Solution of in-class exercise 1: Bounding a sequence

Using the usual approach of subtracting the recurrence for $n - 1$, we obtain that for $n \geq 3$,

$$x_n - x_{n-1} = \sqrt{n} - \sqrt{n-1} + 2x_{n-1}$$

holds and therefore

$$x_n = (\sqrt{n} - \sqrt{n-1}) + 3x_{n-1}.$$ 

Now since $x_1 > 0$ we deduce that $x_n > 0$ for all $n$, and then in particular

$$x_n > 3^{n-1}x_1 \notin O(2^n).$$

So the claim is FALSE.

Solution of in-class exercise 2: Random permutations

We prove the lemma by induction on the size $n$ of the insertion permutation (or equivalently, the resulting tree).

**Induction base case.** If $n = 0$ or $n = 1$, the lemma trivially holds.

**Induction step.** Let $n \geq 2$ and suppose that the lemma holds for all insertion permutations of size strictly less than $n$. Let $\pi = (\pi(1), \ldots, \pi(n))$ be a permutation drawn uniformly at random from $S_n$. The first element $\pi(1)$ will become the root of the tree $T_\pi$. Since the distribution of $\pi(1)$ is u.a.r. from $[n]$, the root of the tree is chosen uniformly at random, as required by the construction of $\hat{B}_{[n]}$.

Now let us tackle the two subtrees of the root: Let $k \in [n]$. We want to show that, conditioned on $\pi(1) = k$, the distribution of the left subtree of the root is the same as $\hat{B}_{[1, \ldots, k-1]}$, and the distribution of the right subtree of the root is the same as $\hat{B}_{[k+1, \ldots, n]}$. To this end, let $\pi^-$ be the sequence of elements in $\pi$ smaller than $k$, and let $\pi^+$ be the sequence of elements in $\pi$ larger than $k$. Note that the insertion sequence will send the keys in $\pi^-(\pi^+)$ in exactly this order to the left (right) subtree of the root. Since $\pi^-$ and $\pi^+$ are uniformly random permutations of their respective element sets $\{1, \ldots, k-1\}$ and $\{k+1, \ldots, n\}$, we can apply the induction hypothesis to obtain that the left and right subtrees of the root are distributed just as we stated.

We conclude that this process produces a tree $T_\pi$ that is distributed like $\hat{B}_{[n]}$. 

Solution 1: A Random Tree? How Random?

According to Lemma 1.1 in the script, the probability of the tree is

\[ \frac{1}{2 \cdot 1 \cdot 7 \cdot 1 \cdot 3 \cdot 1 \cdot 4} = \frac{1}{168} . \]

What do the extremal examples for the probability of a tree look like? In the case of 7 nodes, we consider the trees of height 7 − 1 = 6 (which yields the smallest probability \( \frac{1}{7!} = \frac{1}{5040} \)) and the perfectly balanced tree of height 2 (which yields the largest probability \( \frac{1}{3^2 \cdot 1^7} = \frac{1}{63} \)).

Since the problem on 7 nodes is finite, we could list all cases as a proof. However, there are 429 search trees on 7 nodes which would make the proof lengthy.

Alternatively, for the case of smallest probability, we can argue using Exercise 1.5 from the script (which is the same as in-class exercise 2):

**Lemma 1.** If keys are inserted in a uniformly random order, the resulting search tree is distributed like \( B_s \).

It follows from the lemma that every random search tree is generated by at least one permutation on \( n \) elements (the probability of a random search tree is actually the number of permutations generating it divided by \( n! \)). Thus, the probability of a random search tree is at least \( 1/n! \) and since the extremal example of height \( n − 1 \) achieves exactly this probability, it has smallest probability. This holds for all \( n \).

For the case of largest probability, let \( p_n \) denote the maximum probability for a random search tree with \( n \) vertices. Then we get the following recursion beginning with, by convention, \( p_0 = 1 \) (where \( k \) denotes the rank of the root).

\[ p_n = \frac{1}{n} \cdot \max_{1 \leq k \leq n} (p_{k-1}p_{n-k}) . \]

The first few values for this maximum probability can easily be computed, namely \( p_1 = 1, p_2 = \frac{1}{2}, p_3 = \frac{1}{3}, p_4 = \frac{1}{4}, p_5 = \frac{1}{5}, p_6 = \frac{1}{6}, p_7 = \frac{1}{7}, p_8 = \frac{1}{8} \). This already proves that the perfectly balanced tree on 7 nodes has indeed largest probability. To generalize this case to larger \( n \), one would need to work somewhat harder.

Solution 2: Very Deep Nodes

Let \( N_{\text{deep}} \) denote the number of nodes of depth \( n − 1 \). We observe that a binary search tree \( B \) for \( n \) vertices has one node of depth \( n − 1 \) if and only if it is a path of length \( n − 1 \). Let \( p_n \) denote the probability that there is a node of depth \( n − 1 \). We have

\[ E[N_{\text{deep}}] = p_n \cdot 1 + (1 − p_n) \cdot 0 = p_n . \]
So it remains to compute \( p_n \). Clearly, \( p_1 = 1 \). For \( n \geq 2 \) we apply induction. Note that if a tree \( T \) is a path then its root is either the smallest or the largest key. Hence

\[
\begin{align*}
p_n &= \Pr[\text{rk(root) = 1}] \cdot \Pr[\text{one node has depth } n - 1|\text{rk(root) = 1}] + \\
&\quad \Pr[\text{rk(root) = n}] \cdot \Pr[\text{one node has depth } n - 1|\text{rk(root) = n}] \\
&= \frac{1}{n} p_{n-1} + \frac{1}{n} p_{n-1} \\
&= \frac{2}{n} \cdot p_{n-1}
\end{align*}
\]

Induction yields that \( p_n = \frac{2^{n-1}}{n!} \cdot p_1 = \frac{2^{n-1}}{n!} \) and so we are done.

**Remark.** Note that this also provides us with the number of trees on \( n \) nodes that have height \( n - 1 \): since each such tree is a path of length \( n - 1 \), for each such tree there is exactly one ordering of \( n \) keys which produces this search tree (since a parent must always be inserted before its child). So each tree of height \( n - 1 \) separately has a probability of \( 1/n! \). Since in total \( p_n = 2^{n-1}/n! \), we conclude that there must be exactly \( 2^{n-1} \) trees of this type.

Of course, we could have found this number more easily: the number of trees on \( n \) nodes that are a single path is \( 2^{n-1} \) simply because for each edge (of which there are \( n - 1 \) many) we can decide whether it should point to the left or to the right.

**Solution 3: High Trees**

By an \((n, d)\)-tree we denote a tree on \( n \) vertices of height \( d \). Let \( M_n \) denote the number of \((n, n - 2)\)-trees and let \( M'_n \) denote the number of \((n, n - 1)\)-trees. Our goal is to compute \( M_n \). Clearly, \( M_1 = 0 \), \( M_2 = 0 \) and \( M_3 = 1 \). Now for \( n \geq 4 \), let us consider two ways to proceed.

**Variant 1: by induction.** Note that if a tree \( T \) in \( B_{1...n} \) has height \( n - 2 \) then

- *either* the root of \( T \) is in \( \{1, n\} \) and the subtree of the root is an \((n - 1, n - 3)\)-tree
- *or* the root of \( T \) is in \( \{2, n - 1\} \) and one of the subtrees of the root is an \((n - 2, n - 3)\)-tree.

As \( n \geq 4 \), the numbers \( 1, 2, n - 1, n \) are distinct. So for \( n \geq 4 \),

\[
M_n = 2 \cdot M_{n-1} + 2 \cdot M'_{n-2}, \quad (1)
\]

As a direct consequence of Exercise 3 [Remark] we have \( M'_n = 2^{n-1} \).

Equation (1) then yields that for \( n \geq 4 \),

\[
M_n = 2 \cdot M_{n-1} + 2 \cdot 2^{n-3} = 2 \cdot M_{n-1} + 2^{n-2}.
\]

By induction we obtain that \( M_n = 2^i \cdot M_{n-i} + i \cdot 2^{n-2} \) for \( i \leq n - 3 \). Hence for \( n \geq 4 \),

\[
M_n = 2^{n-3} \cdot M_3 + (n - 3) \cdot 2^{n-2} = 2^{n-3} + (n - 3) \cdot 2^{n-2} = (2n - 5) \cdot 2^{n-3}.
\]
Variant 2: direct counting. We can count the number of trees on \( n \) vertices of height \( n - 2 \) directly for \( n \geq 3 \): if a tree \( T \) in \( B_1,\ldots,n) \) has height \( n - 2 \) then it consists of a path of height \( n - 2 \) plus a leaf that can be attached at any of the first \( n - 2 \) (out of the \( n - 1 \)) vertices of that path.

From Exercise 2 (1.9), we know that the number of paths on \( n - 1 \) vertices is \( 2^{n-2} \) for \( n \geq 1 \). Multiplying by the number of possibilities for attaching the additional leaf, we obtain \( (n - 2)2^{n-2} \).

![Figure 1: Trees that are counted twice.](image)

This number, however, counts a certain kind of trees twice: All trees that have two leaves of depth \( n - 2 \) (cf. Figure 1). How many such trees are there? Such a tree consists of a path on \( n - 2 \) vertices plus two leaves attached to its end, so there are equally many of them as there are paths on \( n - 2 \) vertices. Again using Exercise 2 (1.9), we obtain \( 2^{n-3} \) for this number. So in total, we obtain \( (n - 2)2^{n-2} - 2^{n-3} \) for \( n \geq 3 \) (which is the same as we obtained in Variant 1).

Solution 4: Solving Recurrences

1. First we compute that \( a_1 = 1 \) and \( a_2 = \frac{3}{2} \). Now for \( n \geq 3 \), we multiply the recurrence relation by \( n \) and write it once for \( n \) and once for \( n - 1 \). This yields

\[ na_n = n + \sum_{i=1}^{n-1} a_i \]  \hspace{1cm} (2)

and

\[ (n - 1)a_{n-1} = (n - 1) + \sum_{i=1}^{n-2} a_i \]  \hspace{1cm} (3)

Now subtracting (3) from (2), we obtain

\[ na_n - (n - 1)a_{n-1} = 1 + a_{n-1} \]
and thus
\[ a_n = \frac{1}{n} + a_{n-1}. \]

This recursion can easily be telescoped from which we obtain
\[ a_n = \frac{1}{n} + \frac{1}{n-1} + \ldots + \frac{1}{3} + \sum_{i=1/2+1} a_2 = H_n. \]

Therefore, \( a_n = H_n \) for all \( n \in \mathbb{N} \).

(2) We first compute that \( b_1 = 1 \) and \( b_2 = 3 \). Now for \( n \geq 3 \)
\[
b_n = 2 + \sum_{i=1}^{n-1} b_i \tag{4}
\]
and
\[
b_{n-1} = 2 + \sum_{i=1}^{n-2} b_i \tag{5}
\]
Now subtracting (5) from (4), we obtain
\[ b_n - b_{n-1} = b_{n-1}, \]
therefore
\[ b_n = 2b_{n-1} \]
and thus
\[ b_n = 2^{n-2}b_2 = 3 \cdot 2^{n-2}. \]
Therefore, \( b_1 = 1 \) and \( b_n = 3 \cdot 2^{n-2} \) for all \( n \geq 2 \).

(3) We first compute that \( c_0 = 0 \) and \( c_1 = 0 \). Then for \( n \geq 2 \), we first note that
\[
\sum_{i=1}^{n} \frac{c_{i-1} + c_{n-i}}{2} = \sum_{i=1}^{n} c_{i-1}
\]
which then allows us to write the simpler recurrences for \( c_n \) and \( c_{n-1} \)
\[
c_n = n - 1 + \sum_{i=1}^{n} c_{i-1} \tag{6}
\]
and
\[
c_{n-1} = n - 2 + \sum_{i=1}^{n-1} c_{i-1} \tag{7}
\]
If we now subtract (7) from (6), then
\[ c_n - c_{n-1} = 1 + c_{n-1} \]
and thus
\[ c_n = 1 + 2c_{n-1}. \]
For telescoping, it turns out to be convenient to divide the recurrence by $2^n$, then we have

$$\frac{c_n}{2^n} = \frac{1}{2^n} + \frac{c_{n-1}}{2^{n-1}}$$

and we can telescope for $c_n/2^n$, yielding

$$\frac{c_n}{2^n} = \frac{1}{2^n} + \frac{1}{2^{n-1}} + \cdots + \frac{1}{2^2} + \frac{c_1}{2^1} = \frac{1}{2} - \frac{1}{2^n}.$$ 

Therefore, $c_0 = 0$ and $c_n = 2^{n-1} - 1$ for $n \in \mathbb{N}$.

(4) We compute that $d_0 = 0$ and $d_1 = 1$. Then for $n \geq 2$, we may instantiate

$$d_n = 1 + 2 \sum_{i=0}^{n-1} (-1)^{n-i} d_i \quad (8)$$

and

$$d_{n-1} = 1 + 2 \sum_{i=0}^{n-2} (-1)^{n-1-i} d_i \quad (9)$$

This time, adding the recurrences $(8) + (9)$ turns out to be more helpful as it yields

$$d_n + d_{n-1} = 2 - 2d_{n-1}$$

and thus

$$d_n = 2 - 3d_{n-1}.$$ 

To simplify telescoping, we rearrange this to

$$d_n - \frac{1}{2} = -3 (d_{n-1} - \frac{1}{2})$$

and then use the substitution

$$f_n := d_n - \frac{1}{2}$$

from which

$$f_n = -3 f_{n-1}.$$ 

Telescoping now immediately yields

$$f_n = f_1 (-3)^{n-1},$$

thus

$$f_n = \frac{1}{2} (-3)^{n-1}$$

and so undoing the substitution we end up with

$$d_n = \frac{1}{2} (-3)^{n-1} + \frac{1}{2}.$$ 

In conclusion, $d_0 = 0$ and $d_n = \frac{1}{2}(1 + (-3)^{n-1})$ for $n \in \mathbb{N}$. 
For \( n \geq 1 \), we have the recurrence

\[
e_n = 1 + ne_{n-1}.
\]

It is convenient to divide this recurrence by \( n! \) as then

\[
e_n/n! = \frac{1}{n!} + \frac{e_{n-1}}{(n-1)!}
\]

is a simple recurrence for the series \( e_n/n! \). Telescoping it yields

\[
e_n/n! = \frac{1}{n!} + \frac{1}{(n-1)!} + \ldots + \frac{1}{1!} + \frac{e_0}{0!} = \sum_{i=0}^{n} \frac{1}{i!}
\]

for all \( n \geq 1 \). Therefore,

\[
e_n = \left( \sum_{i=0}^{n} \frac{1}{i!} \right) n!
\]

for all \( n \in \mathbb{N} \) (note that by convention, \( 0! = 1 \)).

The above expression may be explicit but it still involves a sum, so we should routinely ask whether there is way to simplify it. The expression in the sum of course makes us think of the function \( \exp(-) \). In fact, we know that

\[
\sum_{i=0}^{\infty} \frac{1}{i!} = e = 2.71..., \]

and thus in the case of our sequence, \( e_n < e_n! \). Due to the nature of the recursion, however, \( e_n \) is always an integer, thus we also have \( e_n \leq \lfloor e_n! \rfloor \). Let us compare how close this bound is to the truth.

<table>
<thead>
<tr>
<th>( n )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( e_n! )</td>
<td>2.71...</td>
<td>2.71...</td>
<td>5.44...</td>
<td>16.31...</td>
<td>65.24...</td>
</tr>
<tr>
<td>( \lfloor e_n! \rfloor )</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>16</td>
<td>65</td>
</tr>
<tr>
<td>( e_n )</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>16</td>
<td>65</td>
</tr>
</tbody>
</table>

So it seems the bound is tight starting \( n = 1 \). And indeed, if we check,

\[
e_n! - e_n = n! \sum_{i=n+1}^{\infty} \frac{1}{i!} = \frac{1}{n+1} + \frac{1}{(n+1)(n+2)} + \ldots
\]

is decreasing in \( n \). Since it is below 1 for \( n = 1 \), it stays below 1 for all \( n \).

We have established

\[
e_n = \begin{cases} 1, & \text{if } n = 0; \\ \lfloor e_n! \rfloor, & \text{otherwise}. \end{cases}
\]
Solution 5: Descendants of the Smallest Key

Variant 1: Computation via conditioning on the rank of the root. The usual way, we first obtain

\[ \mathbb{E}[S_n] = \sum_{i=1}^{n} \mathbb{E}[S_n | \text{rk(root)} = i] \cdot \mathbb{P}[\text{rk(root)} = i] \]

where

\[ (*) = \begin{cases} 
    n, & \text{if } i = 1, \\
    \mathbb{E}[S_{i-1}], & \text{otherwise}
\end{cases} \]

Denote \( s_n := \mathbb{E}[S_n] \). Then this yields a recurrence of the form.

\[ s_n = \frac{1}{n} \left( n + \sum_{i=1}^{n} s_{i-1} \right), \]

holding for all \( n \geq 1 \). As we are, by now, proficient in solving recurrences, let us multiply by \( n \) and then instantiate the recurrence for both \( n \) and \( n - 1 \) so that for \( n \geq 1 \) we have

\[ ns_n = n + \sum_{i=1}^{n-1} s_i \quad (10) \]

and for \( n \geq 2 \), we get

\[ (n - 1)s_{n-1} = n - 1 + \sum_{i=1}^{n-2} s_i \quad (11) \]

Then subtracting (11) from (10), we obtain

\[ ns_n - (n - 1)s_{n-1} = 1 + s_{n-1}. \]

Rearranging and dividing by \( n \), this yields

\[ s_n = \frac{1}{n} + s_{n-1}. \]

We are familiar with this recursion and know that telescoping it out will produce \( s_n = H_n \).

Variant 2: Computation via indicator variables. Alternatively, we can use the well-known indicator variables

\[ A_i^j := \text{[node } j \text{ is an ancestor of node } i]\]

In that case we obviously have

\[ S_n = \sum_{i=1}^{n} A_i^j \]

\[ \Rightarrow s_n = \mathbb{E}[S_n] = \mathbb{E} \left[ \sum_{i=1}^{n} A_i^j \right] = \sum_{i=1}^{n} \mathbb{E}[A_i^j]. \]
Those expectations have been computed in the lecture notes where we have obtained that

$$E[A_i] = \frac{1}{|i-j|+1}$$

and thus

$$\Rightarrow E[A_i] = \frac{1}{i-1+1} = \frac{1}{i}.$$ 

Therefore, this variant, too, yields

$$s_n = \sum_{i=1}^{n} E[A_i] = \sum_{i=1}^{n} \frac{1}{i} = H_n.$$