Exercise Set Solutions #5
"Discrete Mathematics" (2025)

E1. **(a)** Denote b_n the n-th Catalan number. Show that

$$b_n = \frac{1}{n+1}\binom{2n}{n}$$

by completing the remaining calculation from the lecture.

> ***Solution:*** From the lecture, we have that
>
> $$b_n = -\frac{1}{2}(-4)^{n+1}\binom{\frac{1}{2}}{n+1}$$
>
> where
>
> $$\begin{aligned}\binom{\frac{1}{2}}{n+1} &= \frac{\frac{1}{2}\left(\frac{1}{2}-1\right)\left(\frac{1}{2}-2\right)\ldots\left(\frac{1}{2}-n\right)}{(n+1)!} \\ &= \frac{\frac{1}{2}\left(-\frac{1}{2}\right)\left(-\frac{3}{2}\right)\ldots\left(\frac{1-2n}{2}\right)}{(n+1)n!} \\ &= \frac{1}{2(n+1)}\left(-\frac{1}{2}\right)^n\left(\frac{1.3.5\ldots(2n-1)}{n!}\right)\left(\frac{2.4.6\ldots(2n)}{2^n n!}\right) \\ &= \frac{1}{2}\left(-\frac{1}{4}\right)^n\left(\frac{1}{n+1}\left(\frac{(2n)!}{n!n!}\right)\right)\end{aligned}$$
>
> A small rearranging then gives the final answer.

(b) Verify the formula for the number of binary trees with $0 \leq n \leq 3$ vertices by drawing all binary trees with three or fewer vertices.

> ***Solution:*** There is only one binary tree with one vertex, two with two vertices (the second vertex can be either on left or on the right) and 5 binary trees with 3 vertices (see lecture notes, page 13 Figure 1)

E2. Show that the solution of each of the counting problems below is the same. Try to list the objects in each case for $n = 3$ or $n = 4$.

(a) Number of paths a drunk frog moving away from a wall can take in $2n$ steps given that it returns to the same place. In each step, the frog can either move one inch away from the wall or one inch towards the wall and it starts and ends at 0 distance from the wall. Each step of the frog is of equal distance and the frog cannot jump through the wall.

(b) The number of ways in which n coins of 1 franc and n coins of 2 franc can be distributed among $2n$ people standing in a coffee machine queue in the following way: Each person gets exactly one coin, and when they start buying a coffee in the order they are standing, the coffee machine never runs out of change. The coffee costs 1 franc and the machine has no coins inside to begin with.

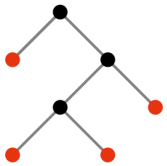
(c) The number of ways you can make arrange n pairs of parenthesis { "(", ")" } such that they form a valid expression. For example, (())() is a valid expression but ())() (is not a valid expression because the left and right-parentheses are not correctly matched.

(d) The number of ways a student can compute the product

$$A_1 \times A_2 \times A_3 \times \cdots \times A_{n+1}$$

of $n+1$ square matrices $\{A_i\}_{i=1}^{n+1}$ of the same size if in each step he can multiply only two adjacent matrices.

(e) The number of "full" binary trees with $n+1$ leaves. A full binary tree is a binary tree in which either both the subtrees at each vertex are empty or both are non-empty. When both the subtrees are empty, such a vertex is called a leaf. Below is an example of such a tree with 4 leaves.



(f) The number of binary trees with n unlabelled vertices.

Solution: All of the above sets are in bijection with each other. Since they are all finite sets, it is also possible to show by some other means that the number of ways to count them is $\frac{1}{n+1}\binom{2n}{n}$. For example, we show that the number of ways of counting (c) satisfies the same recursive relations as Catalan numbers. Let b_n be the number of ways of making a valid parentheses expressions of length $2n$. Then clearly $b_0, b_1 = 1$. Also, any parenthesis expression can be uniquely decomposed as $(w_1)\, w_2$ where w_1 and w_2 are valid parentheses expressions themselves. If w_1 is of length $2k$, then w_2 must be of length $2(n-k-1)$. Hence w_1 can be chosen in b_k ways and w_2 in $b_{(n-1)-k}$ and hence $b_n = \sum_{k=0}^{n-1} b_k b_{(n-1)-k}$.

Below, we will show enough bijections to make all the sets equal to each other, but there can be lots of more solutions and bijections.

[(**a**) $\Leftrightarrow$ (**c**)] The only way a frog can return to the same place is if the number of forward steps and the number of backward steps are both n. To avoid crossing the wall, for any k, after k steps the number of forward steps will have to be greater than or equal to the number of backward steps. Hence, for each forward step the frog takes write a left-parenthesis and for each backward step write a right-parenthesis. This way, for any k there will never be more left-parenthesis than right-parenthesis after parsing the first k letters of the parenthesis expression and the number of left and right parenthesis will be equal.

For the reverse bijection, each parenthesis expression corresponds to a sequence of steps of a frog.

[(**b**) $\Leftrightarrow$ (**c**)] The coffee machine runs out change after k steps only if the number of 2 -franc payments has exceeded the number of 1 -franc payments. Hence, to make sure that this never happens, after any k number of people in the queue the number of 1 franc coins must be greater than or equal to 2 franc coins.

To get the correspondence from such a distribution of coins to a parenthesis expression, put a left-parenthesis at the k th position of the expression if and only if a 1-franc coin is given to the k th person on the queue.

[(**d**) $\Leftrightarrow$ (**c**)] This is somewhat non-intuitive. Here are some hints: Use the parenthesis { " [", "]"} to indicate the order of operations. For example, the number of ways to multiply 4 matrices is

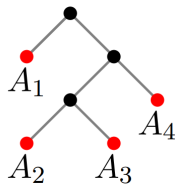
the following.

$$[[[A_1 \times A_2] \times A_3] \times A_4], [[A_1 \times [A_2 \times A_3]] \times A_4], [[A_1 \times A_2] \times [A_3 \times A_4]]$$
$$[A_1 \times [[A_2 \times A_3] \times A_4]], [A_1 \times [A_2 \times [A_3 \times A_4]]]$$

Note that the number of times each character { "[", "]", " × " } appears in any of these expressions is exactly $n = 3$ times. This is going to be true for a general n as well. Now replace each "[" with a "(", each " × " with a ")" and delete all " A_i " and "]".

$$((())), (()()), (())(), ()((()), ()()()$$

[(**d**) ⇔ (**e**)] Putting the matrices at the leaves of the binary tree, we get a way to multiply the $n + 1$ matrices for each tree and vice versa. For example, the following corresponds with $A_1 \times ((A_2 \times A_3) \times A_4)$.



[(**e**) ⇔ (**f**)] We can show by induction that a full binary tree with $n + 1$ leaves has will have exactly n non-leaf vertices, i.e. $2n - 1$ vertices in total. It is clear when $n = 0$, since we have a single leaf. When there are $n + 1 > 1$ leaves, the root cannot be leaf otherwise there is only 1 leaf. So the root has a non-empty left-subtree and a non-empty right sub-tree. Let the two have l_1 and l_2 leaves respectively so we have $l_1 + l_2 = n + 1$. If both $l_1, l_2 < n + 1$ then we are done by induction because then the left-subtree has $l_1 - 1$ non-leaf vertices and the right-subtree has $l_2 - 1$ non-leaf vertices and together with the root we have $l_1 - 1 + l_2 - 1 + 1 = n + 1 - 1 = n$ non-leaf vertices. Otherwise, either $l_1 = 0$ or $l_2 = 0$, which is impossible since the root cannot have any empty subtree.

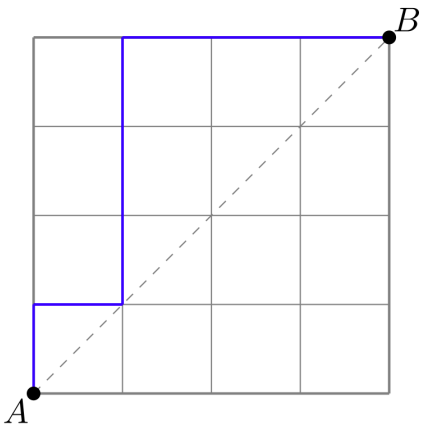
Start with a full binary tree with $n + 1$ leaves and whenever a vertex has a leaf as a leftsubtree or right-subtree, replace that leaf with the empty subtree. This way, we eliminate all $n + 1$ leaves and we obtain a binary tree with n vertices. For the inverse mapping, to a binary tree replace every empty subtree with a binary tree with a single vertex (i.e, add a leaf at every empty vertex). Try to show that the two maps are inverse of each other.

[(**c**) ⇔ (**e**)] This is also not so direct. So we give hints. Each sequence of parentheses can be used to construct a full binary tree in the following way. Start with a with no subtrees and read the parantheses sequence from left to right. We will recursively add subtrees to the root in the following way. - Whenever a "(" is encountered add a vertex on the left of the vertex added the last and move to the newly added vertex to possibly add more subtrees in the subsequent steps - Whenever a ")" is encountered, declare the current vertex as a leaf (it can have no more subtrees now) and keep moving up a vertex until a vertex with no right-subtree is reached. When such a vertex is encoutered, add a vertex to the right of the subtree and move to that vertex to possibly add more subtrees in the subsequent steps. One can show that after k steps

of this process, the only empty left-subtree can be at the last added vertex and the number of empty right-subtrees are exactly the number of left-parenthesis encountered so far minus the number of right-parentheses encountered so far. Hence the algorithm does not run into an impossible scenario when a ")" due to this condition.

To get the inverse mapping that gives a parantheses expression from a full binary tree, do a "depth-first search" on the full binary tree prioritizing left-subtrees. That is, start from the root vertex and travel in the following manner while writing a parantheses seqeunce from left to right. - Whenever a left-subtree exists, move to the root vertex of that subtree and add a "(" to the sequence so far. - If no left-subtree exists, move up a vertex until a right subtree exists and then move to the top vertex of the right subtree of the first such vertex. Add a ")" to the sequence of parantheses. One can then check that both the algorithms are inverse of each other.

E3. Consider an $n \times n$ chessboard:



Consider the shortest paths from the corner A to the corner B following the edges of the squares (each of them consists of $2n$ edges).

(a) How many such paths are there?

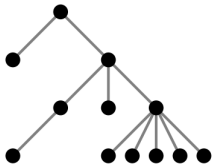
Solution: (Let $(0,0)$ be the position of A and (n,n) be the position of B. Starting from $(0,0)$, each such path can be given by a sequence of $(1,0)$ or $(0,1)$ vectors. To get to (n,n), there must be exactly n many $(0,1)$ vectors and n many $(1,0)$ vertices in the path. The number of ways to arrange this is then $\binom{2n}{n}$.

(b) Show that the number of paths that never go below the diagonal (the line AB) is exactly b_n, i.e. the Catalan number. One such path is drawn in the figure.

Solution: A path that starts with $(0,0)$ will cross the line only if the number of $(1,0)$ vectors in the path exceeds the number of $(0,1)$ vectors. Hence, this is in bijection with (a), (b), and (c) of the previous question.

E4. Let us denote the t_n to be the number of rooted plane trees with n vertices. They are defined as follows:

Each tree has a distinguished vertex called root along with a finite **ordered set** of trees which are called subtrees. For example, the following is a tree



(a) Take $t_0 = 0$. Show that the generating function $t(x) = \sum_{i=0}^{n} t_i x^i$ satisfies

$$t(x)(1 - t(x)) = x.$$

We set $t_0 = 0$ and $t_1 = 1$. The root can then have $r \in \{0, 1, 2, 3 \ldots\}$ number of subtrees. Note that we distinguish the order of the subtrees (if they are different) but compared to the binary trees we do not have empty subtrees. The sum of the vertices of the subtrees then should be $n - 1$ and each subtree should be non-empty. So we get

$$t_{n+1} = \sum_{r=0}^{\infty} \sum_{\substack{(k_1,k_2,\ldots,k_r)\in\mathbb{Z}_{\geq 1}^r \\ k_1+k_2+\cdots+k_r=n}} t_{k_1} t_{k_2} \ldots t_{k_r}$$

This tells us that

$$\begin{aligned}
\sum_{n=0}^{\infty} t_{n+1} x^{n+1} &= \sum_{n=0}^{\infty} \sum_{r=0}^{\infty} \sum_{\substack{(k_1,k_2,\ldots,k_r)\in\mathbb{Z}_{\geq 1}^r \\ k_1+k_2+\cdots+k_r=n}} t_{k_1} t_{k_2} \ldots t_{k_r} x^{n+1} \\
&= x \left(\sum_{n=0}^{\infty} \sum_{r=0}^{\infty} \sum_{\substack{(k_1,k_2,\ldots,k_r)\in\mathbb{Z}_{\geq 1}^r \\ k_1+k_2+\cdots+k_r=n}} t_{k_1} t_{k_2} \ldots t_{k_r} x^{n} \right) \\
&= x \left(\sum_{r=0}^{\infty} \sum_{(k_1,k_2,\ldots,k_r)\in\mathbb{Z}_{\geq 1}^r} \sum_{n=0}^{\infty} t_{k_1} t_{k_2} \ldots t_{k_r} x^{n} \right) \\
&= x \left(\sum_{r=0}^{\infty} \left(t_0 + t_1 x + t_2 x^2 + \ldots \right)^r \right) \\
&= x \left(1 + t(x) + t(x)^2 + t(x)^3 + \ldots \right) \\
&= \frac{x}{1 - t(x)}.
\end{aligned}$$

Note that the exchange of summation is possible since the sum over n is actually finite for a fixed n.

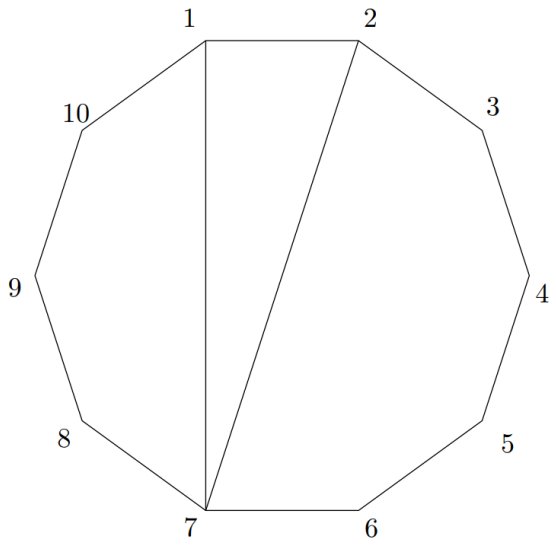
(b) Find t_n for $n = 1, 2, 3, 4, 5$. Do you recognize these numbers? Optional question: Why does this happen?

Solution: $t_n = b_{n-1}$, where b_n are the Catalan numbers.

E5. Show that the number of ways to triangulate a regular polygon with $n + 2$ labelled vertices is the Catalan numbers.

Solution: Let b_n be the number of ways to do this. Clearly $b_1 = 1$. Suppose the vertices are labelled $\{1, 2, \ldots, n+2\}$ where the ordering is along consecutive vertices. Then the side of the polygon that joins the vertices 1 and 2 is part of a triangle which must have a third vertex $k \in \{3, 4, 5, \ldots, n+2\}$. Whenever we k, the regular polygon is now been split into a polygon of sides $k-1$ and $n-k+4$ sides respectively. The number of ways to triangulate those would be b_{k-3} and b_{n-k+2} respectively as k varies from 3 to $n+2$. So we set $k = 3+r$, we get $b_n = \sum_{r=0}^{n-1} b_r b_{(n-1)-r}$.

The following is a picture for the case $n = 8, k = 7$.



Note that the above also gives a bijection between the triangulations of a polygon with $n+2$ and Part 5 of Question 2. The triangle containing the edge joining 1 and 2 can be thought of a as a root and the other $n-1$ triangles appearing in it can be thought of as the other vertices. The $n+1$ edges other than the 12 edge become the leaves. For example, here is a picture showing this bijection in one of the triangulations for the case $n = 8$.