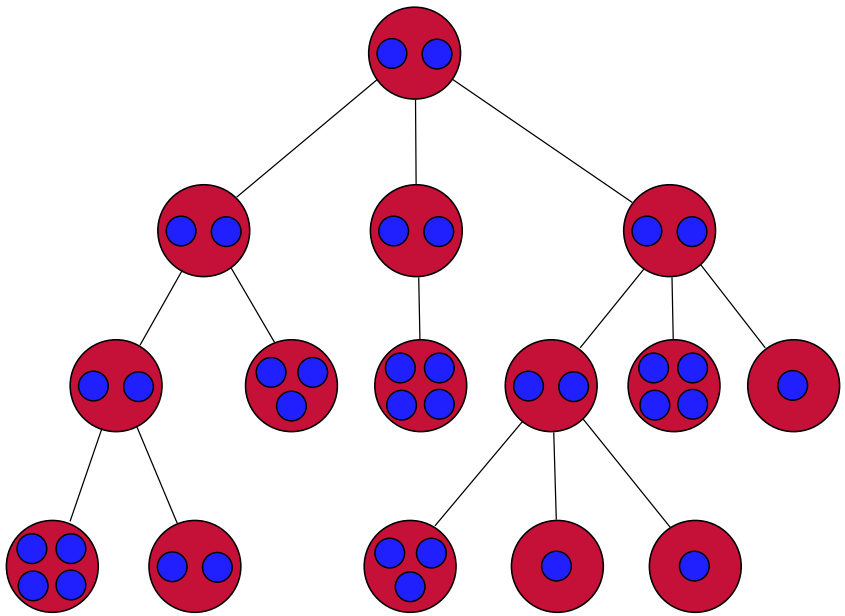
# Algorithmic Construction of Split Trees

- To construct the split tree we will add $n$ balls to the infinite $b$-ary tree one by one. Each ball starts in the root and fall down to an available descendant in the $b$ -ary tree which holds less than $s$ balls by using the probabilities in the split vectors to decide the falling-down procedure
- In each node $u$ that the ball falls to before it stops, it will use the split vector $\mathcal{V}_u = (V_1, \ldots, V_b)$ of $u$ so that it continues from $u$ to child $i \in \{1, \ldots, b\}$ of $u$ with probability $V_i$
- If a ball falls to a descendant $u$ with exactly $s$ balls the node is already full and has to split. This is done by keeping $s_0$ balls in $u$ and sending the other $s + 1 - s_0$ balls to the children of $u$ by using the the split vector $\mathcal{V}_u = (V_1, \ldots, V_b)$ of $u$
- Thus, each ball that is added ends up in some available descendant with less than $s$ balls further down in the tree

UPPSALA UNIVERSITET

# Important Parameters for Split Trees

(Devroye 1998)



- Branch factor b
- Cardinality n
- Node capacity s>0
- Internal node capacity $s_0 \leq s$
- An independent copy of the random split vector $\mathcal{V} = (V_1, V_2, \ldots, V_b)$ is attached to each node.

Figure: A split tree with $n = 35$, $b = 3$, $s = 4$ and $s_0 = 2$. The split vector $\mathcal{V}$ is the most important parameter and its components are probabilities

The binary search tree has $b = 2$, $s = s_0 = 1$ and $\mathcal{V} = (U, 1 - U)$, where $U$ and $1 - U$ are $U(0, 1)$ random variables.

UPPSALA UNIVERSITET

# Split Trees

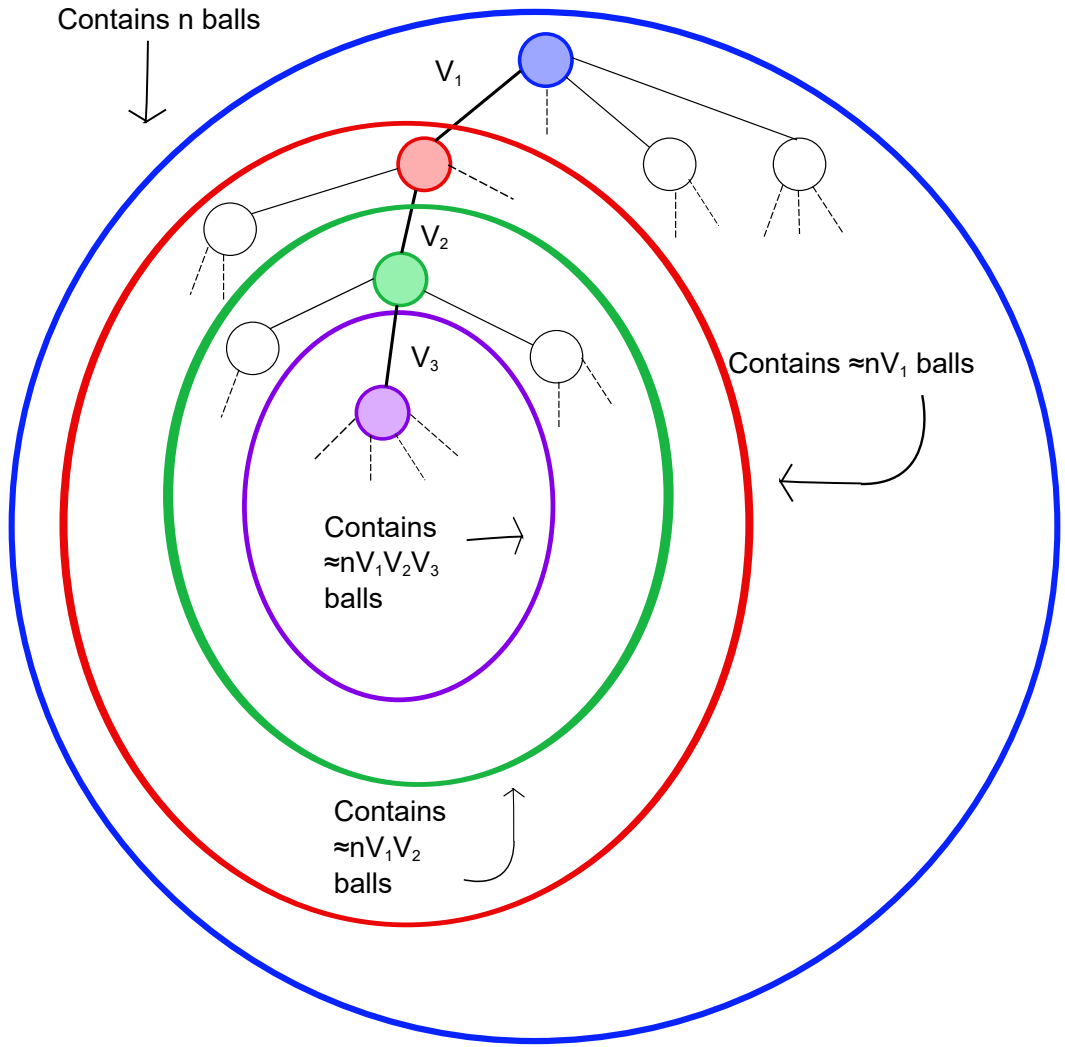



Figure: *Given all split vectors in the tree, $n_v$ for $v$ at depth $d$ is close to $nL_v = n\prod_{j=1}^{d} V_j$, where the $V_j$'s are i.i.d. random variables*

UPPSALA UNIVERSITET

# Split Trees: Most Nodes Close to Depth $\mu^{-1} \ln n$

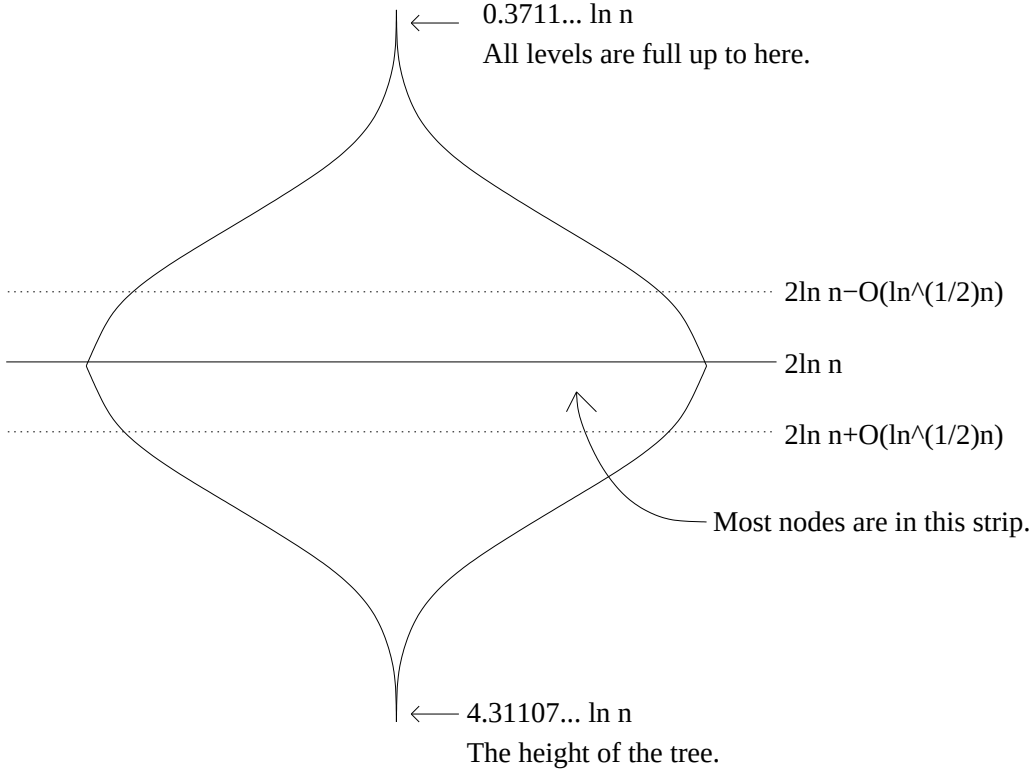



Figure: *The central limit theorem holds for the depths of the nodes. Most nodes are in a strip of width $\mathcal{O}(\sqrt{\ln n})$ around the depth $\mu^{-1} \ln n$, where $\mu := b\mathbf{E}(-V_1 \ln V_1)$. The figure shows the distribution of the nodes in a binary search tree*

UPPSALA UNIVERSITET

# Most Nodes Close to Depth $\mu^{-1} \ln n$

## Theorem (L. Devroye, *Siam J. Comput.* 1998)

*For the depth of the last ball $D_n$ it holds that*

$$\frac{D_n - \mu^{-1} \ln n}{\sqrt{\sigma^2 \mu^{-3} \ln n}} \xrightarrow{d} N(0,1).$$

## Proposition (C. Holmgren, *Electr. Journ. Probab.* 2012)

For the depth of the $k$:th ball $D_k$ such that $\frac{n}{\ln n} \leq k \leq n$ it holds that

$$\frac{D_k - \mu^{-1} \ln n}{\sqrt{\sigma^2 \mu^{-3} \ln n}} \xrightarrow{d} N(0,1).$$

Hence, for the average depth $D_n^* = \sum_{k=1}^{n} \frac{D_k}{n}$ it holds that

$$\frac{D_{n^*} - \mu^{-1} \ln n}{\sqrt{\sigma^2 \mu^{-3} \ln n}} \xrightarrow{d} N(0,1).$$

UPPSALA UNIVERSITET

# Split Trees: Most Nodes Close to Depth $\mu^{-1} \ln n$

For a given $\epsilon > 0$ we say that a node $v$ in $T^n$ is "good" if for its depth $d(v)$ it holds

$$\left| d(v) - \mu^{-1} \ln n \right| \leq \ln^{0.5+\epsilon} n,$$

and "bad" otherwise.

**Theorem** (C. Holmgren, *Electr. Journ. Probab.* 2012)

*For any choice of $\epsilon > 0$, the expected number of bad nodes in $T^n$ is bounded by $\mathcal{O}\left(\frac{n}{\ln^k n}\right)$ for any constant $k$.*

**Corollary** (C. Holmgren, *Electr. Journ. Probab.* 2012)

For any constant $r$ there is a constant $C > 0$ so that the expected number of nodes with depth $d(v) \geq C \ln n$ is bounded by $\mathcal{O}\left(\frac{1}{n^r}\right)$.

UPPSALA UNIVERSITET

# Split Trees: Most Nodes Close to Depth $\mu^{-1} \ln n$

**Theorem** (C. Holmgren, *Electr. Journ. Probab.* 2012)

*For the expected value of the depth of the last ball $D_n$ it holds that*

$$\frac{\mathbf{E}(D_n) - \mu^{-1} \ln n}{\sqrt{\ln n}} \longrightarrow 0,$$

*and the same result holds for the average depth $D_n^*$ i.e.*

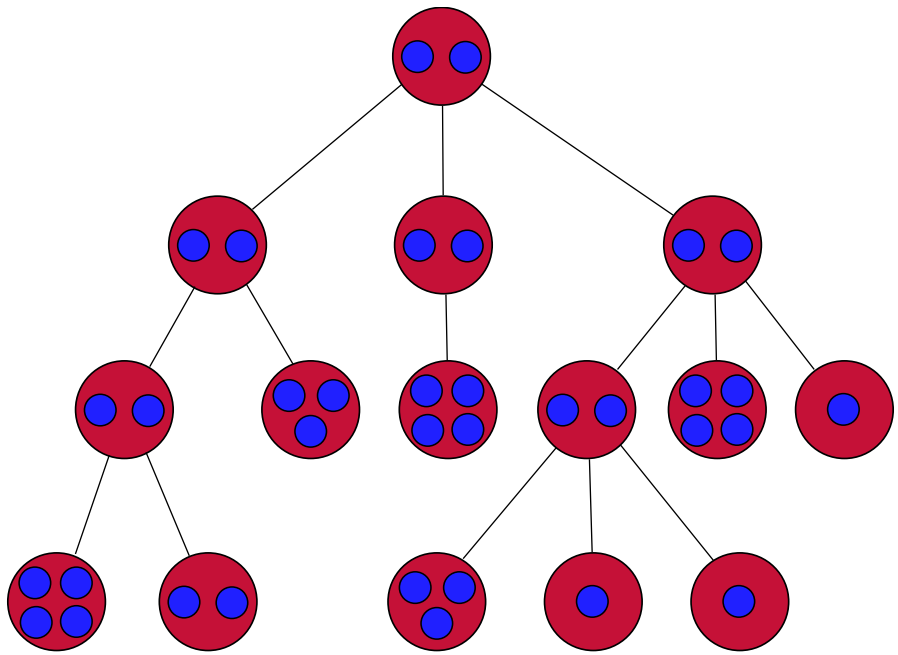
$$\frac{\mathbf{E}(D_n^*) - \mu^{-1} \ln n}{\sqrt{\ln n}} \longrightarrow 0.$$

*Furthermore, for the depth of the k:th ball $D_k$ such that $\frac{n}{\ln n} \leq k \leq n$ it holds that*

$$\frac{\mathbf{Var}(D_k)}{\ln n} \longrightarrow \sigma^2 \mu^{-3}.$$

UPPSALA
UNIVERSITET

# Using Renewal Theory to Study Split Trees



*Cecilia Holmgren*

UPPSALA UNIVERSITET

# Split Trees

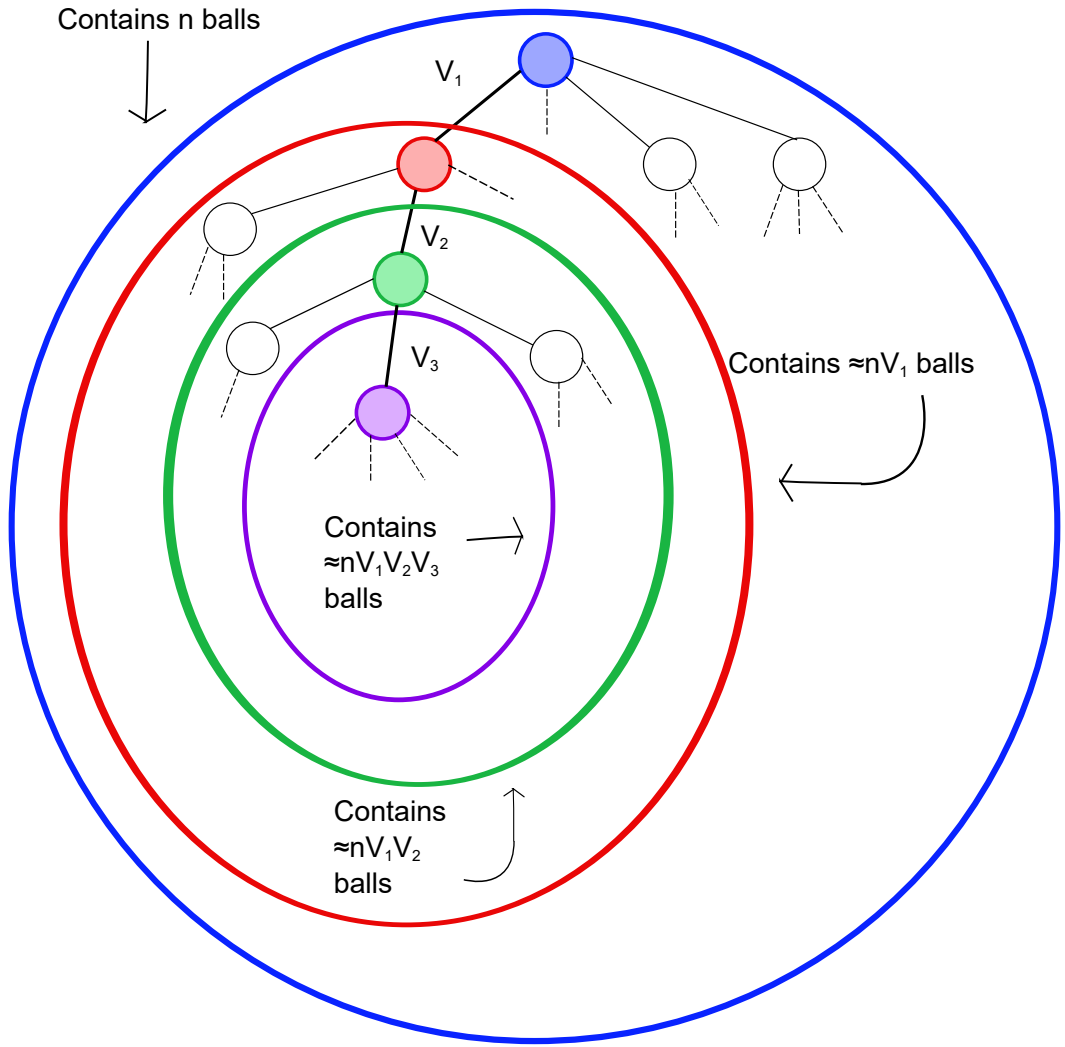



Figure: *Given all split vectors in the tree, $n_v$ for $v$ at depth $d$ is close to $nL_v = n\prod_{j=1}^{d} V_j$, where the $V_j$'s are i.i.d. random variables distributed as $V$*

UPPSALA UNIVERSITET

# Renewal Theory

Study sums $S_k = \sum_{i=1}^{k} X_i$ of *i.i.d* random variables

- A light bulb has a random life time $X_1$ with some distribution and when it breaks it has to be replaced by a new light bulb with a life time $X_2$ of the same distribution
- How many light bulbs are needed say in 1 year time?
- This number is a random variable called **counting process** $\mathcal{N}(t) := \max\{k : S_k \leq t\}$ in renewal theory
- Let $F(t) = \mathbf{P}(X_1 \leq t)$. The **renewal function** is defined as $V(t) := \mathbf{E}(\mathcal{N}(t))$ and satisfies the **renewal equation**

$$\begin{aligned} V(t) &= \sum_{k=1}^{\infty} \mathbf{P}(S_k \leq t) = F(t) + \int_0^t V(t-s)dF(s) \\ &= F(t) + (V * dF)(t) \end{aligned}$$

- Law of large numbers suggests $V(t) = \frac{t}{\mathbf{E}(X)}$+o(t)

UPPSALA UNIVERSITET

# Applying Renewal Theory

- Recall that the subtree sizes $n_v$ for $v$ at depth $d$ are approximated by $n\prod_{j=1}^{d} V_j$
- Let $S_d := -\sum_{j=1}^{d} \ln V_j$. Note that $n\prod_{j=1}^{d} V_j = ne^{-S_d}$
- Define the renewal function

$$U(t) := \sum_{k=1}^{\infty} b^k \mathbf{P}(S_k \leq t),$$

  and let $F(t) := b\mathbf{P}(-\ln V \leq t)$
- For $U(t)$ we obtain the following renewal equation

$$U(t) = F(t) + (U * dF)(t)$$

- As $t \to \infty$, $U(t)$ satisfies

$$U(t) = (\mu^{-1} + o(1))e^t$$

  for the constant $\mu := b\mathbf{E}(-V_1 \ln V_1)$.

UPPSALA UNIVERSITET

# The Random Number of Nodes in a Split Tree

In renewal theory one needs to distinguish between non-lattice and lattice distributions, where non-lattice distributions are much more common. For the rest of the talk we will therefore, for simplicity assume that $\ln(V)$ is non-lattice, where $V$ is distributed as the components $V_i$ in the split vector. This holds true for most split trees.

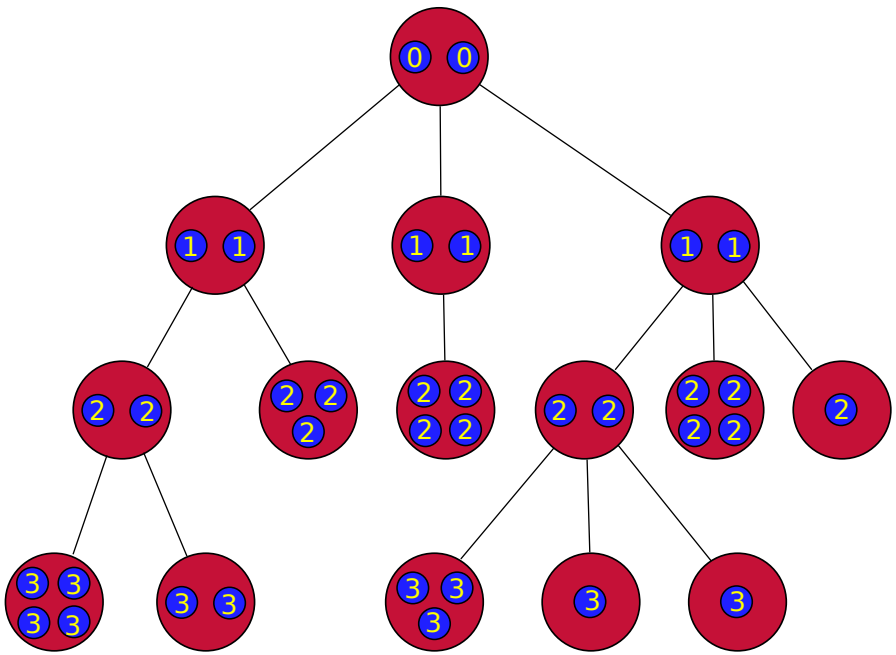
**Theorem** (C. Holmgren, *Electr. Journ. Probab.* 2012)

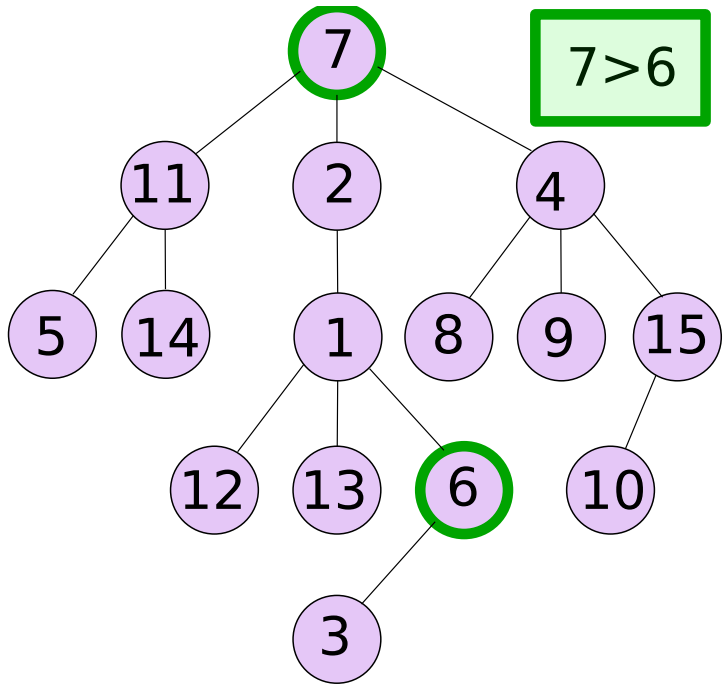
Let $N$ be the random number of nodes in a split tree $T^n$ with $n$ balls. Then it holds that there is a constant $\alpha$ depending on the type of the split tree such that

$$\mathbf{E}[N] = \alpha n + o(n) \text{ and } \mathrm{Var}(N) = o(n^2).$$

UPPSALA UNIVERSITET

# Total Path Length and Number of Inversions

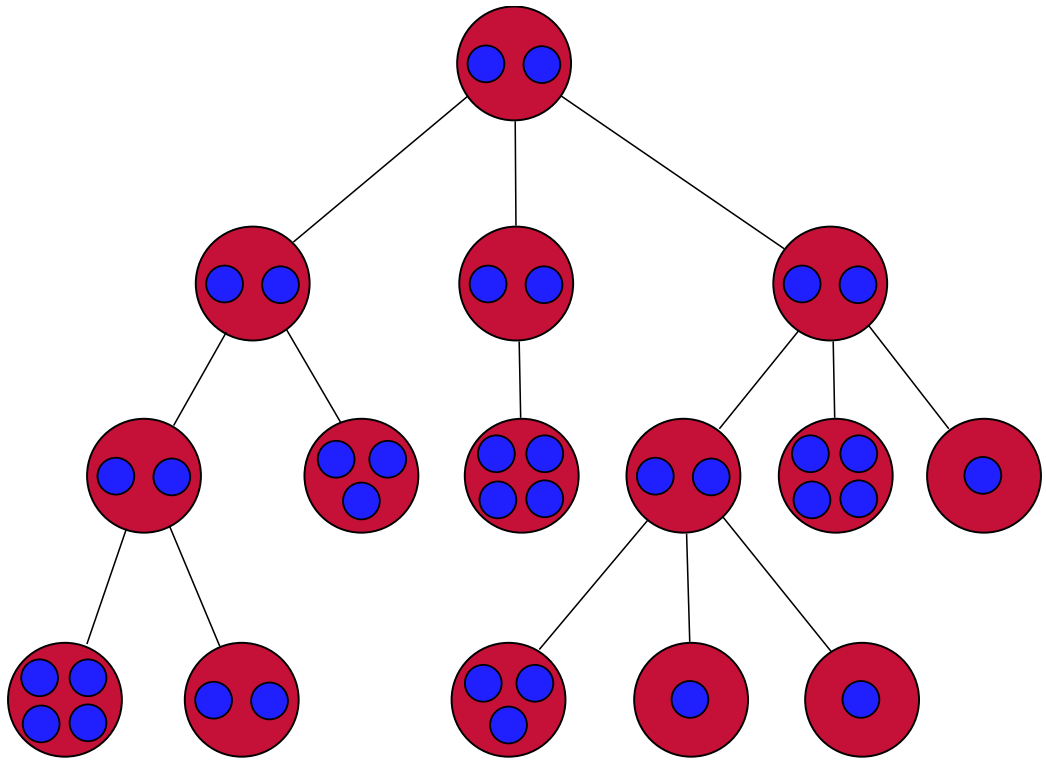




UPPSALA UNIVERSITET

# The Total Path Length of A Rooted Tree

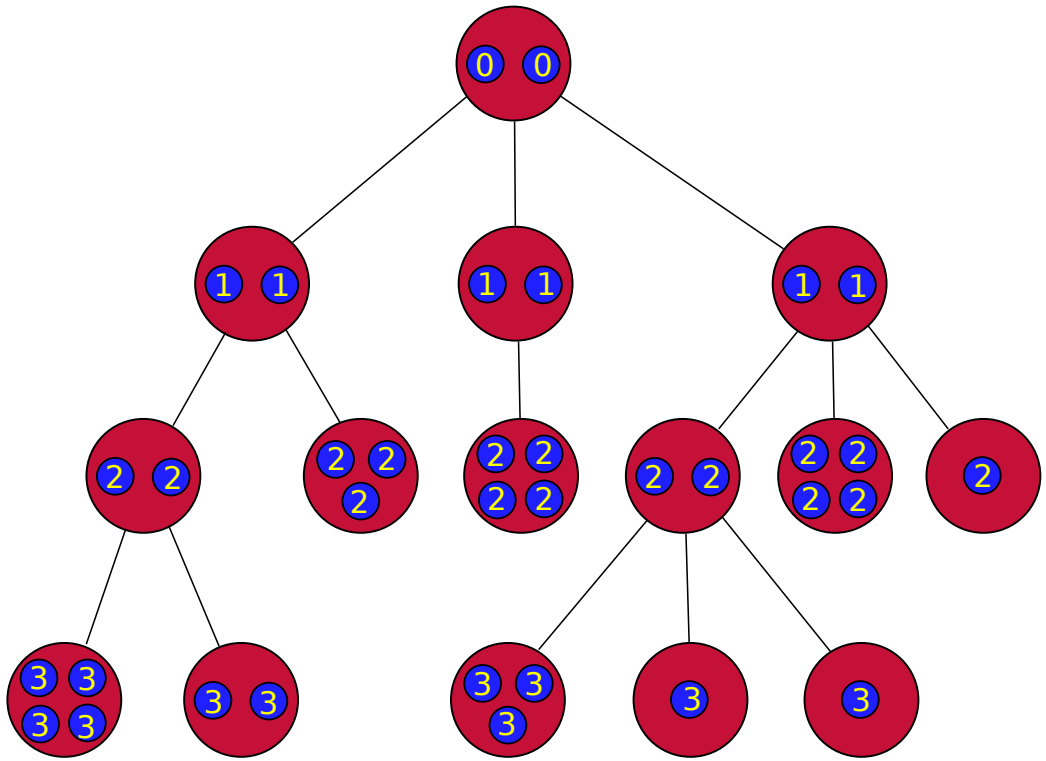
- The total path length is the sum of the depths of all balls in the tree (often represented by keys in tree data structures).



UPPSALA UNIVERSITET

# The Total Path Length of A Rooted Tree

- The total path length is the sum of the depths of all balls in the tree (often represented by keys in tree data structures).



UPPSALA
UNIVERSITET

# The Total Path Length/Running-Time of Sorting Algorithms

- Sorting algorithms sort a collection of data items (often called keys) by comparisons of the input data
- The number of comparisons for a certain key is given by its depth in the tree
- **The total number of comparisons is thus the sum of all depths i.e., the total path length. Hence, the total path length represents a natural cost measure or running time of these algorithms**
- Effective sorting algorithms are represented by $\mathcal{O}(\log n)$ trees with total path length, i.e., running time $\mathcal{O}(n \log n)$

UPPSALA UNIVERSITET

# The Running Time of Quicksort

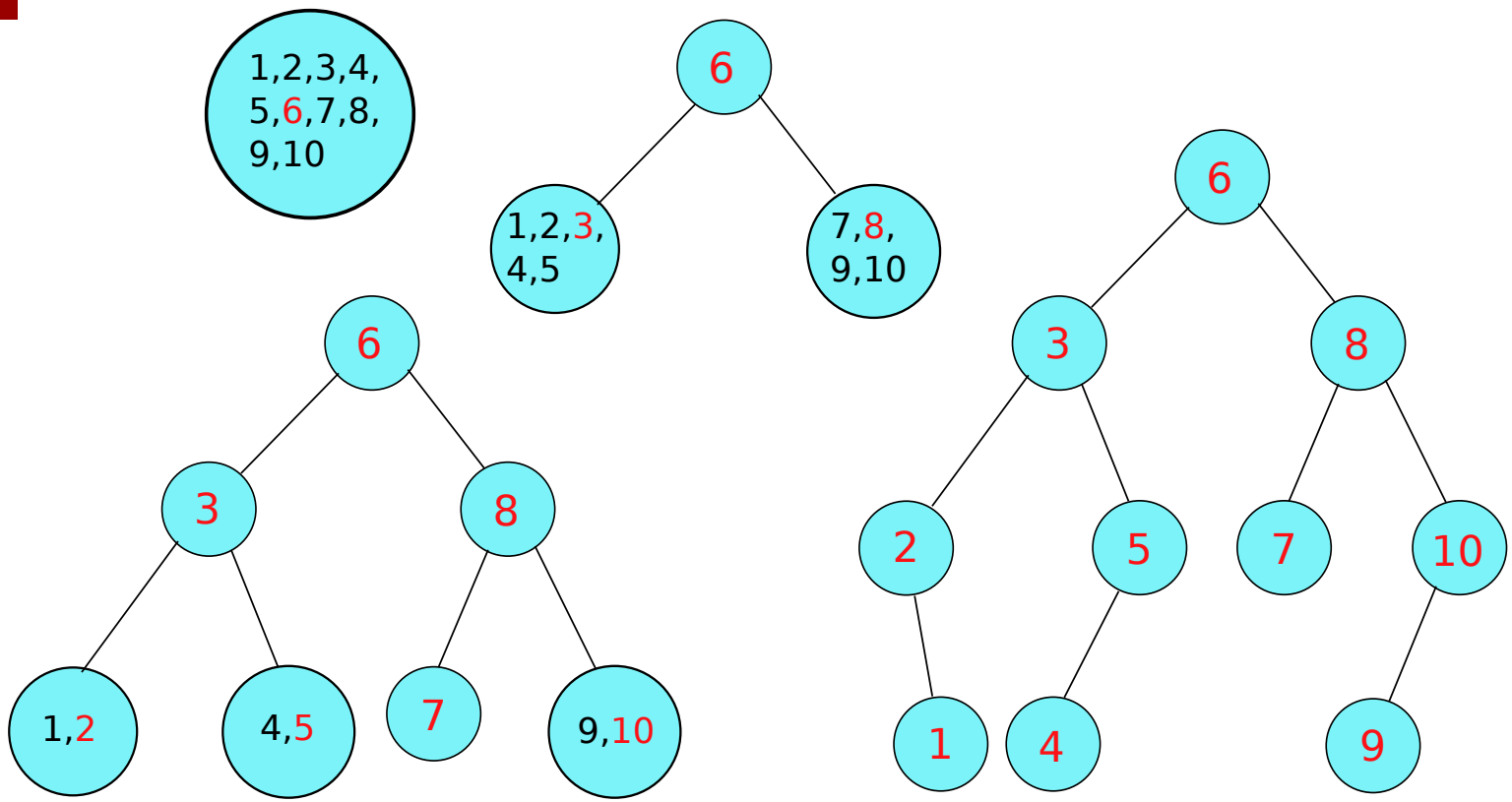
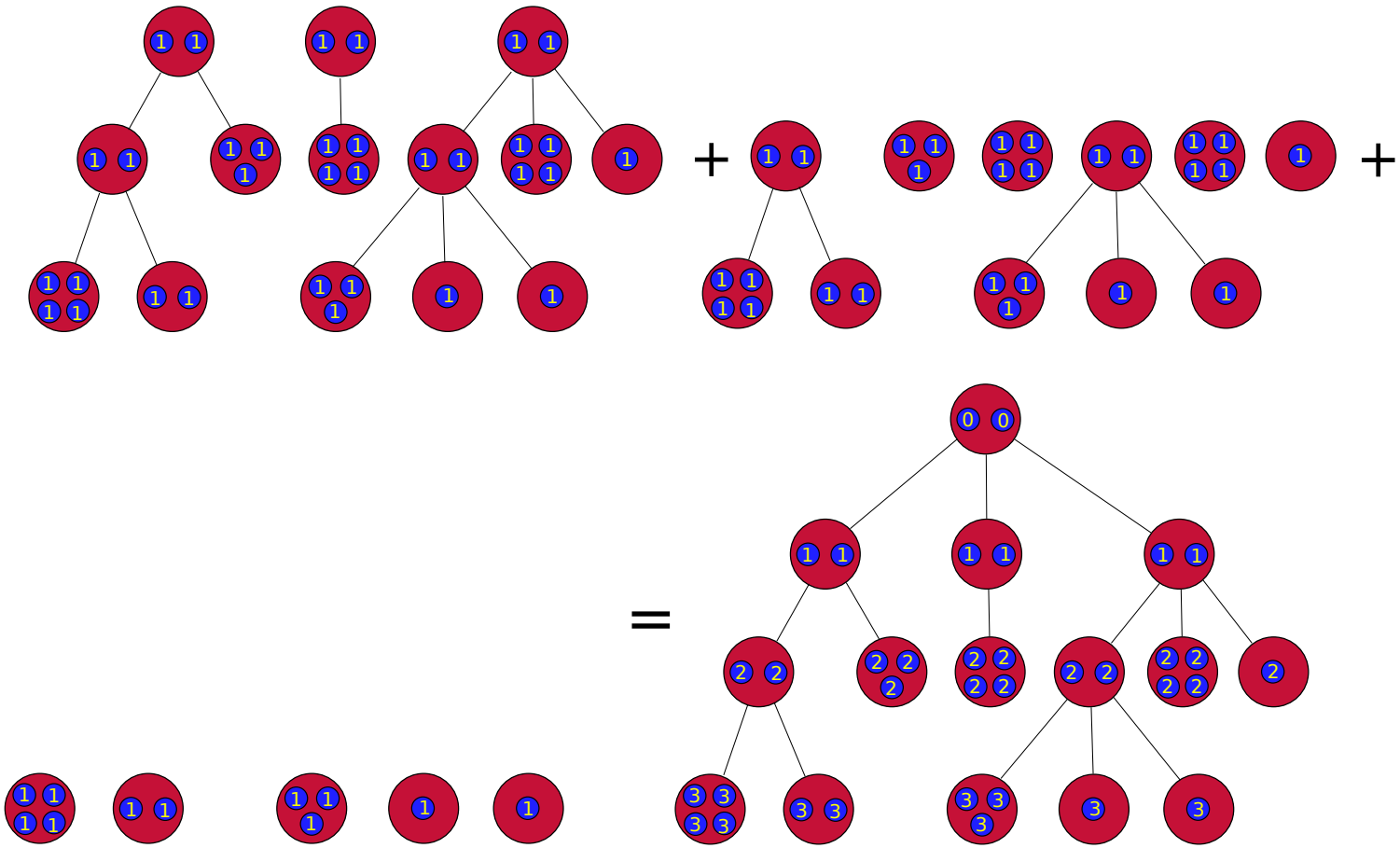



Figure: *The total number of comparisons of Quicksort is equivalent to the total number of black digits in the figure, or equivalently the sum of all depths in the final tree i.e., the total path length of the binary search tree*

UPPSALA UNIVERSITET

# An Equivalent Definition of the Total Path Length

- The total path length can also be defined as the sum of all subtree sizes, i.e., $\Psi(T^n) = \sum_{v \neq r} n_v$



UPPSALA UNIVERSITET

# The Total Path Length of Split Trees

**Theorem** (N. Broutin and **C. Holmgren**, *Annals of Applied Probab.* 2012)

Let $\Psi(T^n)$ be the total path length of a split tree $T^n$ with split vector $\mathcal{V} = (V_1, \ldots, V_b)$. Then

$$\mathbf{E}\left[\Psi(T^n)\right] = \mu^{-1} n \ln n + C_1 n + o(n),$$

where $\mu := b\mathbf{E}(-V_1 \ln V_1)$ and $C_1$ is another constant. Let

$$X_n := \frac{\Psi(T^n) - \mathbf{E}[\Psi(T^n)]}{n}.$$

Then $X_n \to X$ in distribution, where $X$ is the unique solution of the fixed point equation

$$X \stackrel{d}{=} \sum_{k=1}^{b} V_k X^{(k)} + C(\mathcal{V}),$$

where $X^{(k)}$ are *i.i.d.* copies of $X$, with $\mathbf{E}[X] = 0$ and $\mathrm{Var}(X) < \infty$ and $C(\mathcal{V})$ is a function of the split vector $\mathcal{V}$.

UPPSALA UNIVERSITET

# The Total Path Length of Split Trees

- The first asymptotic in $E\Psi(T^n) = \mu^{-1} n \ln n + C_1 n + o(n)$ is trivial from the fact that most nodes (and balls) are concentrated around $\mu^{-1} \ln n$. We showed the second asymptotics $C_1 n$ by using renewal theory

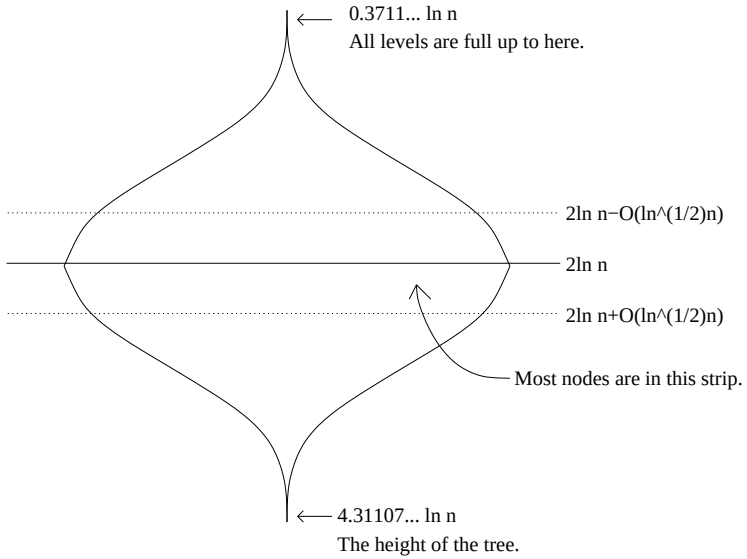



Figure: *The central limit theorem holds for the depths of the nodes. Most nodes are in a strip of width $\mathcal{O}(\sqrt{\ln n})$ around the depth $\mu^{-1} \ln n$, where $\mu := b\mathbf{E}(-V_1 \ln V_1)$. The figure shows the distribution of the nodes in a binary search tree.*


# The Total Path Length of Split Trees

- In a split tree $T^n$ we recall that for each ball $k \in \{1, \ldots, n\}$ its depth $D_k$ converges to a normal distribution, i.e., we have a central limit theorem for the depth $D_k$
- However, although the total path length $\Psi(T^n)$ is the sum of all the depths $D_k$:s it does not converge to a normal distribution!
- Our theorem instead shows that the limiting distribution of the total path length $\Psi(T^n)$ (after normalization) converges to a random variable which solves a fixed-point equation. To show this result we used the so-called contraction method, which is based on the Banach fixed-point theorem

UPPSALA UNIVERSITET

# The Total Path Length of Split Trees

**Corollary** (N. Broutin and **C. Holmgren**, *Annals of Applied Probab.* 2012)

The fixed point equation of $X$ implies that

$$\mathbf{Var}(\Psi(T^n)) \sim \zeta n^2,$$

where the constant $\zeta$ is given by

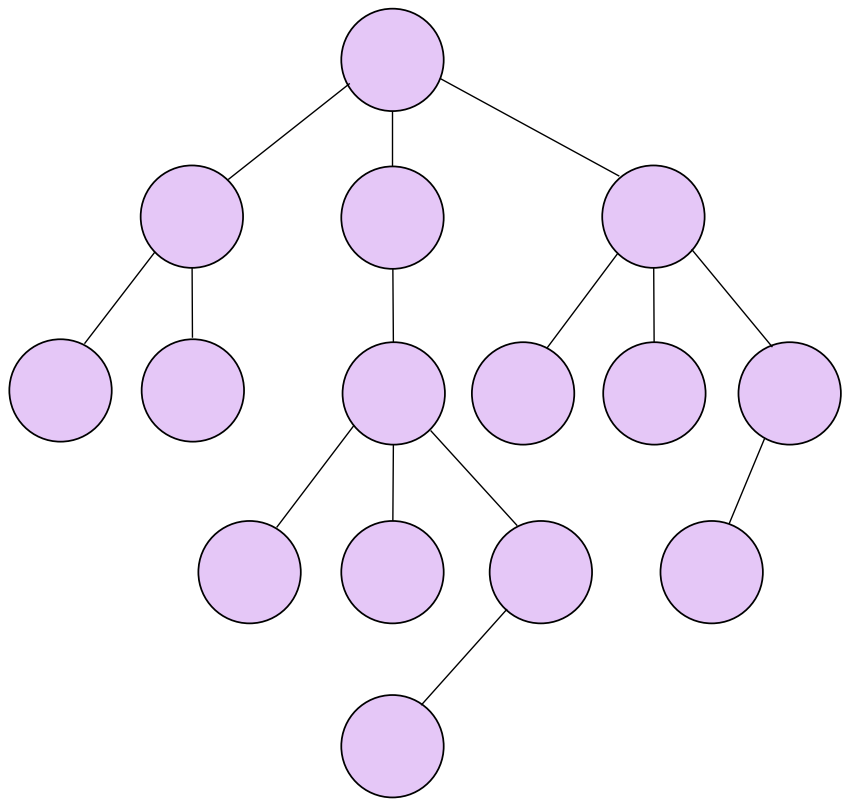
$$\zeta = \mathbf{Var}(X) = \frac{\mu^{-2}\mathbf{E}[(\sum_{i=1}^{b} V_i \ln V_i)^2] - 1}{1 - \sum_{i=1}^{b} \mathbf{E}[V_i^2]};$$

recall the split vector $\mathcal{V} = (V_1, V_2, \ldots, V_b)$.

UPPSALA UNIVERSITET

# What is an Inversion of a Rooted Tree?

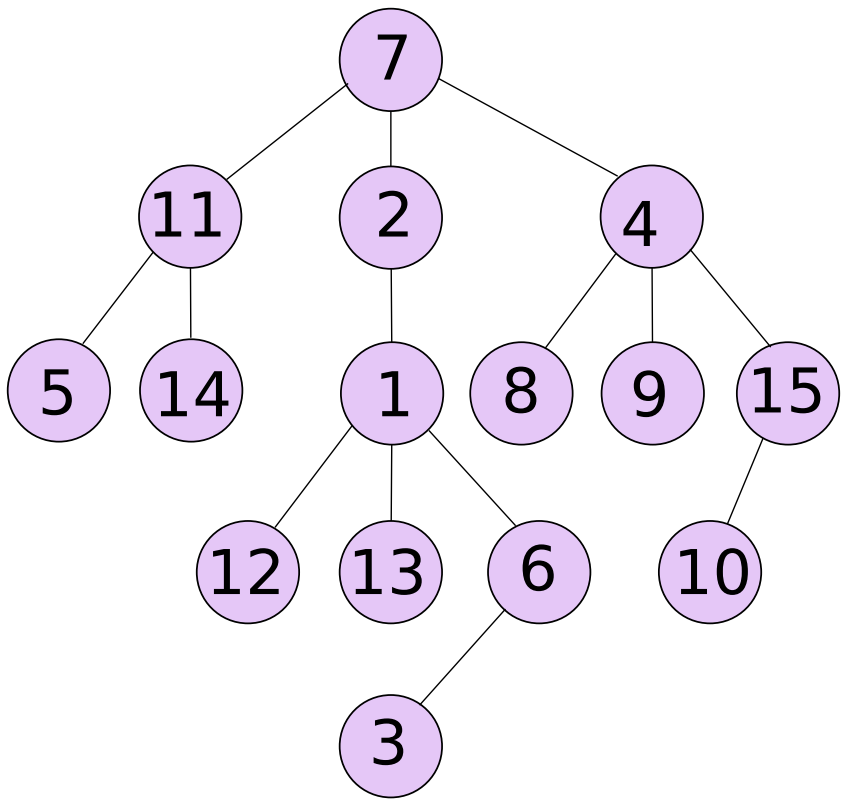
- Write $u < v$ if $u$ is an ancestor of $v$. Given a bijection $\lambda : V \to \{1, \ldots, |V|\}$ (a **node labelling**), define an **inversion** as a pair $(u, v)$, where $u < v$ so that $\lambda(u) > \lambda(v)$



UPPSALA UNIVERSITET

# What is an Inversion of a Rooted Tree?

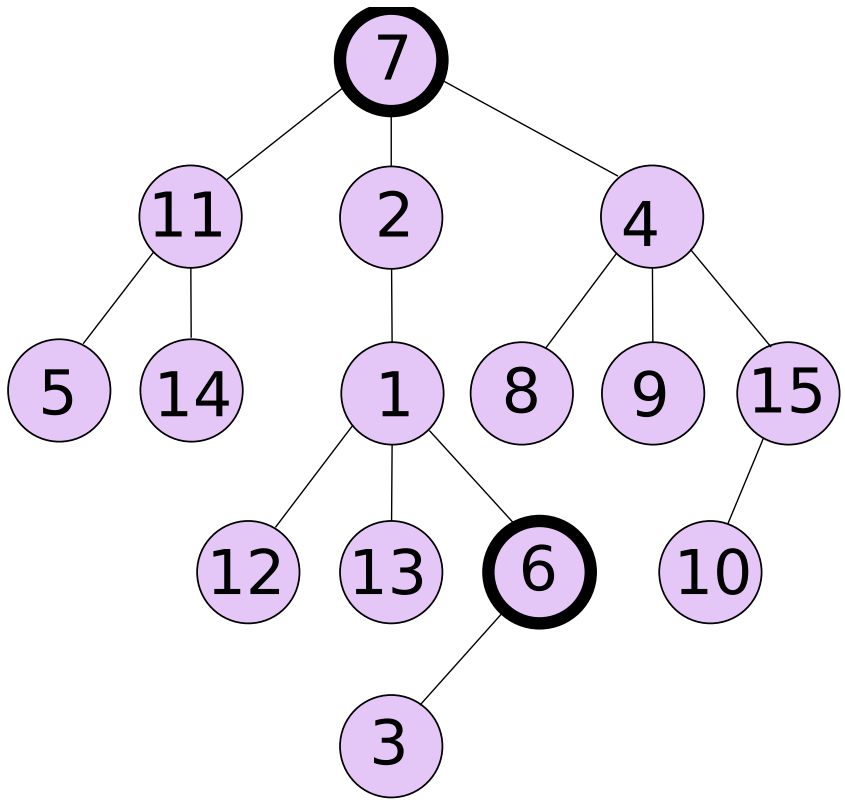
- Write $u < v$ if $u$ is an ancestor of $v$. Given a bijection $\lambda : V \to \{1, \ldots, |V|\}$ (a **node labelling**), define an **inversion** as a pair $(u, v)$, where $u < v$ so that $\lambda(u) > \lambda(v)$



UPPSALA UNIVERSITET

# What is an Inversion of a Rooted Tree?

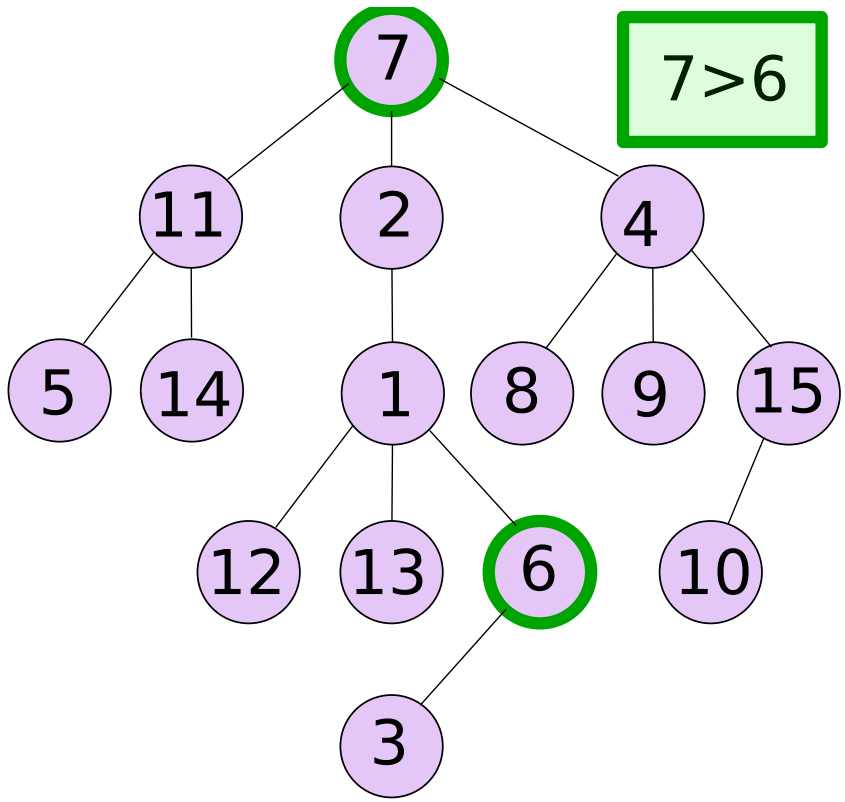
- Write $u < v$ if $u$ is an ancestor of $v$. Given a bijection $\lambda : V \to \{1, \ldots, |V|\}$ (a **node labelling**), define an **inversion** as a pair $(u, v)$, where $u < v$ so that $\lambda(u) > \lambda(v)$



UPPSALA UNIVERSITET

# What is an Inversion of a Rooted Tree?

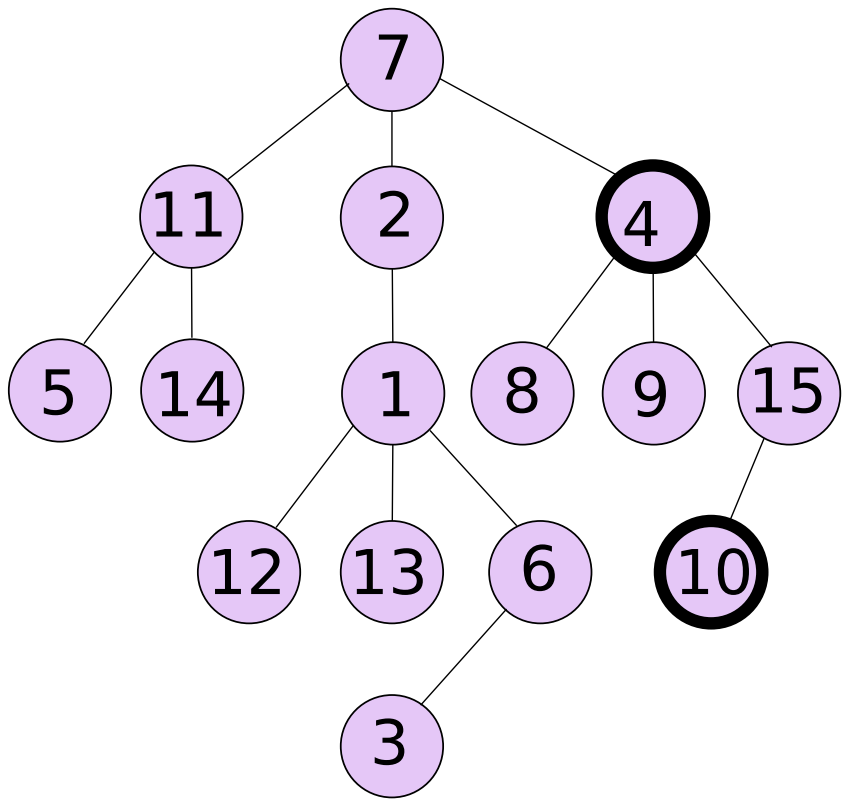
- Write $u < v$ if $u$ is an ancestor of $v$. Given a bijection $\lambda : V \to \{1, \ldots, |V|\}$ (a **node labelling**), define an **inversion** as a pair $(u, v)$, where $u < v$ so that $\lambda(u) > \lambda(v)$



UPPSALA UNIVERSITET

# What is an Inversion of a Rooted Tree?

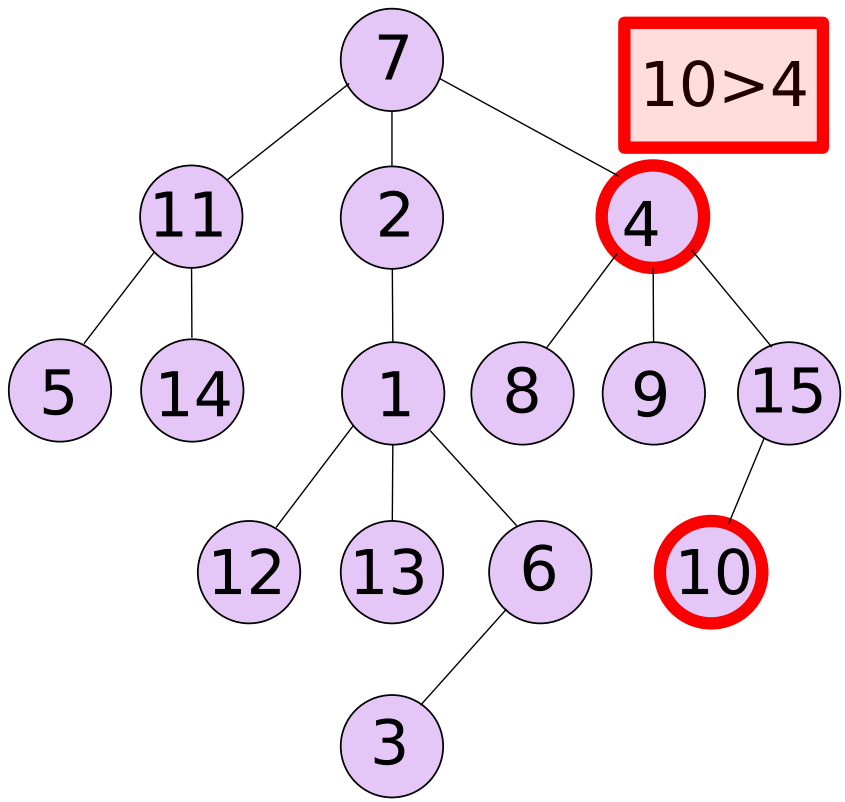
- Write $u < v$ if $u$ is an ancestor of $v$. Given a bijection $\lambda : V \to \{1, \ldots, |V|\}$ (a **node labelling**), define an **inversion** as a pair $(u, v)$, where $u < v$ so that $\lambda(u) > \lambda(v)$



UPPSALA UNIVERSITET

# What is an Inversion of a Rooted Tree?

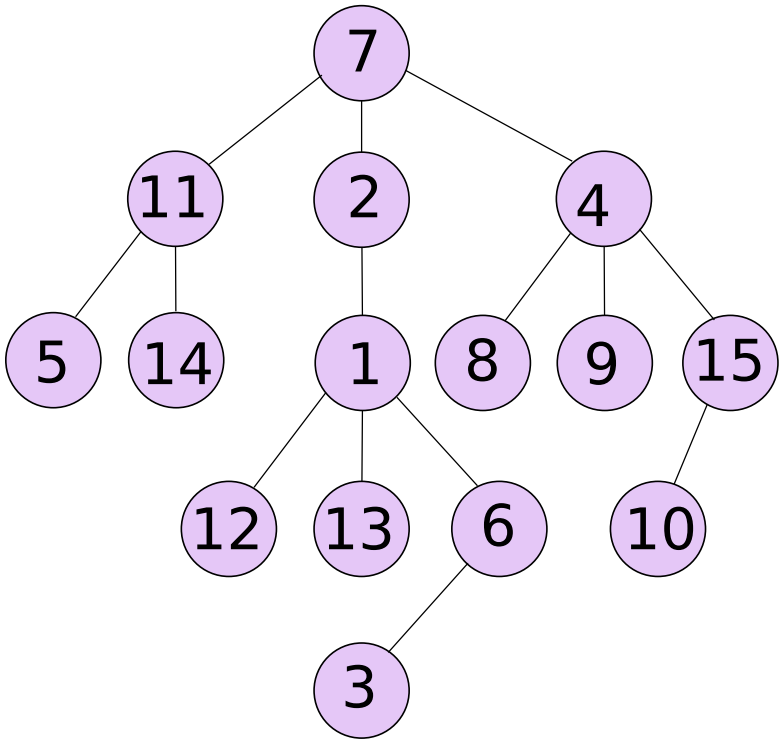
- Write $u < v$ if $u$ is an ancestor of $v$. Given a bijection $\lambda : V \to \{1, \ldots, |V|\}$ (a **node labelling**), define an **inversion** as a pair $(u, v)$, where $u < v$ so that $\lambda(u) > \lambda(v)$



UPPSALA UNIVERSITET

# Inversions and Total Path Length

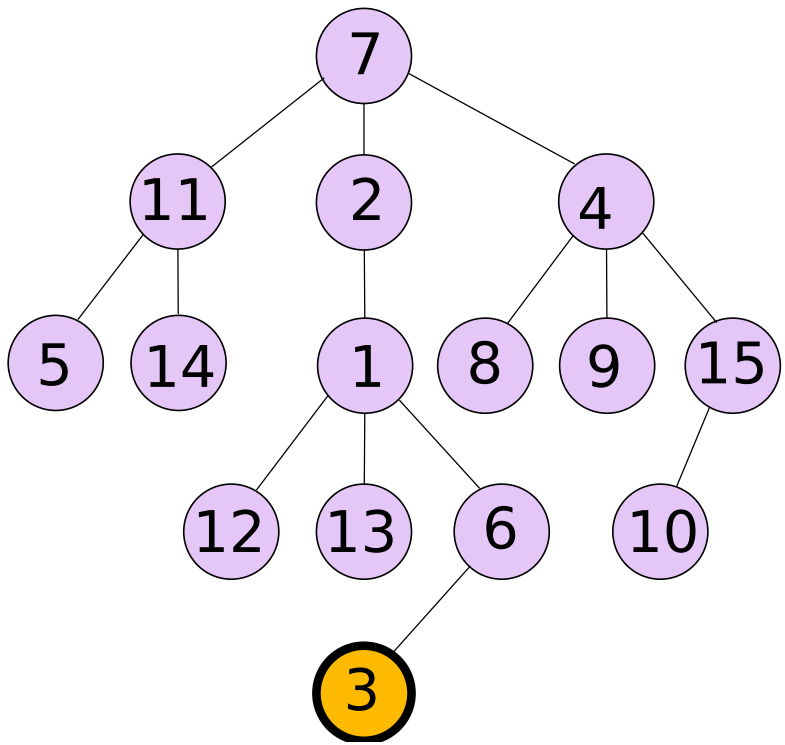
- Label the nodes in the tree randomly with the numbers $1, \ldots, n$



UPPSALA UNIVERSITET

# Inversions and Total Path Length

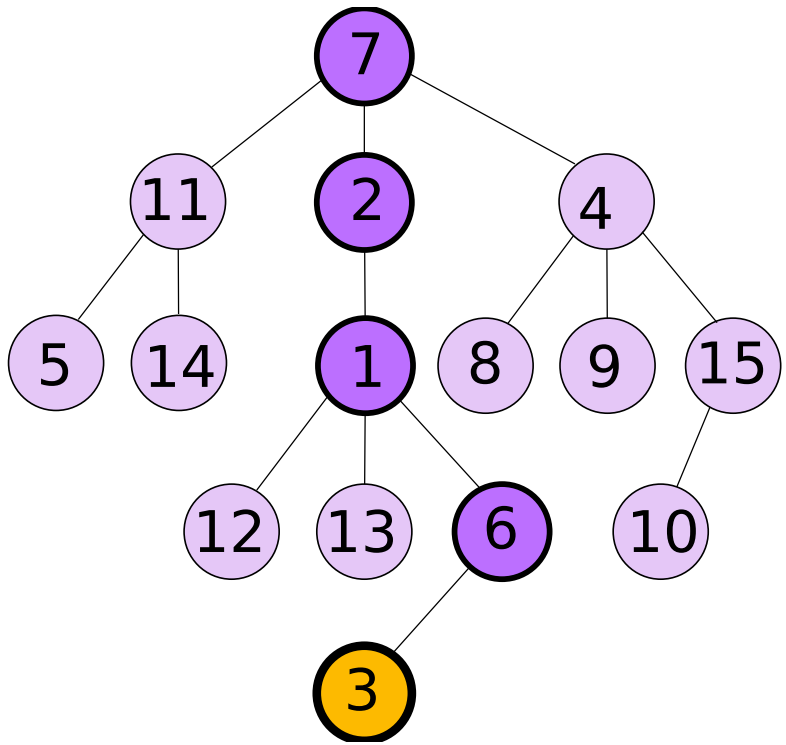
- Label the nodes in the tree randomly with the numbers $1, \ldots, n$
- A node is compared with all the values in its path up to the root



UPPSALA UNIVERSITET

# Inversions and Total Path Length

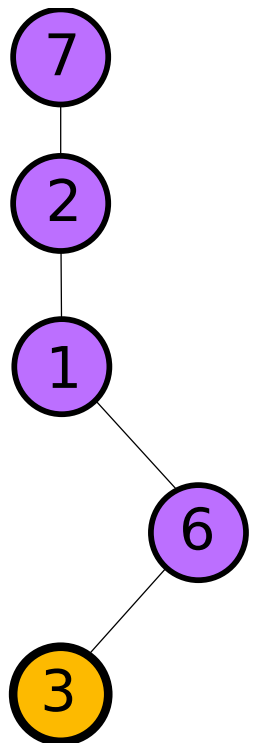
- Label the nodes in the tree randomly with the numbers $1, \ldots, n$
- A node is compared with all the values in its path up to the root



UPPSALA UNIVERSITET

# Inversions and Total Path Length

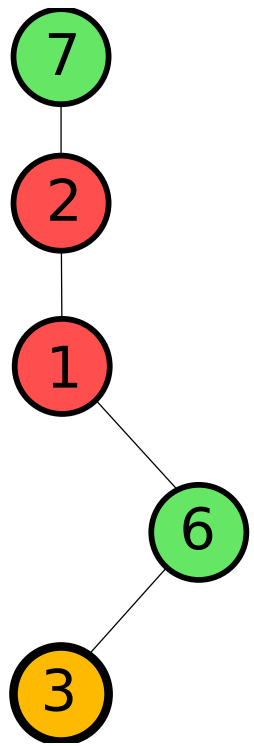
- Label the nodes in the tree randomly with the numbers $1, \ldots, n$
- A node is compared with all the values in its path up to the root



UPPSALA UNIVERSITET

# Inversions and Total Path Length

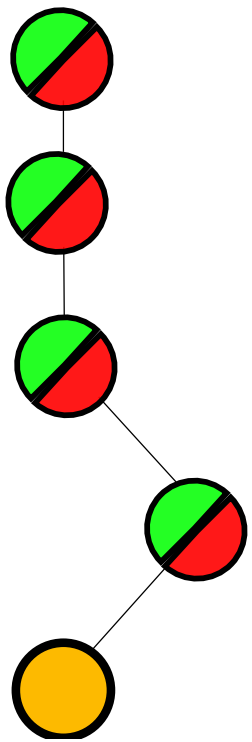
- Label the nodes in the tree randomly with the numbers $1, \ldots, n$
- A node is compared with all the values in its path up to the root
- Each ancestor with a larger value results in one inversion



UPPSALA UNIVERSITET

# Inversions and Total Path Length

- Each ancestor with a larger value results in one inversion
- For each ancestor there is 50 % chance that it has a larger value



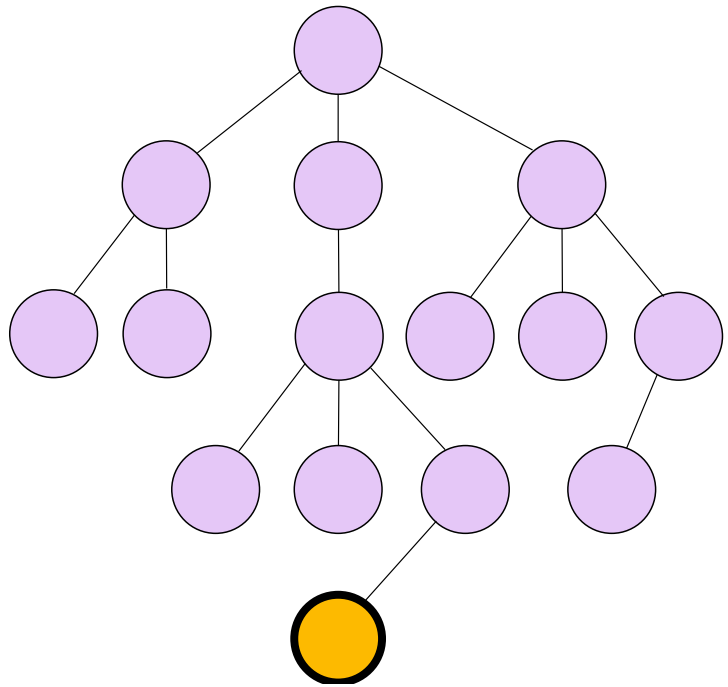

# Inversions and Total Path Length

- Let $T$ be a rooted labelled tree where each node $u$ has a label $\lambda(u) \in \{1, \ldots, n\}$ and all labels are different
- Let $I(T)$ be the number of inversions of $T$
- As we just illustrated for any $u < v$ (i.e., $u$ is an ancestor of $v$) we have $\mathbf{P}(\lambda(u) > \lambda(v)) = 1/2$

UPPSALA UNIVERSITET

# Inversions and Total Path Length

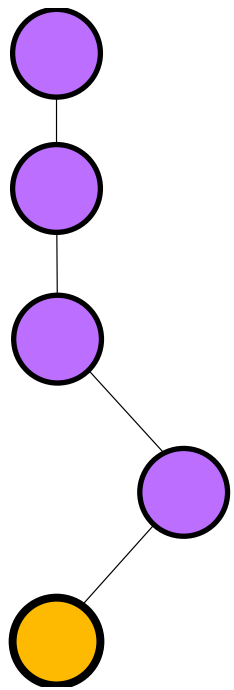
- We will now relate the number of inversions to the total path length of $T$
- The number of ancestors of a node is the same as its depth



UPPSALA UNIVERSITET

# Inversions and Total Path Length

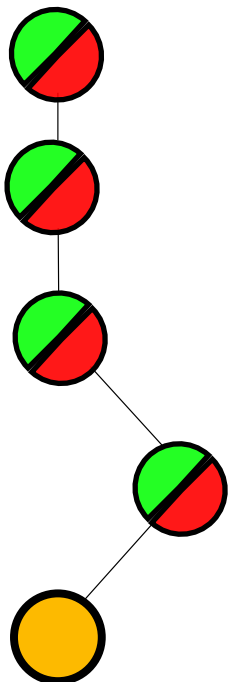
- We will now relate the number of inversions to the total path length of $T$
- The number of ancestors of a node is the same as its depth
- Each ancestor with a larger value results in one inversion



UPPSALA UNIVERSITET

# Inversions and Total Path Length

- We will now relate the number of inversions to the total path length of $T$
- The number of ancestors of a node is the same as its depth
- Each ancestor with a larger value results in one inversion
- For each ancestor there is 50 % chance that it has a larger value




# Inversions and Total Path Length

- The sum of all the depths of the nodes of $T$ is equal to the total path length. Thus, we write $\Psi(T) = \sum_v d(v)$ for the total path length of $T$

- Then it is easy to see from our previous illustration that the expected number of inversions is equal to,

$$\mathbf{E}\left[I(T)\right] = \frac{1}{2}\sum_v d(v) = \frac{1}{2}\Psi(T)$$

UPPSALA UNIVERSITET

# The Number of Inversions in Split Trees

- Thus, we just showed that the expected number of inversions in a rooted tree $T$ is equal to half of its total path length,

$$\mathbf{E}\left[I(T)\right] = \frac{1}{2}\sum_{v} d(v) = \frac{1}{2}\Psi(T)$$

- We can now show the analogous result for a random split tree $T^n$ constructed from $n$ balls. Let $I(T^n)$ be the number of inversions in a split tree with split vector $\mathcal{V} = (V_1, \ldots, V_b)$. Then it simply follows from our previous argument that $\mathbf{E}\left[I(T^n)\right] = \frac{1}{2}\mathbf{E}\left[\Psi(T^n)\right]$

UPPSALA UNIVERSITET

# The Number of Inversions of Split Trees

Let $I(T^n)$ be the number of inversions in an arbitrary split tree $T^n$. We simply observed that $\mathbf{E}[I(T^n)] = \frac{1}{2}\mathbf{E}[\Psi(T^n)]$.

**Theorem** (X. S. Cai, **C. Holmgren**, S. Janson, T. Johansson & F. Skerman *Combin. Probab. Comput.* 2019)

*Let* $X_n = \frac{I(T^n) - \mathbf{E}[I(T^n)]}{n}$. *Then* $X_n \to X$ *in distribution, where* $X$ *is the unique solution of the fixed point equation*

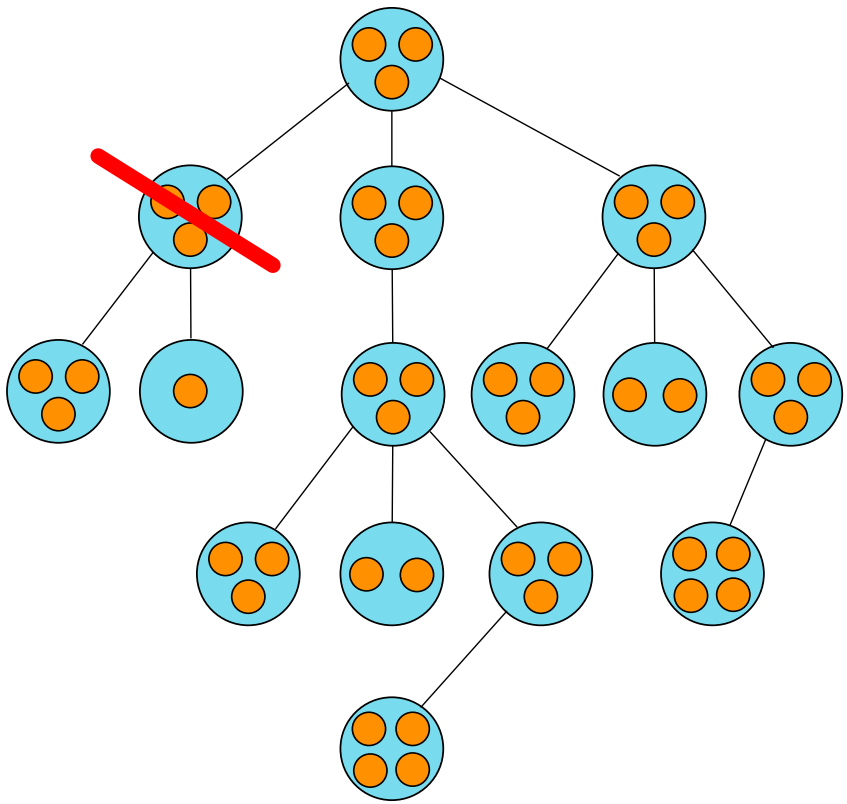
$$X \stackrel{d}{=} \sum_{k=1}^{b} V_k X^{(k)} + \sum_{j=1}^{s_0} U_j + (C(\mathcal{V}) - 1)\frac{s_0}{2}$$

*where* $U_j \sim \mathrm{Unif}(0,1)$, *each* $X^{(i)}$ *is an independent copy of* $X$ *and* $C(\mathcal{V})$ *is a function of the vector* $\mathcal{V}$. *The variables are independent except for the* $V_i$*'s.*

In M. Albert, **C. Holmgren**, T. Johansson & F. Skerman *Algorithmica* (2020) we also study the number of general permutations (the number of inversions is one example of such a permutation) in split trees.

UPPSALA
UNIVERSITET

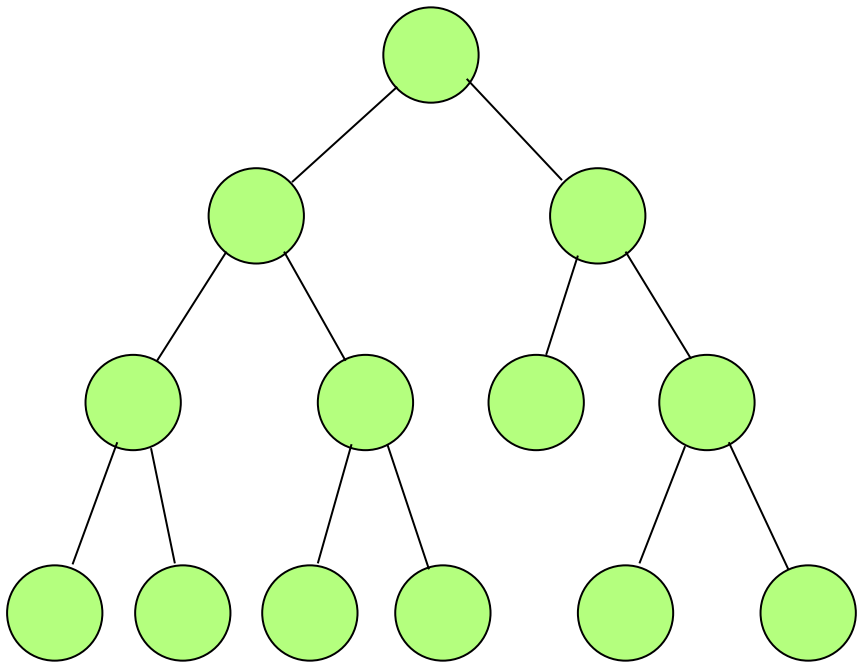
# Cutting Down Split Trees



UPPSALA UNIVERSITET

# What is a Cutting in a Rooted Tree?

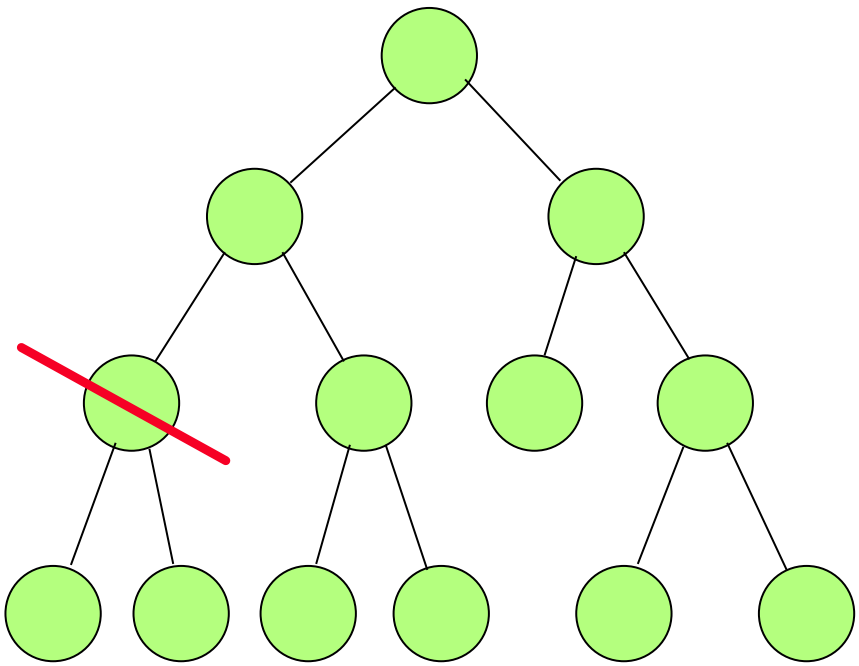
- Choose a random node



UPPSALA UNIVERSITET

# What is a Cutting in a Rooted Tree?

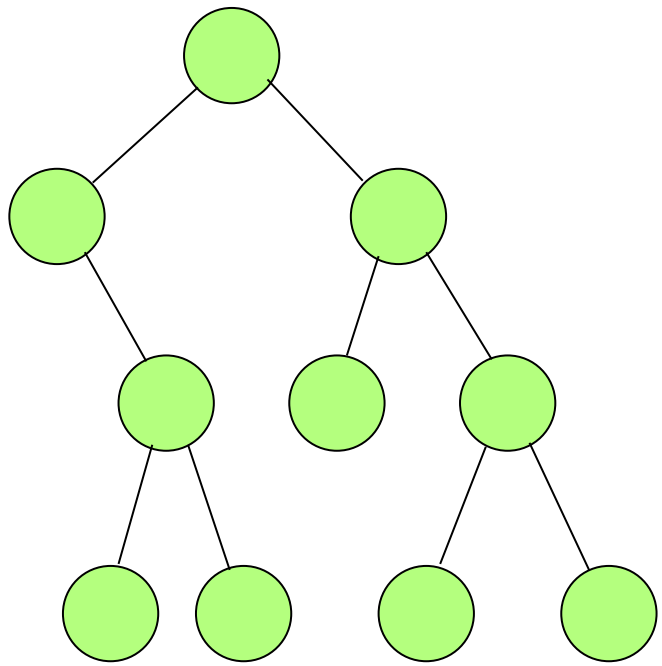
- Choose a random node
- Cut in this node so that the tree separates into two parts and keep only the part which contains the root



UPPSALA UNIVERSITET

# What is a Cutting in a Rooted Tree?

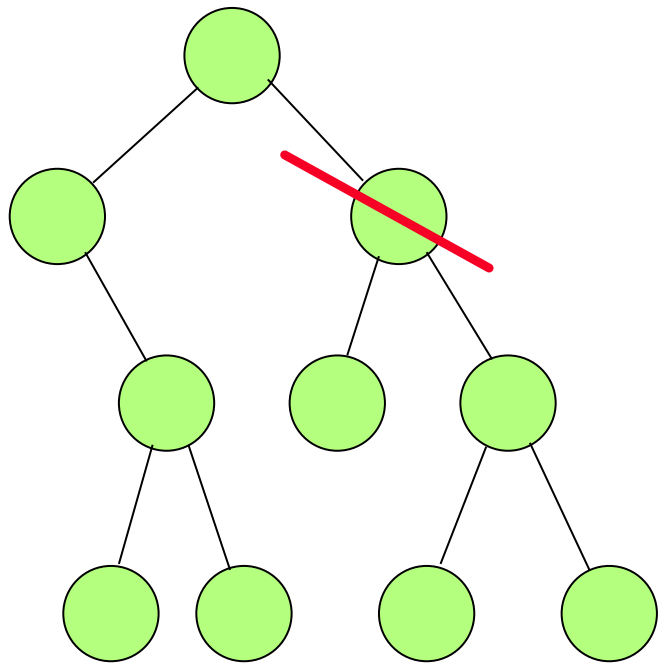
- Choose a random node
- Cut in this node so that the tree separates into two parts and keep only the part which contains the root



UPPSALA UNIVERSITET

# What is a Cutting in a Rooted Tree?

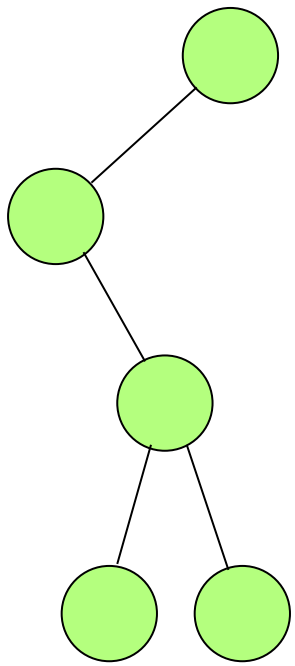
- Choose a random node
- Cut in this node so that the tree separates into two parts and keep only the part which contains the root



UPPSALA UNIVERSITET

# What is a Cutting in a Rooted Tree?

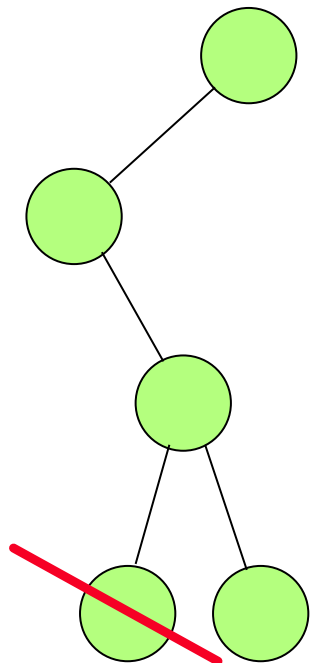
- Choose a random node
- Cut in this node so that the tree separates into two parts and keep only the part which contains the root



UPPSALA UNIVERSITET

# What is a Cutting in a Rooted Tree?

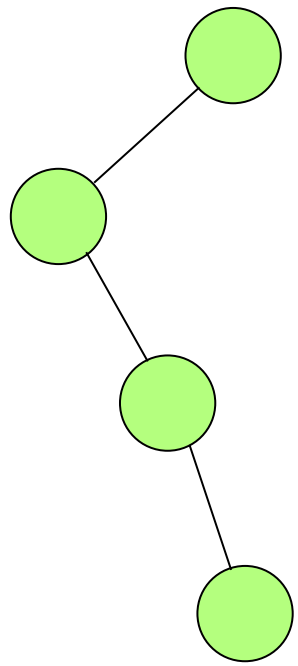
- Choose a random node
- Cut in this node so that the tree separates into two parts and keep only the part which contains the root



UPPSALA UNIVERSITET

# What is a Cutting in a Rooted Tree?

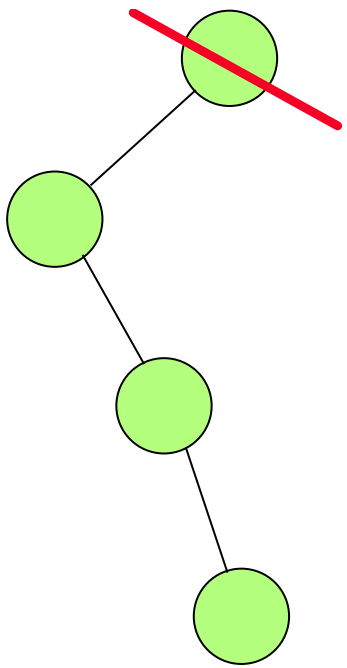
- Choose a random node
- Cut in this node so that the tree separates into two parts and keep only the part which contains the root



UPPSALA UNIVERSITET

# What is a Cutting in a Rooted Tree?

- Choose a random node
- Cut in this node so that the tree separates into two parts and keep only the part which contains the root
- What is the total number of cuts required until the root is cut?



UPPSALA
UNIVERSITET

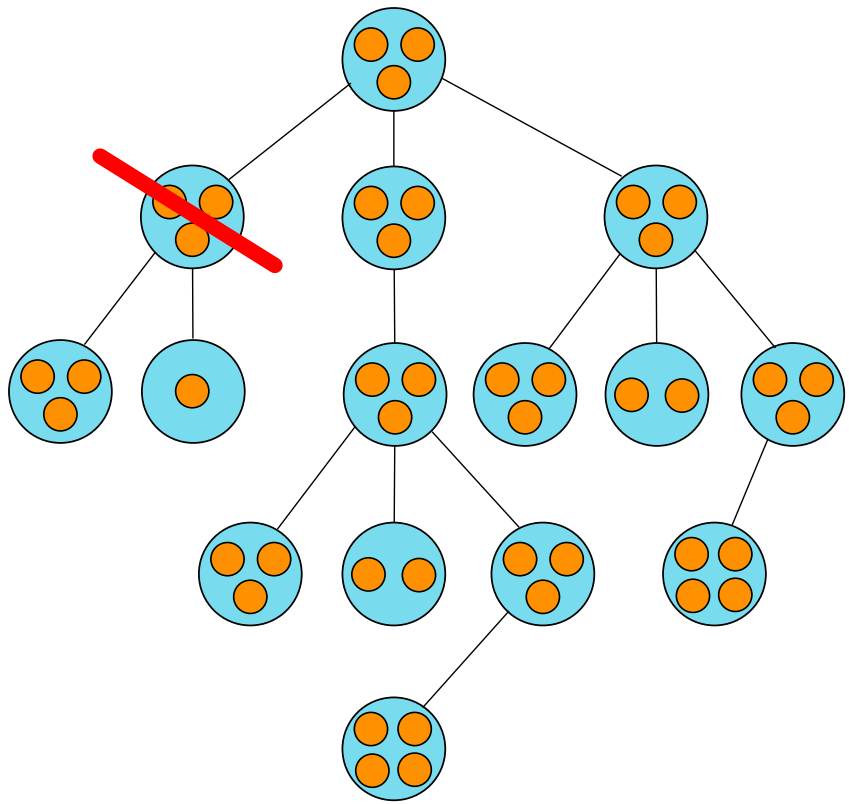
# Cuttings and Botnets

- The model of cuttings can also be applied to *botnets*, i.e., malicious computer networks consisting of compromised machines which are often used in spamming or attacks
- The nodes in a tree (or another type of graph) represent the computers in a botnet, and the root represents the bot-master
- The effectiveness of a botnet can be measured using the size of the component containing the root, which indicates the resources available to the bot-master
- To take down a botnet means to reduce the size of this root component as much as possible
- The number of cuttings then measures how difficult it is to completely isolate the bot-master

UPPSALA UNIVERSITET

# The Number of Cuts to Cut Down a Split Tree

- For an arbitrary split tree $T^n$, we will analyze the total (and random) number of cuts performed until the root is finally cut



UPPSALA UNIVERSITET

# The Number of Cuts to Cut Down a Split Tree

## Theorem ( **C. Holmgren**, *Adv. Appl. Probab.* 2011 )

Let $X(T^n)$ be the random number of cuts that is required to cut down a split tree with $n$ balls. Then it holds that as $n \to \infty$,

$$\left(X(T^n) - f(n)\right) \Big/ \frac{\alpha n}{\mu^{-2} \ln^2 n} \xrightarrow{d} -W, \quad \text{where}$$

$$f(n) := \frac{\alpha n}{\mu^{-1} \ln n} + \frac{\alpha n \ln\ln n}{\mu^{-1} \ln^2 n} - \frac{c_1 n}{\mu^{-1} \ln^2 n}$$

for constants $\mu := b\mathbf{E}(-V_1 \ln V_1)$, $\alpha$ and $c_1$ depending on the type of the split tree, where $W$ has a weakly 1-stable distribution.

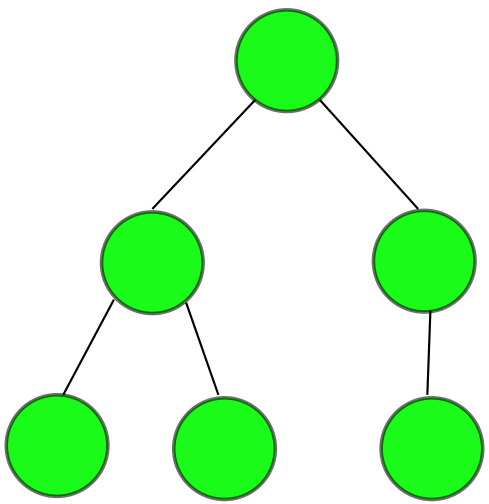
**C. Holmgren**, *Combinat. Probab. & Comput.* (2010) (for the specific case of the binary search tree)

The theorem tells that the first order asymptotics of the number of cuts $X(T^n)$ is $\frac{\alpha n}{\mu^{-1} \ln n}$ and the second order asymptotics is $\frac{\alpha n \ln\ln n}{\mu^{-1} \ln^2 n}$.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees



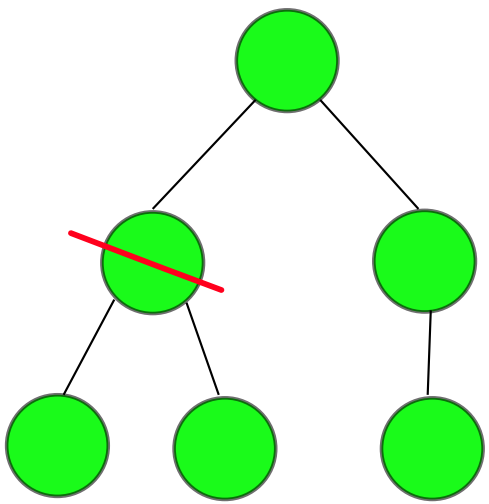
X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees



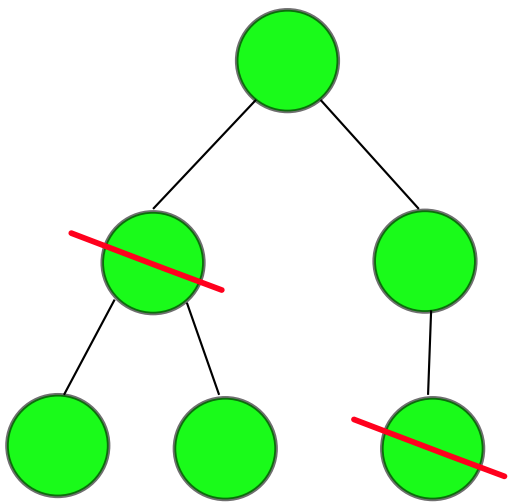
X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees



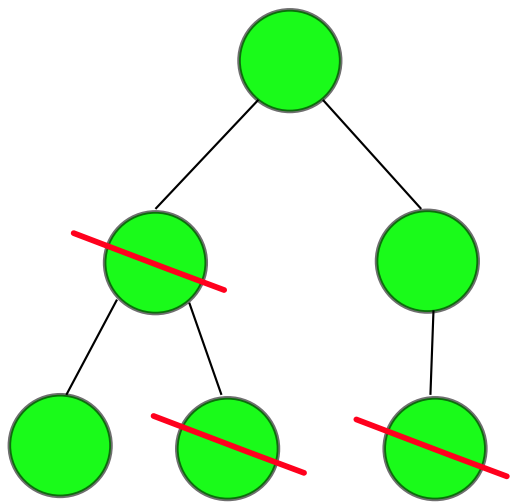
X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees



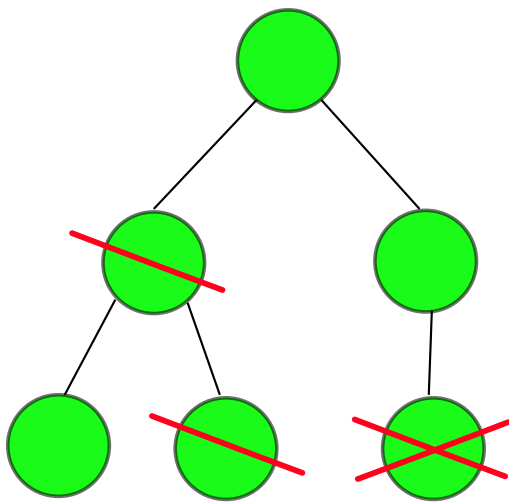
X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees



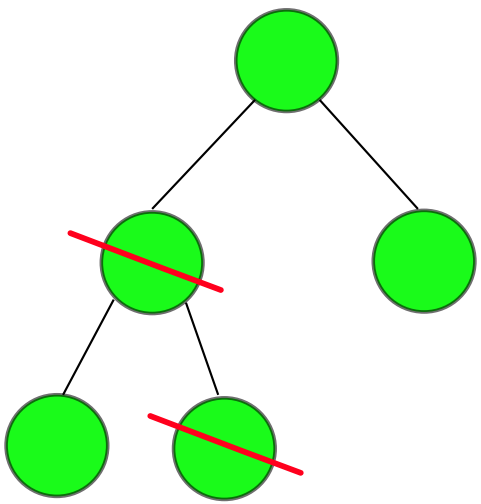
X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees



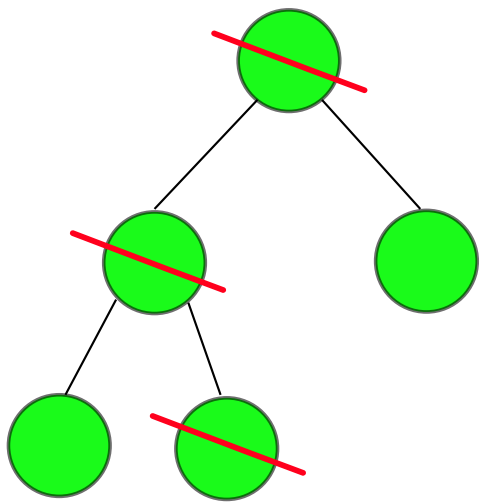
X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees



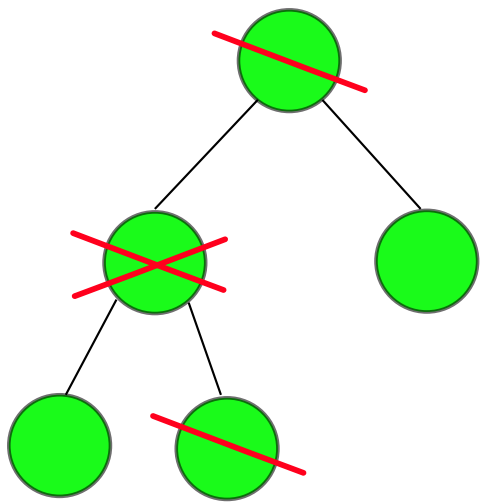
X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees



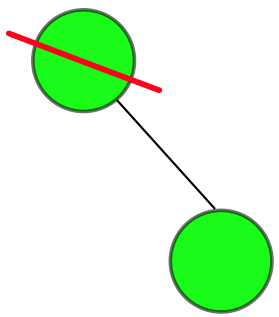
X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees



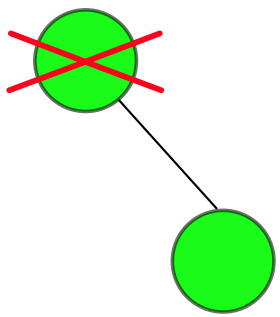
X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA UNIVERSITET

# The Number of $k$-Cuts to Cut Down a Split Tree

- More recently, we have generalised the cutting model by introducing the so-called ***k*-cut model**, where each node has to be cut $k$ times (instead of 1 time) before it is deleted
- What is the total number of cuts needed until the root is cut $k$ times (i.e., the whole tree is destroyed)? We have studied this model for different types of trees including (random) split trees

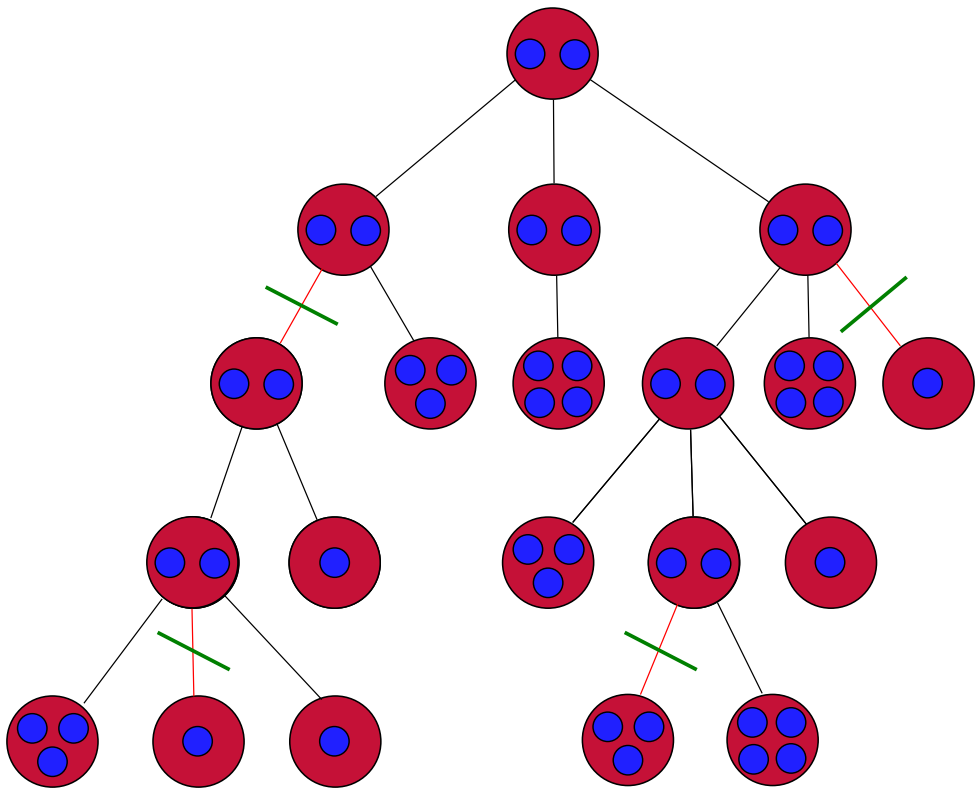


X.S. Cai, L. Devroye, **C. Holmgren** & F. Skerman *Electron. Journal of Probab.* 2019.

X.S. Cai and **C. Holmgren** *Electron. Journal of Combinatorics* 2019.

UPPSALA
UNIVERSITET

# Bond Percolation on Split Trees

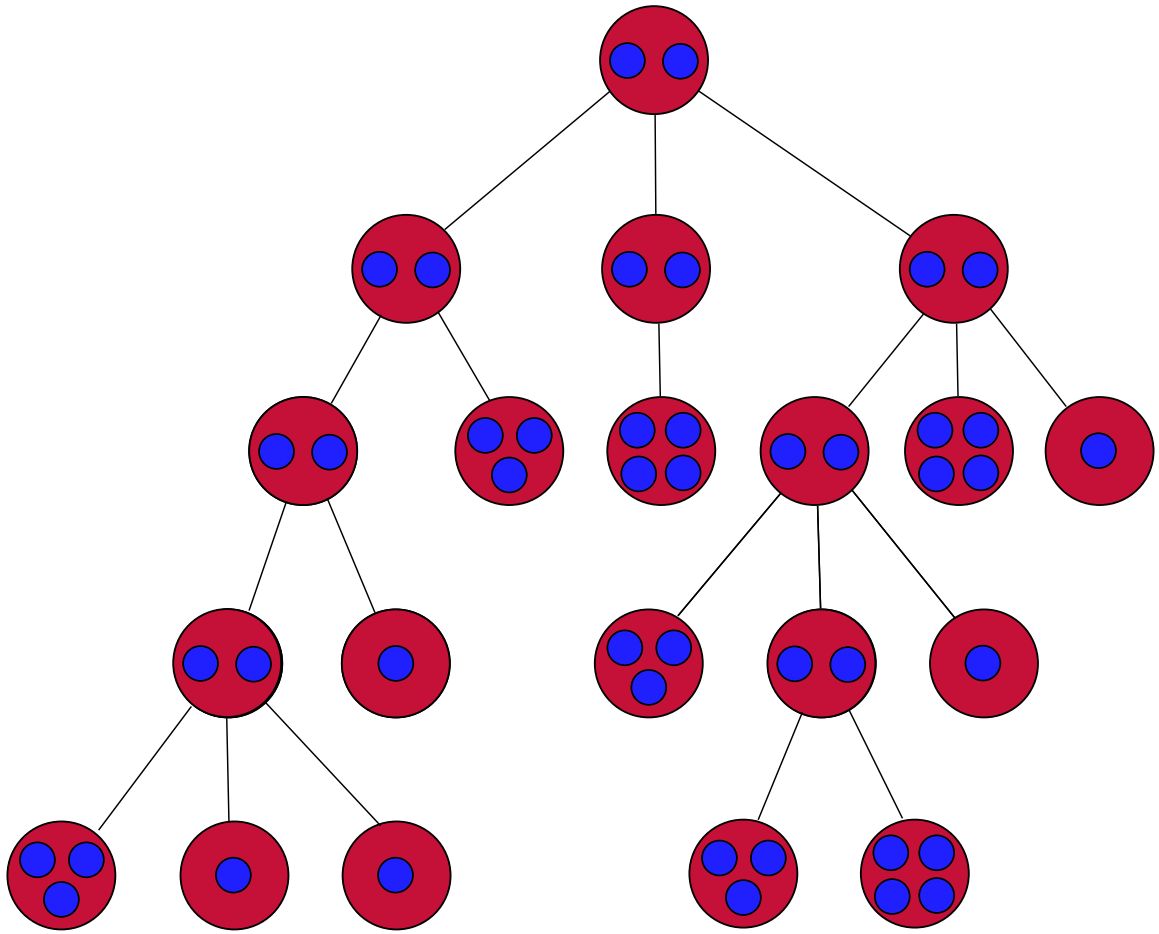


*Cecilia Holmgren*

UPPSALA UNIVERSITET

# Bond Percolation on Split Trees

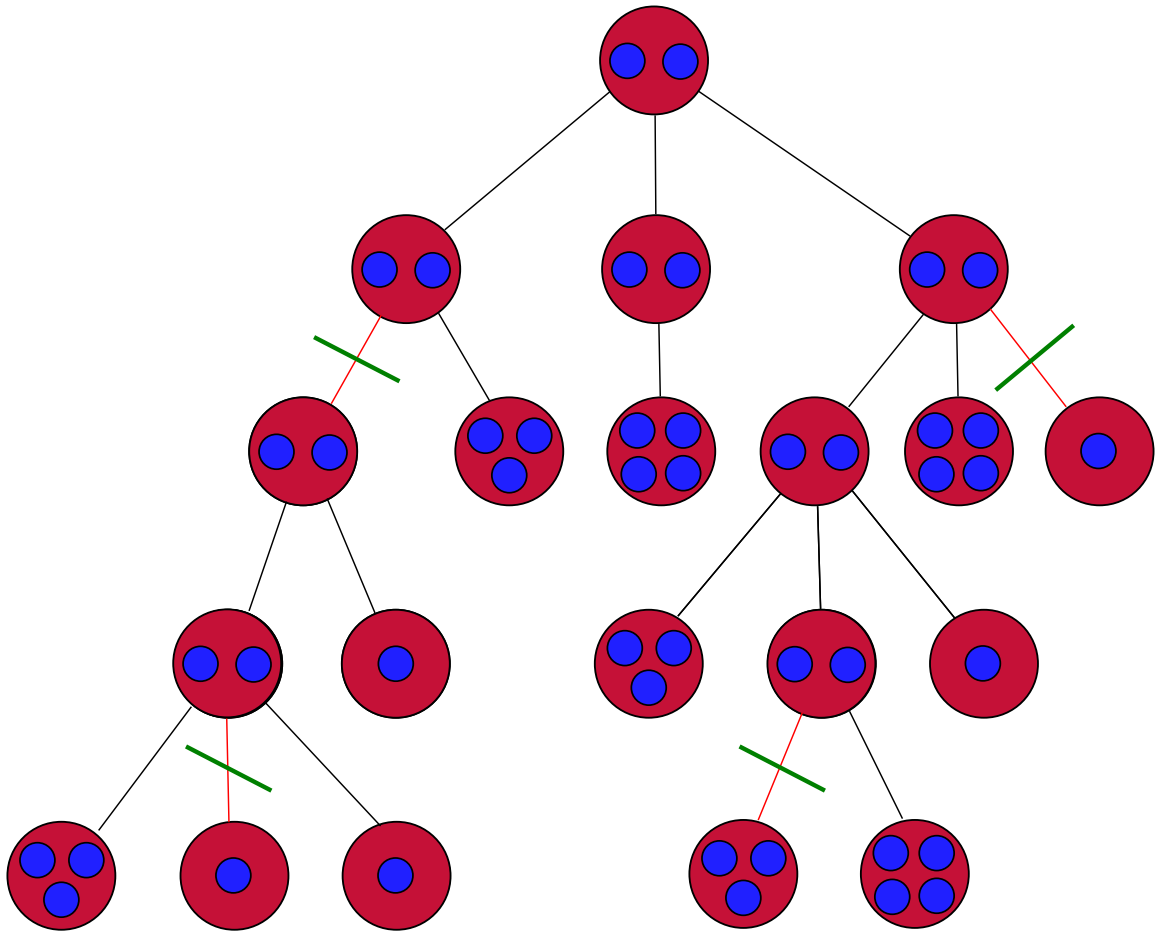
**(1)** Start with an arbitrary random split tree $T^n$



UPPSALA UNIVERSITET

# Bond Percolation on Split Trees

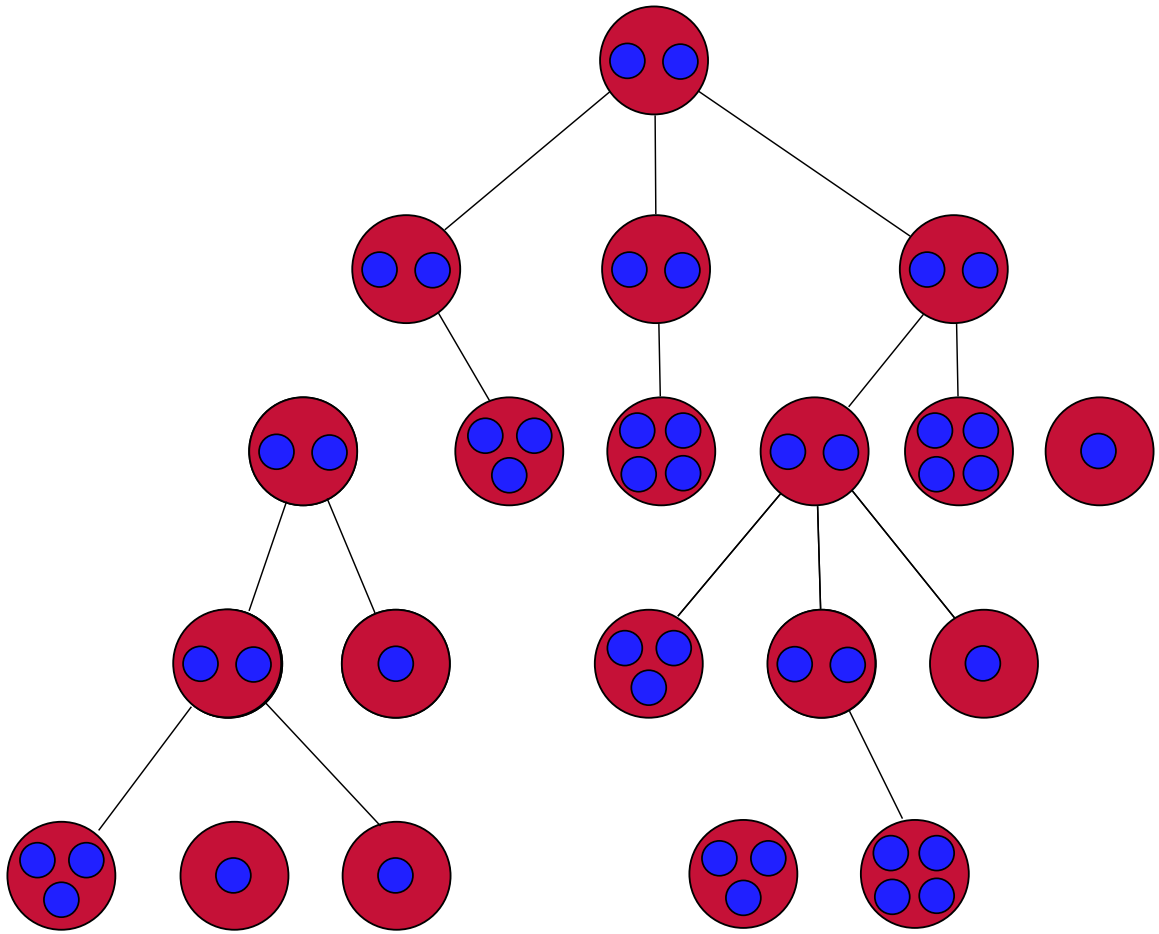
**(2)** Remove each edge with probability $1 - p_n$



UPPSALA UNIVERSITET

# Bond Percolation on Split Trees

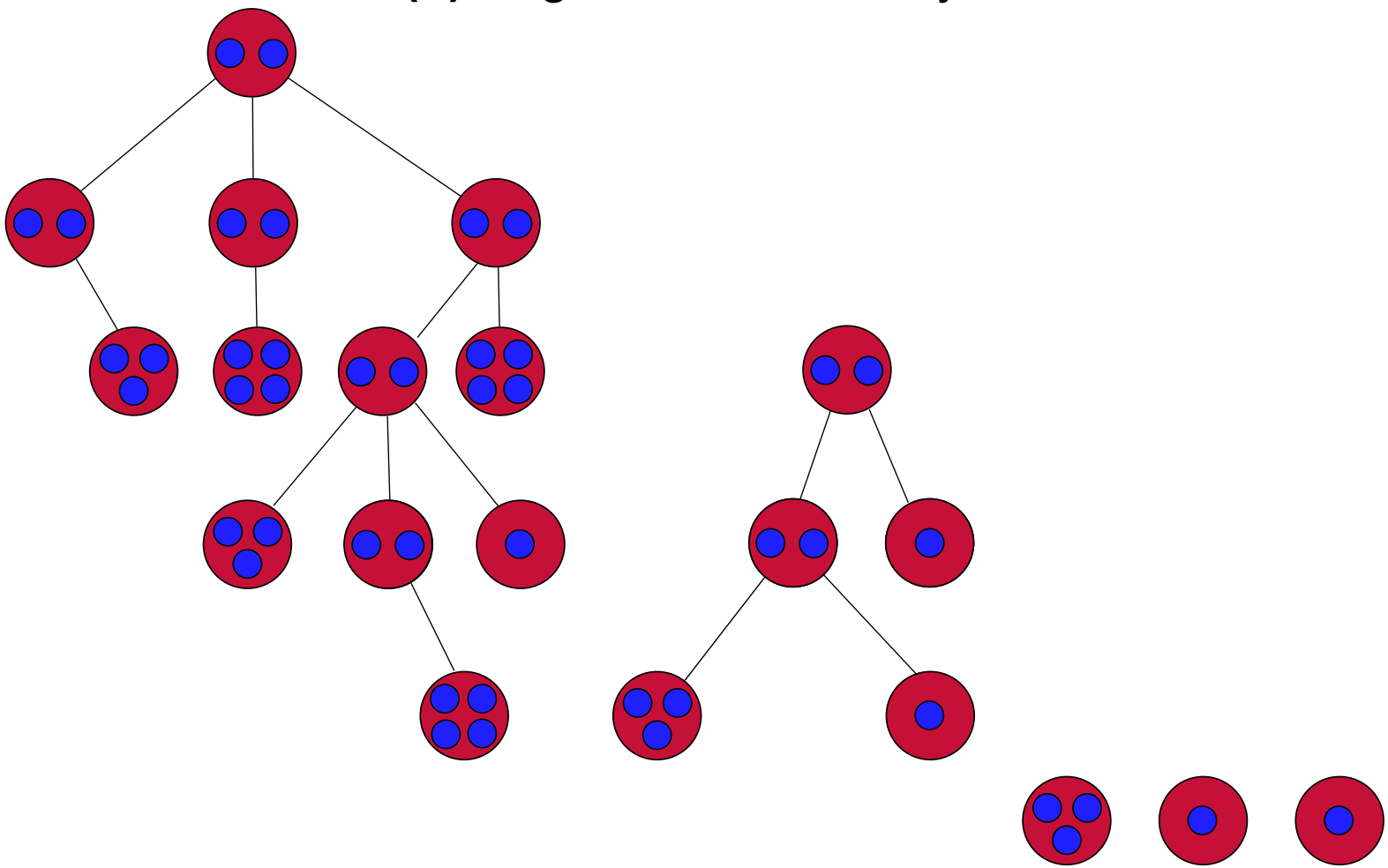
**(2)** Remove each edge with probability $1 - p_n$



UPPSALA UNIVERSITET

# Bond Percolation on Split Trees

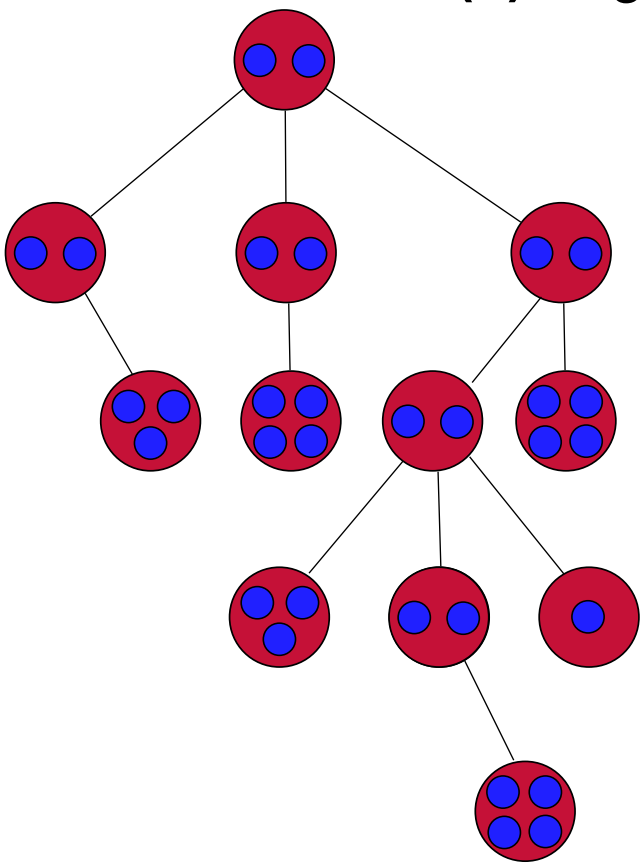
**(3)** Organize clusters by size



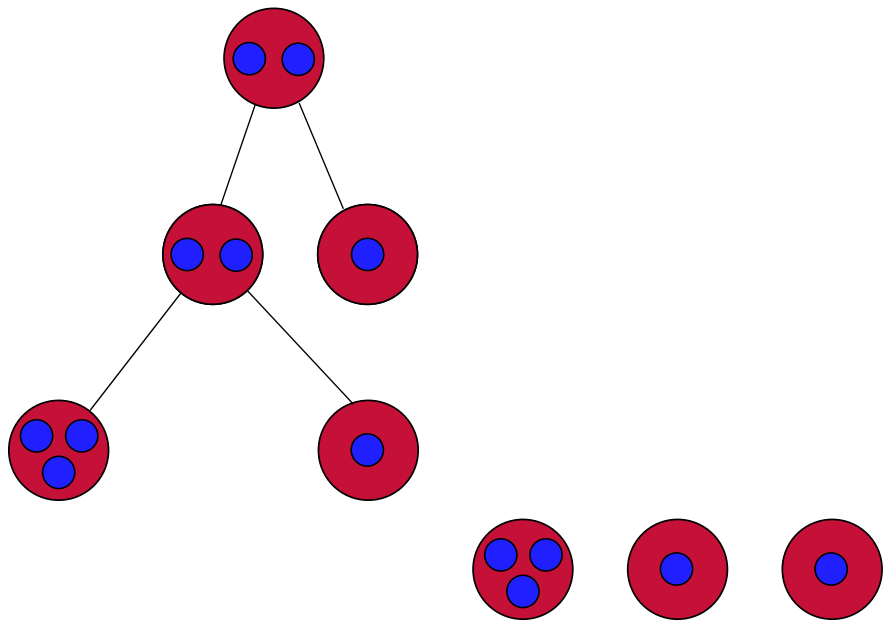
UPPSALA UNIVERSITET

# Bond Percolation on Split Trees

**(3)** Organize clusters by size



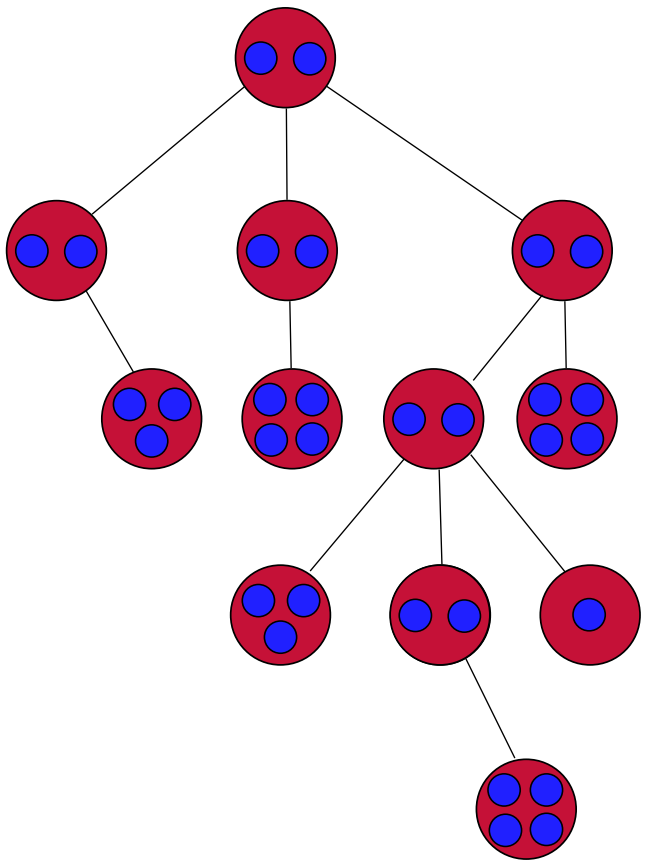
**Aim:** **To find the maximal cluster size (size of the root cluster) in an arbitrary random split tree after performing bond percolation**



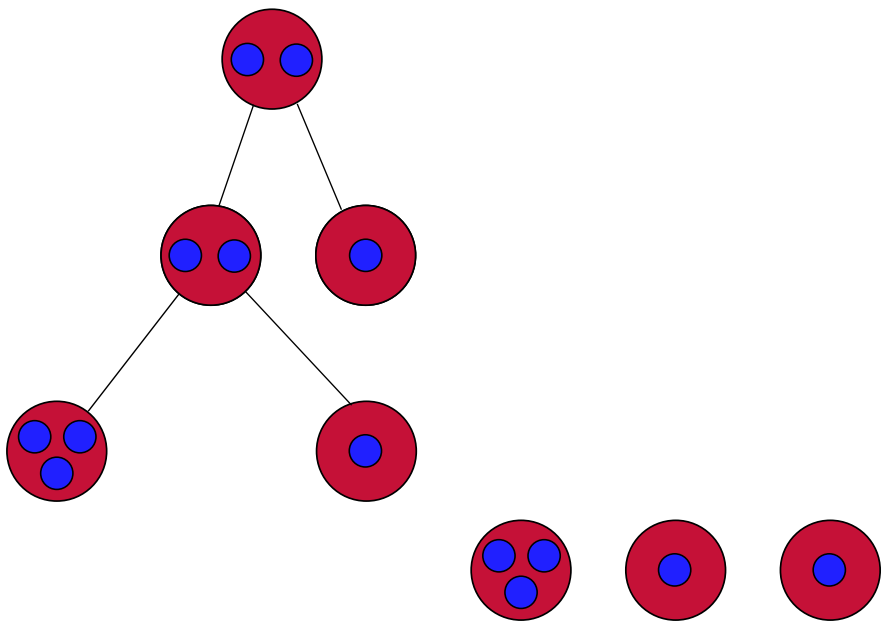
UPPSALA UNIVERSITET

# Bond Percolation on Split Trees

**(3)** Organize clusters by size



**Aim:** • **Expected Value (Convergence in Probability): To find the the first asymptotics of the maximal cluster size (root-cluster size)** • **Fluctuations: To find the limiting probability distribution of the maximal cluster size after normalization**



UPPSALA UNIVERSITET

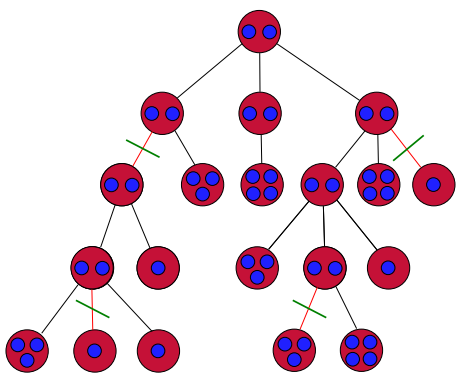
# Bond Percolation: Existence of a Giant Cluster?

Let $T^n$ be an arbitrary random split tree and perform bond percolation with parameter $p_n \in (0,1)$ (i.e., each edge is removed with probability $1 - p_n$ independently of the other edges).

**Question:** Do we have a giant cluster (containing the root) in $T^n$? More precisely, let $G_{p_n}$ be the size of the cluster that contains the root. If $p_n$ is large enough (this is called the supercritical regime) does

$$\frac{G_{p_n}}{n} \xrightarrow[n\to\infty]{p} \text{constant?}$$

If the answer is yes, then with high probability, there exists a *giant cluster*, that is of size comparable to that of the entire tree.




# Bond Percolation: The Size of the Maximal Cluster

Let $\hat{G}_{p_n} =$ # of balls of the root–cluster after percolation and $G_{p_n} =$ # of nodes of the root cluster. We consider the supercritical regime:$1 - p_n = \frac{c}{\ln n} + o(\ln^{-1} n)$, $c > 0$.
The next theorem shows that the maximal cluster is of the same order $n$ as the size of the whole tree i.e., we have a giant cluster.

**Theorem** (G. Berzunza, X.S. Cai and **C. Holmgren** *Latin American Journal of Probability and Mathematical Statistics 2022* )

*Let $\mathcal{V} = (V_1, V_2, \ldots, V_b)$ be the split vector and $\mu := b\mathbb{E}[-V_1 \ln V_1]$ (i.e., $\mu$ is a constant that depends on the split vector $\mathcal{V}$). Then*

(i) $\frac{\hat{G}_{p_n}}{n} \xrightarrow[n\to\infty]{p} e^{-c/\mu}$ *and*

(ii) $\frac{G_{p_n}}{n} \xrightarrow[n\to\infty]{p} \alpha e^{-c/\mu}$ .

*Moreover, the root–cluster is for both cases the unique giant cluster.*

UPPSALA UNIVERSITET

# The Size of the Giant Cluster after Bond Percolation

The next theorem describes the fluctuations of the giant cluster.

**Theorem** (G. Berzunza, X.S. Cai and **C. Holmgren** *Latin American Journal of Probability and Mathematical Statistics 2022* )

*Let* $\hat{G}_{p_n} =$ *# of balls in the maximal cluster (the root cluster) in a split tree. Then for* $1 - p_n = \frac{c}{\ln n} + o(\ln^{-1} n)$ *(the supercritical regime),*

$$\left(\frac{\hat{G}_{p_n}}{n} - e^{-\frac{c}{\mu}}\right) \ln n - \frac{c}{\mu} e^{-\frac{c}{\mu}} \ln\ln n \xrightarrow[n\to\infty]{d}$$

$$- \frac{c}{\mu} e^{-\frac{c}{\mu}} \left( Z + \ln\left(\frac{c}{\mu}\right) + \varsigma\mu + \frac{(\mu^2 - \sigma^2)(c+\mu)}{2\mu^2} - \gamma + 1 \right),$$

*where* $\mu$, $\sigma^2$ *and* $\varsigma$ *are constants depending on the split vector* $\mathcal{V}$, $\gamma$ *is the Euler constant, and* $Z$ *is a random variable with the continuous Luria-Delbrück distribution (a weakly 1-stable distribution).*

UPPSALA UNIVERSITET

# The $i$ largest Clusters after Bond Percolation

The following theorem tells, while the maximal cluster has a size "close to" $n$, the next largest clusters have only a size "close to" $\frac{n}{\ln n}$.

**Theorem** (G. Berzunza and **C. Holmgren** *Random. Struct. Alg. 2022* )

*Let $C_0 := \hat{G}_{p_n}$ be the number of balls of the root-cluster (maximal cluster) and $C_1 \geq C_2 \geq \cdots$, the sequence of the number of balls of the remaining clusters ranked in the decreasing order. Then*

$$n^{-1} C_0 \xrightarrow{p} e^{-c/\mu}, \quad \text{as } n \to \infty,$$

*Further, for every fixed $i \in \mathbb{N}$, we have the convergence in distribution*

$$\left( \frac{\ln n}{n} C_1, \ldots, \frac{\ln n}{n} C_i \right) \xrightarrow{d} (\mathrm{x}_1, \ldots, \mathrm{x}_i), \quad \text{as } n \to \infty,$$

*where $\mathrm{x}_1 > \mathrm{x}_2 > \cdots$ denotes the sequence of the atoms of a Poisson process on $(0, \infty)$ with intensity $c\mu^{-1} e^{-c/\mu} x^{-2} \mathrm{d}x$.*

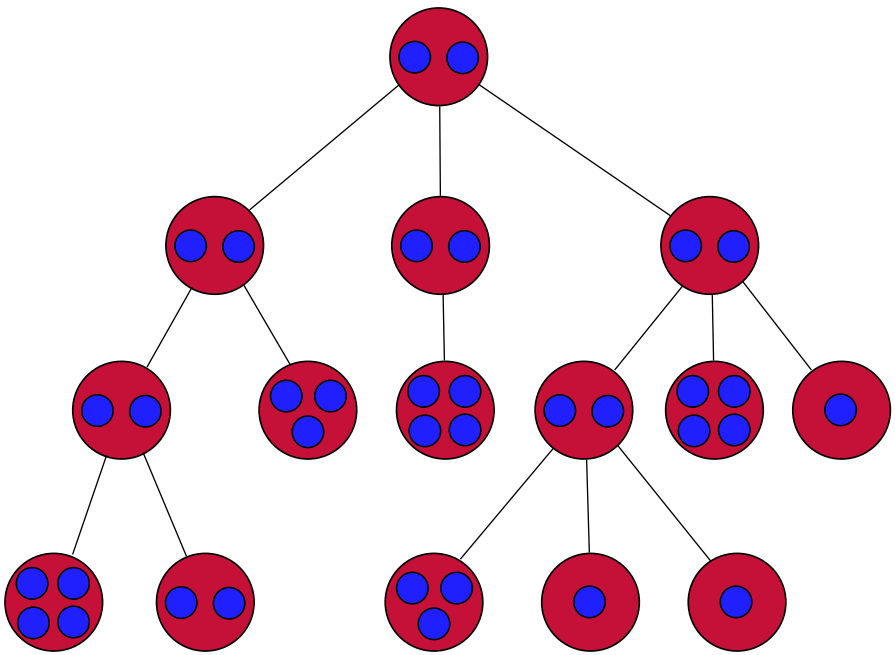
UPPSALA
UNIVERSITET

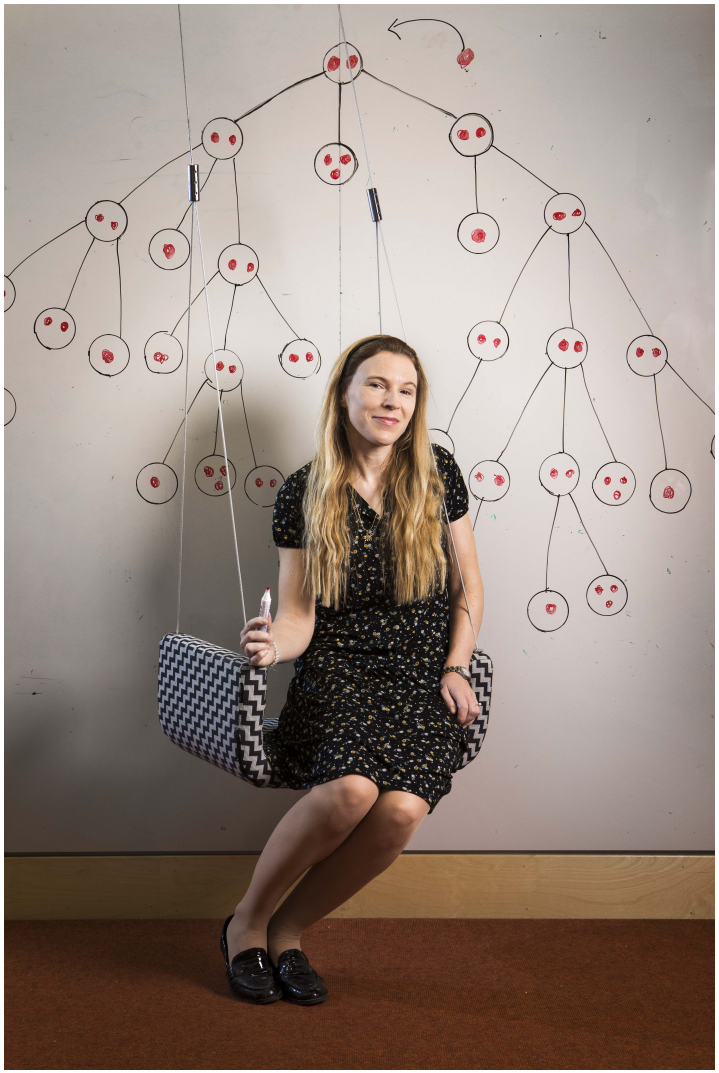
# Summary

- We have studied **split trees** which is a large class of trees of logarithmic height (many of these are used as **sorting algorithms** e.g., the binary search tree/Quicksort)
- By the introduction of **split trees** one can show general results for many trees at once, e.g., showing that most depths are concentrated around a logarithmic height $\mu^{-1} \ln n$, where the constant $\mu^{-1}$ depends on the type of split tree
- My results use **renewal theory** as a general method to analyze split trees
- Using renewal theory, sometimes in combination with other probabilistic methods such as the contraction method, we have shown general results for arbitrary random split trees e.g., on *the size (number of nodes), total path length, number of cuttings and inversions as well as on the size of the giant and smaller components after bond percolation*

UPPSALA UNIVERSITET

# Thanks for Your Attention!





*Cecilia Holmgren*