UNIVERSITÀ DEGLI STUDI DI FIRENZE
Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea Magistrale in Informatica

Elaborato d'Esame

PROGETTAZIONE E ANALISI DI ALGORITMI

MASSIMO NOCENTINI

Donatella Merlini
Maria Cecilia Verri

Anno Accademico 2012-2013

Massimo Nocentini: *Progettazione e Analisi di Algoritmi*, Corso di Laurea Magistrale in Informatica, © Anno Accademico 2012-2013

CONTENTS

1 Lectures notes 5
- 1.1 Sorting algorithms 5
  - 1.1.1 MERGESORT algorithm 5
  - 1.1.2 QUICKSORT algorithm 5
  - 1.1.3 QUICKSORT: On the average number of checks 6
  - 1.1.4 QUICKSORT: On the average number of swaps 9
  - 1.1.5 Final consideration 11
- 1.2 Elements of combinatorics calculus 12
  - 1.2.1 Permutations P_n 12
  - 1.2.2 Dispositions $D_{n,k}$ 12
  - 1.2.3 Combinations $C_{n,k}$ 13
  - 1.2.4 Binomial coefficients 13
  - 1.2.5 Central Binomial Coefficients and Catalan Numbers 15
- 1.3 Generating functions basics 17
  - 1.3.1 Approaching 17
  - 1.3.2 Linearity of $\mathcal{G}$ 18
  - 1.3.3 Convolution of $\mathcal{G}$ 18
  - 1.3.4 Translation of $\mathcal{G}$ 19
  - 1.3.5 Identity respect to product 20
  - 1.3.6 Inverse of a formal power series 21
  - 1.3.7 Derivation for $\mathcal{G}$ 22
  - 1.3.8 Identity Principle 22
  - 1.3.9 Composition for $\mathcal{G}$ 23
  - 1.3.10 $\mathcal{G}(\frac{1}{n}a_n) = \int \frac{\mathcal{G}(a_n)-a_0}{t}dt$ 24
  - 1.3.11 $\mathcal{G}(\frac{1}{n+1}a_n) = \frac{1}{t}\int \mathcal{G}(a_n)dt$ 24
- 1.4 Examples of generating functions 25
  - 1.4.1 $\{c^n\}_{n\in\mathbb{N}}$ 25
  - 1.4.2 $\{(-1)^n a_n\}_{n\in\mathbb{N}}$ 25
  - 1.4.3 $\{n\}_{n\in\mathbb{N}}$ 25
  - 1.4.4 $\{n^2\}_{n\in\mathbb{N}}$ 26
  - 1.4.5 $\{a_n = 1 \text{ if } n \text{ is even },0 \text{ otherwise}\}_{n\in\mathbb{N}}$ 26
  - 1.4.6 $\{a_n = (1,0,0,1,0,0,1,0,0,1,\dots)\}_{n\in\mathbb{N}}$ 26
  - 1.4.7 $\{a_n = (1,0,0,2,0,0,4,0,0,8,\dots)\}_{n\in\mathbb{N}}$ 27
  - 1.4.8 $\{a_n = \frac{1}{n}\}_{n\in\mathbb{N}} \wedge a_0 = 0$ 27
  - 1.4.9 $\{H_n = \sum_{k=1}^{n}\frac{1}{k}\}_{n\in\mathbb{N}}$ 28
  - 1.4.10 $\{F_{n+2} = F_{n+1} + F_n\}_{n\in\mathbb{N}} \wedge F_1 = 1 \wedge F_0 = 0$ 29
  - 1.4.11 $\{T_{n+3} = T_{n+2} + T_{n+1} + T_n\}_{n\in\mathbb{N}} \wedge T_2 = 1 \wedge T_1 = 1 \wedge T_0 = 0$ 29

1.4.12 $\{a_n = \binom{p}{n}\}_{n\in\mathbb{N}}$ 30
1.4.13 $\{a_n = \binom{2n}{n}\}_{n\in\mathbb{N}}$ 32
1.4.14 $\{a_n = \frac{1}{n+1}\binom{2n}{n}\}_{n\in\mathbb{N}}$ 33
2 Sequential search simulation 35
2.1 Implementation 35
2.2 Results 36
3 Generating binary trees at random 41
3.1 Atkinson and Sack algorithm 41
3.1.1 Definitions 41
3.1.2 Splitting a *reducible* word 42
3.1.3 Generating a *balanced* word 43
3.1.4 Transform a *balanced* word in a *well-formed* word 44
3.1.5 Complexity and Space 44
3.2 Implementation using R 45
3.3 Checking randomness 46
3.4 Means of leaves and heights 47
3.5 Standardized means and asymptotic distributions 47
3.6 Drawing trees 49
4 Appendix 53
4.1 Project description 53
4.2 Original article about random binary trees generation 53

Text contents

Sources

1 LECTURES NOTES

1.1 SORTING ALGORITHMS

We study two algorithms that are based on checks between keys, those are MERGESORT and QUICKSORT.

1.1.1 *MERGESORT algorithm*

The MERGESORT algorithm is independent from the keys present in the input vector and its methodology behaves always the same. Let $n = 2^m$ be the length of the vector to be ordered, we can define a function C which count the number of checks needed to order the input vector. We define the function C using the method used by the algorithm at each step:

$$C(2^m) = 2C(2^{m-1}) + 2^m$$

Solving the recurrence[1] we obtain $C(n) \in O(n \log n)$. We can observe that if we use a method (like the ones we're studying in this section) based on checks between keys, isn't possible to do better than build a "checks tree", this allow use to have a lower bound for the complexity of those algorithms.[2]

1.1.2 *QUICKSORT algorithm*

The QUICKSORT algorithm depends on the distribution of the keys in the input vector. For what follow we assume to have a probability space $\Omega = D_n$, where D_n is the set of permutation of length n without repetition over $\{1, \ldots, n\}$. We focus on the simpler variant where the pivot is chosen as the right-most key[3]. We study the behavior of an application to the vector $(20, 25, 7, 3, 30, 8, 41, 18)$, reporting in Table 1 the steps performed. We can observe that in order to move the $pivot$ element in its final position, it is necessary for two keys (7, 25) to be checked twice. Hence, given a vector of length n, the number of checks performed before recurring on left and right partitions is $(n-1)+2$ (where $n-1$ is explained by the $pivot$ isn't indexed neither with i nor with j).

We can analyze the number of checks performed, partitioning it in the following cases:

1 put here the proof
2 in the slides of the first lecture maybe there's more material about this topic
3 report here the code

Table 1: Quicksort example

20	25	7	3	30	8	41	18		
$\uparrow$ i					$\uparrow$ j		$\uparrow$ pivot	$\rightarrow$	$\{20, 41, 8\}$
8	25	7	3	30	20	41	18		
	$\uparrow$ i		$\uparrow$ j				$\uparrow$ pivot	$\rightarrow$	$\{25, 30, 3\}$
8	3	7	25	30	20	41	18		
		$\uparrow$ j	$\uparrow$ i				$\uparrow$ pivot	$\rightarrow$	$\{7, 25, 7\}$
8	3	7	18	30	20	41	25		
			$\uparrow$ pivot					$\rightarrow$	recursion

WORST CASE when the vector is already ordered (in one of the two directions). In this case we got:

$$C(n) = (n-1) + 2 + C(n-1)$$

recurring only on one partition because the other have to be empty. We can expand the recurrence, fixing $C(0) = 0$:

$$\begin{aligned} C(n) &= (n+1) + C(n-1) = (n+1) + n + C(n-2) = \\ &= (n+1) + n + (n-1) + \ldots + 2 + C(0) = \\ &= \sum_{k=2}^{n+1} k + C(0) = \sum_{k=1}^{n+1} k - 1 + C(0) = \frac{(n+1)(n+2)}{2} - 1 \end{aligned}$$

so $C(n) \in O(n^2)$.

BEST CASE when the partition phase puts the $pivot$ in the middle, hence the QUICKSORT recurs on balanced partitions. In this case we have the same complexity of MERGESORT, hence $C(n) \in O(n \log n)$

We explain the average case in a dedicated section.

1.1.3 *QUICKSORT: On the average number of checks*

To study this case we have to consider all elements of Ω (recall that $w \in \Omega \rightarrow (w[i] \in \{1, \ldots, n\}) \wedge (\forall i \neq j : w[i] \neq w[j])$). First of all we can suppose that j is the $pivot$, hence a generic w will have this structure:

$$w = (C_{j-1} \quad C_{n-j} \quad j)$$

where C_k is a vector of length k. We can consider the probability to have j as $pivot$ considering the uniform distribution on Ω:

$$\mathbb{P}\left(w \in \Omega : w[n] = j\right) = \frac{(n-1)!}{n!} = \frac{1}{n}$$

Our goal here is to build a function $C(n)$ which counts the average number of checks during an execution of the algorithm given an input vector of length n. In order to do that observe that every keys $j \in \{1, \ldots, n\}$ can be the $pivot$, we can write:

$$C(n) = (n+1) + \frac{1}{n} \sum_{j=1}^{n} C(j-1) + C(n-j)$$

Observing the sum when j runs:

$$\begin{aligned} j = 1 &\rightarrow C(0) + C(n-1) \\ j = 2 &\rightarrow C(1) + C(n-2) \\ &\ldots \\ j = n-1 &\rightarrow C(n-2) + C(1) \\ j = n &\rightarrow C(n-1) + C(0) \end{aligned}$$

Hence we can rewrite:

$$C(n) = (n+1) + \frac{2}{n} \sum_{j=0}^{n-1} C(j)$$

Now we do some manipulation:

$$\begin{aligned} C(n) &= (n+1) + \frac{2}{n} \sum_{j=0}^{n-1} C(j) \\ nC(n) &= n(n+1) + 2 \sum_{j=0}^{n-1} C(j) \end{aligned}$$

Subtract the previous $(n-1)$ term to both members:

$$\begin{aligned}
nC(n)-(n-1)C(n-1) &= n(n+1)+2\sum_{j=0}^{n-1}C(j) \\
&\quad -\left((n-1)((n-1)+1)+2\sum_{j=0}^{(n-1)-1}C(j)\right) \\
&= n(n+1)+2\sum_{j=0}^{n-1}C(j) \\
&\quad -n(n-1)-2\sum_{j=0}^{n-2}C(j) \\
&= n(n+1-(n-1))+2C(n-1) \\
&= 2(n+C(n-1))
\end{aligned}$$

Getting $nC(n) = 2n+(n+1)C(n-1)$, divide both member by $n(n+1)$:

$$\frac{C(n)}{n+1} = \frac{2}{n+1} + \frac{C(n-1)}{n}$$

We arrived at a recurrence $A(n) = b(n) + A(n-1)$, where $A(n) = \frac{C(n)}{n+1}$ and $b(n) = \frac{2}{n+1}$. So we expand, fixing $C(0) = 0$:

$$\begin{aligned}
\frac{C(n)}{n+1} &= \frac{2}{n+1} + \frac{C(n-1)}{n} = \frac{2}{n+1} + \frac{2}{n} + \frac{C(n-2)}{n-1} \\
&= \frac{2}{n+1} + \frac{2}{n} + \ldots + \frac{2}{3} + \frac{2}{2} + \frac{C(0)}{1} \\
&= \frac{2}{n+1} + \frac{2}{n} + \ldots + \frac{2}{3} + 1 \\
&= 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots + \frac{1}{3}\right) + 1 \\
&= 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots + \frac{1}{3}\right) + 2\frac{1}{2} + 2 + 1 - 2\frac{1}{2} - 2 \\
&= 2\left(\frac{1}{n+1} + \frac{1}{n} + \ldots + \frac{1}{3} + \frac{1}{2} + 1\right) - 2 \\
&= 2(H_{n+1} - 1)
\end{aligned}$$

Having recognized the harmonic numbers $H_{n+1} = \left(\frac{1}{n+1} + \frac{1}{n} + \ldots + \frac{1}{3} + \frac{1}{2} + 1\right)$, the final result is

$$C(n) = 2(n+1)(H_{n+1} - 1)$$

In order to bound $C(n)$ we have to recall that $H_n \sim \ln(n) + \gamma$, hence $C(n) \in O(n\log n)$.

From a practical point of view, to avoid the worst case, when a sorting problem is approached with the QUICKSORT algorithm we can choose to do one of the following actions before starting the sorting process:

- shuffling the input vector and proceed with the algorithm described above;
- choose the pivot element at random, move it in the right-most position and proceed with the algorithm described above.

Each of the two tricks require linear time in the dimension of the input vector and allow us to use the result of the average case and working with $O(n \log n)$ number of checks.

1.1.4 *QUICKSORT: On the average number of swaps*

Our goal here is to build a function $S(n)$ which counts the average number of swaps during an execution of the algorithm given an input vector of length n.

The recurrence for the average number of swaps is as follow:

$$S(n) = \frac{n-2}{6} + \frac{1}{n}\sum_{j=1}^{n} S(j-1) + S(n-j)$$

We proceed the proof in two stages, first studying

$$\mathit{one} : \mathbb{E}\left[K_j\right] = \frac{(j-1)(n-j)}{n-1}$$

where K_j is a random variable that depends on j, after

$$\mathit{two} : \frac{n-2}{6} = \frac{1}{n}\sum_{j=1}^{n} \mathbb{E}\left[K_j\right]$$

In what follow, suppose to have a probability space Ω as defined in the previous sections and an uniform distribution above it.

Proof of one. Again, suppose the pivot is $j \in \{1, \dots, n\}$. We define the random variable $K_j : \Omega \to \mathbb{R}$ such that:

$$w[n] = j \to K_j(w) = s$$
$$w[n] \neq j \to K_j(w) = 0$$

where s is the number of swaps performed by the partitioning phase of QUICKSORT given an input vector w of length n (hence the variable K_j counts the number of swaps). Now we can compute the probability to have k swaps when j is the pivot:

$$\mathbb{P}\left(K_j = k\right) = \frac{\binom{n-j}{k}\binom{j-1}{k}(n-j)!(j-1)!}{(n-1)!}$$

We can justify the above formula in the following steps:

- we have k swaps when $|\{w_i : w_i < j\}| = k$ and $|\{w_i : w_i > j\}| = k$
- we can choose k keys from $\{w_i : w_i < j\}$ in $\binom{j-1}{k}$ ways and, for each choice, there are $(j-1)!$ ways to sort them;
- we can choose k keys from $\{w_i : w_i > j\}$ in $\binom{n-j}{k}$ ways and, for each choice, there are $(n-j)!$ ways to sort them;
- the total number of possible permutation of n keys excluding the $pivot$ (which is fixed in the right-most position) is $(n-1)!$

Now we study the mean of the variable K_j:

$$\mathbb{E}\left[K_j\right] = \sum_{k \geqslant 0} k\mathbb{P}\left(K_j = k\right)$$

Using $\binom{n}{m} = \frac{n!}{m!(n-m)!}$, we can rewrite:

$$\mathbb{P}\left(K_j = k\right) = \binom{n-j}{k}\binom{j-1}{k}\frac{(n-j)!(j-1)!}{(n-1)!} = \binom{n-j}{k}\binom{j-1}{k}\binom{n-1}{j-1}^{-1}$$

We put the previous rewrite of $\mathbb{P}\left(K_j = k\right)$ into the definition of $\mathbb{E}\left[K_j\right]$:

$$\mathbb{E}\left[K_j\right] = \sum_{k \geqslant 0} k\mathbb{P}\left(K_j = k\right) = \sum_{k \geqslant 0} k\frac{\binom{n-j}{k}\binom{j-1}{k}}{\binom{n-1}{j-1}}$$

Using the following rewrite for $\binom{j-1}{k}$:

$$\binom{j-1}{k} = \frac{(j-1)!}{k!(j-1-k)!} = \frac{(j-1)}{k}\frac{(j-2)!}{(k-1)!(j-1-k)!} = \frac{(j-1)}{k}\binom{j-2}{k-1}$$

And $\binom{n}{m} = \binom{n}{n-m}$ implies $\binom{j-2}{k-1} = \binom{j-2}{j-2-(k-1)} = \binom{j-2}{j-k-1}$, then:

$$\begin{aligned}\mathbb{E}\left[K_j\right] &= \sum_{k \geqslant 0} k\mathbb{P}\left(K_j = k\right) = \frac{1}{\binom{n-1}{j-1}}\sum_{k \geqslant 0} k\binom{n-j}{k}\binom{j-1}{k} \\ &= \frac{1}{\binom{n-1}{j-1}}\sum_{k \geqslant 0} k\binom{n-j}{k}\frac{j-1}{k}\binom{j-2}{j-k-1} = \frac{j-1}{\binom{n-1}{j-1}}\sum_{k \geqslant 0}\binom{n-j}{k}\binom{j-2}{j-k-1}\end{aligned}$$

Now we recognize the Vandermonde result:

$$\sum_{k \geqslant 0}\binom{r}{k}\binom{s}{n-k} = \binom{r+s}{n}$$

which can be proved directly because the ipergeometric distribution has exactly the same structure, and being a distribution, it sum up to 1. So we use this result applying it to $\sum_{k \geqslant 0}\binom{n-j}{k}\binom{j-2}{j-k-1} = \binom{n-2}{j-1}$, obtaining:

$$\begin{aligned}\mathbb{E}\left[K_j\right] &= \frac{j-1}{\binom{n-1}{j-1}}\binom{n-2}{j-1} = \frac{(j-1)(n-2)!(j-1)!(n-j)!}{(n-1)!(j-1)!(n-j-1)!} = \\ &= \frac{(j-1)(n-2)!(j-1)!(n-j)(n-j-1)!}{(n-1)(n-2)!(j-1)!(n-j-1)!} = \frac{(j-1)(n-j)}{n-1}\end{aligned}$$

□

Now we proceed with the second proof:

Proof of two. Let us start with:

$$\begin{aligned}
\frac{1}{n}\sum_{j=1}^{n}\mathbb{E}\left[K_j\right] &= \frac{1}{n}\sum_{j=1}^{n}\frac{(j-1)(n-j)}{n-1} = \frac{1}{n(n-1)}\sum_{j=1}^{n}(j-1)(n-j) = \\
&= \frac{1}{n(n-1)}\sum_{j=1}^{n}(j(1+n)-j^2-n) \\
&= \frac{1}{n(n-1)}\left((n+1)\sum_{j=1}^{n}j - \sum_{j=1}^{n}j^2 - n\sum_{j=1}^{n}1\right) \\
&= \frac{1}{n(n-1)}\left(\frac{n(n+1)^2}{2} - \frac{n(n+1)(2n+1)}{6} - n^2\right) = \ldots = \frac{n-2}{6}
\end{aligned}$$

□

We are now ready to solve the main recurrence using the same strategy for the average number of checks, fixing $S(0) = S(1) = S(2) = 0$:

$$\begin{aligned}
nS(n) - (n-1)S(n-1) &= \frac{2n-3}{6} + 2S(n-1) \\
\frac{S(n)}{n+1} &= \frac{S(n-1)}{n} + \frac{2n-3}{6n(n+1)} = \sum_{k=3}^{n}\frac{2k-3}{6k(k+1)}
\end{aligned}$$

Study the general term of the summation $\frac{2k-3}{6k(k+1)}$ and decompose it in partial fractions using Maxima:

```
(%i131) generalTerm:(2*k-3)/(6*k*(k-1));
        pf:partfrac(generalTerm,k);
        integrate(pf , k);
```

(%o131) $\frac{2\,k-3}{6\,(k-1)\,k}$

(%o132) $\frac{1}{2\,k} - \frac{1}{6\,(k-1)}$

(%o133) $\frac{\log(k)}{2} - \frac{\log(k-1)}{6}$

Hence $S(n) \in O(n \log n)$.

1.1.5 *Final consideration*

We've finished our treatment of sorting algorithms based on checks between keys. We've seen theoretical results under a probability space Ω composed by permutations of n objects. This allow us to remark that if we setup a simulation where we apply one algorithm seen above to the entire space Ω with vectors of a given dimension n, then the average number of checks and swaps must be equal to the theoretical results obtained in the previous formulas.

1.2 ELEMENTS OF COMBINATORICS CALCULUS

1.2.1 *Permutations* P_n

Given a vector $(a_1, \ldots, a_n)$ the number of permutations (or the possible ways to order the elements a_i) is $n!$. A simple proof of that is as follow: for the position 1 are available n elements, for the second position (after fixing the first) are available $n-1$ elements arriving to the position n where only one element is left, hence $n!$.

It is interesting to show how to add a new element a_{n+1} to existing permutations from the set $\{a_1, \ldots, a_n\}$, this is indeed another proof (by induction) to the number $n!$ justified above. Having the permutations of length n by hypothesis, we study how is it possible to add a_{n+1} into a given permutation $w = (a_1, a_2, \ldots, a_n)$:

$$
\begin{aligned}
&(a_{n+1}, a_1, a_2, \ldots, a_n) \\
&(a_1, a_{n+1}, a_2, \ldots, a_n) \\
&\ldots \\
&(a_1, a_2, \ldots, a_{n+1}, a_n) \\
&(a_1, a_2, \ldots, a_n, a_{n+1})
\end{aligned}
$$

Those new vectors are $n+1$ because in the original permutation $w = (a_1, a_2, \ldots, a_n)$ there are $n-1$ "internal holes" between the consecutive elements a_i and a_{i+1}, plus the left-most and right-most "external holes" $(\square, a_1, ...)$ and $(..., a_n, \square)$. So far we've considered only one permutation w of n elements, repeating the same reasoning for the others w_i (which are $n!-1$), it must exists $n+1$ new vectors for each of them. So the number of permutation of $n+1$ elements is $(n+1)n! = (n+1)!$ (what we would have expected).

1.2.2 *Dispositions* $D_{n,k}$

A special case of permutations are dispositions, that is the number of different ways in which k elements can be chosen from a set of length n, with $n \geqslant k$. We have n elements available for the first choice, $n-1$ for the second choice arriving to $n-k+1$ elements for the k-th choice, hence $D_{n,k} = n(n-1)\cdots(n-k+1)$. We can write an example to show how to build dispositions. Let $\Omega = \{a, b, c, d\}$ and we want to build $D_{4,2}$, that is the set of ordered pairs, paying attention that

$\forall i : (i, i) \notin D_{n,2}$. The following pairs are the candidates (the pairs of the form (i, i) aren't listed because after we choose i for the first position it isn't possible to choose it again for the second):

$$\begin{matrix} (a, b) & (a, c) & (a, d) \\ (b, a) & (b, c) & (b, d) \\ (c, a) & (c, b) & (c, d) \\ (d, a) & (d, b) & (d, c) \end{matrix}$$

We've build the previous matrix because we take care about the order in which the pairs are built. In the next section we'll see that there exists another set that doesn't take care of the order.

1.2.3 *Combinations* $C_{n,k}$

A special case of dispositions are combinations. With combinations, notation $C_{n,k}$, we mean a set of sets $\{\{a_{i_1}, \dots, a_{i_k}\}\}$ without regard to the order in which $a_{i_1}, \dots, a_{i_k}$ appear. We can see combinations like dispositions modulo the symmetric relation: $(j, i) \in C_{n,2} \to (i, j) \notin C_{n,2}$ and $(i, i) \notin C_{n,2}, \forall i)$. The following sub matrix is to combinations as the previous one is to dispositions:

$$\begin{matrix} (a, b) & (a, c) & (a, d) \\ & (b, c) & (b, d) \\ & & (c, d) \end{matrix}$$

The combinations matrix is obtained by the disposition matrix, breaking the symmetry, only half of them belongs to the dispositions $C_{4,2}$. Hence $C_{n,k} = \binom{n}{k} = \frac{D_{n,k}}{k!} = \frac{n(n-1)\cdots(n-k+1)}{k!} = \frac{n!}{k!(n-k)!)}$.

1.2.4 *Binomial coefficients*

In this section we study the binomial coefficients, and in the particular the writing $(a + b)^n, n \in \mathbb{N}$. It is known that:

$$(a + b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}$$

Now we study $(a + b)^r = \sum_{k=0}^{r} \binom{r}{k} a^k b^{r-k}, r \in \mathbb{R}$, in particular we have to pay attention to the binomial coefficient when $r \in \mathbb{R}$:

$$\binom{r}{k} = \frac{r(r-1)\cdots(r-k+1)}{k!}$$

The previous product make sense, instead we would have had problem with $\frac{r!}{k!(r-k)!}$. Now do simple manipulation:

$$(a+b)^r = \left(a\left(1+\frac{b}{a}\right)\right)^r = a^r\left(1+\frac{b}{a}\right)^r$$

In general we can call $z = \frac{b}{a}$ and reduce to study the function $f(z) = (1+z)^r$, which can be expanded with Taylor:

$$f(z) = \sum_{k \geqslant 0} \frac{f^{(k)}(0)}{k!} z^k$$

Study the derivatives of $f(z)$:

$$\begin{aligned}
&f^{(0)}(0) = 1 \\
&f^{(1)}(z) = r(1+z)^{r-1} \quad \rightarrow \quad f^{(1)}(0) = r \\
&f^{(2)}(z) = r(r-1)(1+z)^{r-2} \quad \rightarrow \quad f^{(2)}(0) = r(r-1) \\
&f^{(k)}(z) = r(r-1)\cdots(r-k+1)(1+z)^{r-k} \quad \rightarrow \quad f^{(k)}(0) = r(r-1)\cdots(r-k+1)
\end{aligned}$$

Now we can substitute the previous into $f(z)$:

$$f(z) = \sum_{k \geqslant 0} \frac{f^{(k)}(0)}{k!} z^k = \sum_{k \geqslant 0} \frac{r(r-1)\cdots(r-k+1)}{k!} z^k = \sum_{k \geqslant 0} \binom{r}{k} z^k$$

Exercise 1.2.1. *Compute* $\binom{\frac{1}{2}}{3}$.

Sol. It is sufficient to build a function $f(z) = (1+z)^r$ with the right value for r and then expand $f(z)$ with Taylor, the coefficient of z^3 is the desired value $\binom{r}{3}$.

In this case $r = \frac{1}{2}$, so $f(z) = \sqrt{1+z} = \binom{\frac{1}{2}}{0} z^0 + \binom{\frac{1}{2}}{1} z^1 + \binom{\frac{1}{2}}{2} z^2 + \binom{\frac{1}{2}}{3} z^3 + \ldots$

Using Maxima we expand with Taylor:

```
f(z):=sqrt(1+z);
taylor(f(z), z, 0, 6),numer;
```

Getting:

$$f(z) = 1 + z/2 - z^2/8 + z^3/16 - (5*z^4)/128 + (7*z^5)/256 - (21*z^6)/1024 + ...$$

Now we do a simple check:

```
mapBinomial(x):=binomial (1/2, x);
map (mapBinomial, [0, 1, 2, 3, 4, 5, 6]);
```

Getting:

$$[1, 1/2, -1/8, 1/16, -5/128, 7/256, -21/1024]$$

□

Now suppose to have $\binom{-n}{k}, n \in \mathbb{N}$:

$$\binom{-n}{k} = \frac{-n(-n-1)\cdots(-n-k+1)}{k!}$$

At the numerator there are k terms, so factor -1:

$$\binom{-n}{k} = \frac{(-1)^k(n+k-1)\cdots(n+1)n}{k!} = \frac{(-1)^k(n+k-1)\cdots(n+1)n(n-1)!}{k!(n-1)!}$$
$$= \frac{(-1)^k(n+k-1)!}{k!(n-1)!} = (-1)^k\binom{n+k-1}{k}$$

1.2.5 *Central Binomial Coefficients and Catalan Numbers*

We introduce in the following paragraph a very important class of numbers, the Catalan numbers. We proceed step by step, making an observation on a rewriting for $\binom{n}{k}$ where $n, k \in \mathbb{N}$.

Given a set of n elements $a_1, \ldots, a_n$ it is possible to generate the set S of the combinations of length k. Now choose an element, let say a_j, and partition S in two set:

$$S^+_{a_j} = \left\{(a_{\pi(1)}, \ldots, a_{\pi(k)}) | \exists i : a_j = a_{\pi(i)}\right\}$$
$$S^-_{a_j} = \left\{(a_{\pi(1)}, \ldots, a_{\pi(k)}) | \forall i : a_j \neq a_{\pi(i)}\right\}$$

where $\pi : \{1, \ldots, k\} \to \{1, \ldots, n\}$ is one-to-one. The following holds:

$$\binom{n}{k} = \left|S^+_{a_j}\right| + \left|S^-_{a_j}\right| = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Proof. We've chosen a_j to partition S, hence

- $\left|S^+_{a_j}\right| = \binom{n-1}{k-1}$ because $S^+_{a_j}$ contains sets which everyone contains a_j. But we can build $S^+_{a_j}$ from scratch choosing $k-1$ elements (a_j is contained in every set of $S^+_{a_j}$, hence we have to build combinations of length $k-1$) from $\{a_1, \ldots, a_n\} \setminus \{a_j\}$ (a_j is an implicit choice)

- $\left|S^-_{a_j}\right| = \binom{n-1}{k}$ because $S^-_{a_j}$ contains sets which everyone doesn't contains a_j. But we can build $S^-_{a_j}$ from scratch choosing k elements (a_j isn't contained in any set of $S^-_{a_j}$, hence we have to build combinations of length k) from $\{a_1, \ldots, a_n\} \setminus \{a_j\}$ (a_j is an impossible choice)

□

The recurrence $\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$ allow us to build an infinite matrix reported in Table 2 with interesting numbers written inside a ticket as $\boxed{j}$. The numbers $\binom{2h}{h}$ are called *Central Binomial Coefficients*, while $\frac{1}{h+1}\binom{2h}{h}$ are called *Catalan Numbers*.

We finish this section on combinatorics with some exercises.

Table 2: Central Binomial Coefficients

	0	1	2	3	4	5	6	...	k
0	$\boxed{1}$								
1	1	1							
2	1	$\boxed{2}$	1						
3	1	3	3	1					
4	1	4	$\boxed{6}$	4	1				
5	1	5	10	10	5	1			
6	1	6	15	$\boxed{20}$	15	6	1		
$\vdots$								$\ddots$	
$\vdots$								$\binom{n-1}{k-1}$	$\binom{n-1}{k}$
n									$\binom{n}{k}$

Exercise 1.2.2. *Compute* $\binom{-\frac{1}{2}}{k}$.

Sol. Using the approach followed in Exercise 1.2.1, we can build the function $f(z) = \frac{1}{\sqrt{1+z}}$ and expand it with Taylor. Here we use another approach that follow the definition:

$$\begin{aligned}
\binom{-\frac{1}{2}}{k} &= \frac{-\frac{1}{2}(-\frac{1}{2}-1)(-\frac{1}{2}-2)\cdots(-\frac{1}{2}-k+1)}{k!} = \\
&= \frac{-\frac{1}{2}(-\frac{3}{2})(-\frac{5}{2})\cdots(-\frac{2k-1}{2})}{k!} = \frac{(-1)^k 135\cdots(2k-1)}{2^k k!} = \\
&= \frac{(-1)^k 135\cdots(2k-1) \quad 246\cdots 2k}{2^k k! \quad 246\cdots 2k} = \frac{(-1)^k (2k)!}{2^k k! \quad 2^k(123\cdots k)} = \\
&= \frac{(-1)^k(2k)!}{2^k k! \quad 2^k k!} = \frac{(-1)^k}{4^k}\binom{2k}{k}
\end{aligned}$$

Doing a little check with Maxima:

```
(%i117) f(z):=1/sqrt(1+z);
        taylor(f(z), z, 0, 6);
        mapBinomial(x):=binomial (-1/2, x);
        map (mapBinomial, [0, 1, 2, 3, 4, 5, 6]);
        byDef(k):=(((-1)^k)/(4^k))*binomial(2*k,k);
        map (byDef, [0, 1, 2, 3, 4, 5, 6]);
```

(%o117) $f(z) := \frac{1}{\sqrt{1+z}}$

(%o118) $1-\frac{z}{2}+\frac{3\,z^2}{8}-\frac{5\,z^3}{16}+\frac{35\,z^4}{128}-\frac{63\,z^5}{256}+\frac{231\,z^6}{1024}+\ldots$

(%o119) $\mathrm{mapBinomial}\,(x) := \binom{-\frac{1}{2}}{x}$

(%o120) $[1,-\frac{1}{2},\frac{3}{8},-\frac{5}{16},\frac{35}{128},-\frac{63}{256},\frac{231}{1024}]$

(%o121) $\mathrm{byDef}\,(k) := \frac{(-1)^k}{4^k}\binom{2\,k}{k}$

(%o122) $[1,-\frac{1}{2},\frac{3}{8},-\frac{5}{16},\frac{35}{128},-\frac{63}{256},\frac{231}{1024}]$

□

1.3 GENERATING FUNCTIONS BASICS

In this section we approach the basics for the generating functions topic.

1.3.1 *Approaching*

The main goal of this section is to manipulate a sequence $\{a_n\}_{n\in\mathbb{N}}$ with a function $a(t)$ such that:

$$a(t) = \sum_{n\geqslant 0} a_n t^n$$

Using $a(t)$ allow us to let t be unbounded and we wont assign any value to it (it is just a "placeholder").

We introduce an operator $\mathcal{G}$ which consume a sequence $\{a_n\}_{n\in\mathbb{N}}$ and produce a function $a(t)$ as defined above. In the following examples we experiment a little with Maxima, which implement $\mathcal{G}$ with the function `ggf(s)`, for some sequence s.

Let $\{a_n = n\}$:

```
(%i109) sequence:makelist(n,n,0, 6);
        taylor(ggf(sequence), x, 0, 8);
```

(%o109) $[0,1,2,3,4,5,6]$
(%o110) $x+2\,x^2+3\,x^3+4\,x^4+5\,x^5+6\,x^6+\ldots$
So $\mathcal{G}(a_n) = a(x) = x+2\,x^2+3\,x^3+4\,x^4+5\,x^5+6\,x^6+\ldots$

Let $\{a_n = 1$ if n is even $, 0$ otherwise$\}$:

```
(%i105) onlyEvenTerms(n):=if is(mod (n,2)=0) then 1 else 0$
        sequence:makelist(onlyEvenTerms(n),n,0, 6);
        ggfOnlyEvenTerms:ggf(sequence);
        taylor(ggfOnlyEvenTerms, x, 0, 12);
```

(%o106) $[1, 0, 1, 0, 1, 0, 1]$
(%o107) $-\frac{1}{x^2-1}$
(%o108) $1 + x^2 + x^4 + x^6 + x^8 + x^{10} + x^{12} + ...$
So $\mathcal{G}(a_n) = a(x) = 1 + x^2 + x^4 + x^6 + x^8 + x^{10} + x^{12} + ...$

Let $\{a_n = 1$ if $n \leqslant 2, 0$ otherwise$\}$:

```
(%i97) onlyFirstThreeTerms(n):=if is(n <= 2) then 1 else 0$
       sequence:makelist(onlyFirstThreeTerms(n),n,0, 6);
       ggfOnlyFirstThreeTerms:ggf(sequence);
       taylor(ggfOnlyFirstThreeTerms, x, 0, 100);
```

(%o98) $[1, 1, 1, 0, 0, 0, 0]$
(%o99) $x^2 + x + 1$
(%o100) $1 + x + x^2 + ...$
So $\mathcal{G}(a_n) = a(x) = 1 + x + x^2$.

1.3.2 *Linearity of* $\mathcal{G}$

Let $\{a_n\}_{n\in\mathbb{N}}$ and $\{b_n\}_{n\in\mathbb{N}}$ be two sequences, $a(t)$ and $b(t)$ their formal power series respectively. Lets sum both of them:

$$\begin{aligned}
\mathcal{G}(a_n) + \mathcal{G}(b_n) &= a_0 + a_1 t + a_2 t^2 + \ldots + a_n t^n + \ldots \\
&+ b_0 + b_1 t + b_2 t^2 + \ldots + b_n t^n + \ldots = \\
&= (a_0 + b_0) + (a_1 + b_1)t + (a_2 + b_2)t^2 + \ldots + (a_n + b_n)t^n + \ldots = \\
&= \mathcal{G}(a_n + b_n)
\end{aligned}$$

It is possible to generalize, obtaining:

$$\begin{aligned}
\alpha\mathcal{G}(a_n) &= \mathcal{G}(\alpha a_n) \\
\alpha\mathcal{G}(a_n) + \beta\mathcal{G}(b_n) &= \mathcal{G}(\alpha a_n + \beta b_n)
\end{aligned}$$

So $\mathcal{G}$ is a *linear operator*.

1.3.3 *Convolution of* $\mathcal{G}$

Let $\{a_n\}_{n\in\mathbb{N}}$ and $\{b_n\}_{n\in\mathbb{N}}$ be two sequences, $a(t)$ and $b(t)$ their formal power series respectively. We study the product $a(t)b(t)$ with Maxima:

```
(%i190) makeSequence(t,j):=sum (j[i]*t^i, i, 0, inf);
        as:taylor(makeSequence(t,a), t, 0, 3);
        bs:taylor(makeSequence(t,b), t, 0, 3);
        as*bs;
        sum ((sum(a[k]*b[n-k],k,0,n))*t^n, n, 0, inf);
        taylor(sum((sum(a[k]*b[n-k],k,0,n))*t^n,n,0,inf), t, 0, 3);
```

(%o190) $\mathrm{makeSequence}\,(t,j) := \sum_{i=0}^{\infty} j_i\, t^i$

(%o191) $a_0 + a_1\, t + a_2\, t^2 + a_3\, t^3 + \ldots$

(%o192) $b_0 + b_1\, t + b_2\, t^2 + b_3\, t^3 + \ldots$

(%o193) $a_0\, b_0 + (a_1\, b_0 + a_0\, b_1)\, t + (a_2\, b_0 + a_1\, b_1 + a_0\, b_2)\, t^2 + (a_3\, b_0 + a_2\, b_1 + a_1\, b_2 + a_0\, b_3)\, t^3 + \ldots$

(%o194) $\sum_{n=0}^{\infty} \left(\sum_{k=0}^{n} a_k\, b_{n-k} \right) t^n$

(%o195) $\left(\sum_{k=0}^{0} b_{-k}\, a_k \right) + \left(\sum_{k=0}^{1} b_{1-k}\, a_k \right) t + \left(\sum_{k=0}^{2} b_{2-k}\, a_k \right) t^2 + \left(\sum_{k=0}^{3} b_{3-k}\, a_k \right) t^3 + \ldots$

So $\mathcal{G}(a_n)\mathcal{G}(b_n) = \mathcal{G}\left(\sum_{k=0}^{n} a_k\, b_{n-k}\right)$.

1.3.4 *Translation of $\mathcal{G}$*

Let $\{a_n\}_{n\in\mathbb{N}}$ and $\{a_{n+1}\}_{n\in\mathbb{N}}$ be two sequences, $a(t)$ and $b(t)$ their formal power series respectively. We study how we can obtain $\mathcal{G}(a_{n+1}$ from $\mathcal{G}(a_n)$ with Maxima:

```
(%i78)  makeSequence(t,j,offset):=if is(offset >= 0) then sum
        (j[i+offset]*t^i, i, 0, inf) else sum (j[i]*t^(i-offset), i, 0, inf)$
        as:taylor(makeSequence(t,a,0), t, 0, 4);
        bs:taylor(makeSequence(t,a,1), t, 0, 3);
        (as-a[0])/t;
```

(%o79) $a_0 + a_1\, t + a_2\, t^2 + a_3\, t^3 + a_4\, t^4 + \ldots$

(%o80) $a_1 + a_2\, t + a_3\, t^2 + a_4\, t^3 + \ldots$

(%o81) $a_1 + a_2\, t + a_3\, t^2 + a_4\, t^3 + \ldots$

So $\mathcal{G}(a_{n+1}) = \frac{\mathcal{G}(a_n)-a_0}{t}$. We can generalize:

```
(%i249) load("simplify_sum");
        sequence:makeSequence(t,a,0);
        translated:makeSequence(t,a,s)$
        (sequence - sum(a[i]*t^i,i,0,s-1))/t^s;
        (taylor(sequence, t, 0, 55) - sum(a[i]*t^i,i,0,49))/t^50;
```

(%o250) $\sum_{i=0}^{\infty} a_i t^i$

(%o252) $\frac{\left(\sum_{i=0}^{\infty} a_i t^i\right) - \sum_{i=0}^{s-1} a_i t^i}{t^s}$

(%o253) $a_{50} + a_{51}\, t + a_{52}\, t^2 + a_{53}\, t^3 + a_{54}\, t^4 + a_{55}\, t^5 + ...$

where in the last line we've fixed $s = 50$. In general holds $\mathcal{G}(a_{n+s}) = \frac{\mathcal{G}(a_n) - \sum_{i=0}^{s-1} a_i t^i}{t^s}$ for $s \geqslant 0$. If we have $\mathcal{G}(a_{n-s})$, where $n \geqslant s$, then $\mathcal{G}(a_{n-s}) = a_0 t^s + a_1 t^{s+1} + a_2 t^{s+2} + \ldots = t^s \mathcal{G}(a_n)$. With Maxima:

```
(%i129) p:5$
        cs:taylor(makeSequence(t,a,-p), t, 0, 10);
        ds:taylor((t^p)*makeSequence(t,a,0), t, 0, 10);
```

(%o130) $a_0\, t^5 + a_1\, t^6 + a_2\, t^7 + a_3\, t^8 + a_4\, t^9 + a_5\, t^{10} + ...$

(%o131) $a_0\, t^5 + a_1\, t^6 + a_2\, t^7 + a_3\, t^8 + a_4\, t^9 + a_5\, t^{10} + ...$

1.3.5 *Identity respect to product*

There exists a function $c(t)$ such that $a(t)c(t) = a(t)$, where $a(t) = \mathcal{G}(a_n)$ and where $c(t) = \mathcal{G}(c_n)$? To answer we apply the convolution of $\mathcal{G}$ to the product:

$$\mathcal{G}(a_n)\mathcal{G}(c_n) = \mathcal{G}\left(\sum_{k=0}^{n} a_k\, c_{n-k}\right)$$

We impose the equality as desired $\mathcal{G}(a_n) = \mathcal{G}\left(\sum_{k=0}^{n} a_k\, c_{n-k}\right)$ which holds if the sequences to which $\mathcal{G}$ is applied to are the same. Hence:

$$a_n = \sum_{k=0}^{n} a_k\, c_{n-k} = a_0 c_n + a_1 c_{n-1} + \ldots + a_{n-1} c_1 + a_n c_0$$

Which holds if and only if $c_0 = 1 \wedge \forall i \in \{1, \ldots, n\} : c_i = 0$. The sequence $\{c_n\}_{n \in \mathbb{N}}$ is something like $(1, 0, 0, 0, 0, \ldots)$, its formal power series $c(t) = \mathcal{G}(c_n) = 1 + 0t + 0t^2 + 0t^3 + \ldots$, hence we rename $c(t) = 1(t)$.

1.3.6 *Inverse of a formal power series*

There exists a function $b(t)$ such that $a(t)b(t) = 1(t)$, where $a(t) = \mathcal{G}(a_n)$ and where $b(t) = \mathcal{G}(b_n)$? Lets study the product $a(t)b(t)$:

$$\begin{aligned} a(t)b(t) &= a_0\, b_0 \quad + (a_1\, b_0 + a_0\, b_1)\, t + (a_2\, b_0 + a_1\, b_1 + a_0\, b_2)\, t^2 \quad + \dots \\ &= 1 \qquad\qquad\qquad +0+0 \qquad\qquad\qquad + \dots \end{aligned}$$

That product have to be equal to $1(t)$, this allow to build a system, which we solve it with Maxima:

```
(%i109) makeSequence(t,j):=sum (j[i]*t^i, i, 0, inf)$
        makeSum(n):=sum (a[i]*b[n-i], i, 0, n)$
        as:taylor(makeSequence(t,a), t, 0, 4);
        bs:taylor(makeSequence(t,b), t, 0, 4);
        prod:as*bs;
        solve([makeSum(0)=1, makeSum(1)=0, makeSum(2)=0,
        makeSum(3)=0, makeSum(4)=0],[b[0],b[1],b[2],b[3],b[4]]);
```

(%o110) $a_0 + a_1\, t + a_2\, t^2 + a_3\, t^3 + a_4\, t^4 + ...$
(%o111) $b_0 + b_1\, t + b_2\, t^2 + b_3\, t^3 + b_4\, t^4 + ...$
(%o112) $a_0\, b_0 + (a_1\, b_0 + a_0\, b_1)\, t + (a_2\, b_0 + a_1\, b_1 + a_0\, b_2)\, t^2 +$
$(a_3\, b_0 + a_2\, b_1 + a_1\, b_2 + a_0\, b_3)\, t^3 + (a_4\, b_0 + a_3\, b_1 + a_2\, b_2 + a_1\, b_3 + a_0\, b_4)\, t^4 +$
...
(%o113) $[[b_0 = \frac{1}{a_0}, b_1 = -\frac{a_1}{a_0^2}, b_2 = -\frac{a_0\, a_2 - a_1^2}{a_0^3}, b_3 = -\frac{a_0^2\, a_3 - 2\, a_0\, a_1\, a_2 + a_1^3}{a_0^4},$
$b_4 = -\frac{a_0^3\, a_4 + a_0^2\, \left(-2\, a_1\, a_3 - a_2^2\right) + 3\, a_0\, a_1^2\, a_2 - a_1^4}{a_0^5}]]$

So the inverse of $a(t)$ is the function $a^{-1}(t) = b(t)$ such that $b(t) = \mathcal{G}(b_0, b_1, b_2, b_3, b_4, \dots)$, where every b_i is one of the solution reported in the Maxima output above. It is interesting to observe that it is possible to find $a^{-1}(t)$ if the first term of $\{a_n\}_{n\in\mathbb{N}}$ is such that $a_0 \neq 0$.

Exercise 1.3.1. *Let* $a(t) = 1 - t$ *be a formal power series such that* $a(t) = \mathcal{G}(a_n)$. *Find* $a^{-1}(t)$ *and the sequence* $\{b_n\}_{n\in\mathbb{N}}$ *such that* $a^{-1}(t) = \mathcal{G}(b_n)$.

Sol. We can find it immediately setting $a(t)b(t) = 1(t)$ and resolving respect to $b(t)$ which yields $b(t) = \frac{1}{1-t}$.

To find the sequence $\{b_n\}_{n\in\mathbb{N}}$ such that $a^{-1}(t) = \mathcal{G}(b_n)$, we use Maxima (where `sol` is the vector obtained in the previous Maxima output):

```
(%i90)  as:[1,-1,0,0,0];
        ev(sol[1],a[0]=as[1],a[1]=as[2],a[2]=as[3],a[3]=as[4],a[4]=as[5]);
        b(t):=1/(1-t);
        taylor(b(t),t,0,8);
```

(%o90) $[1,-1,0,0,0]$
(%o91) $[b_0=1,b_1=1,b_2=1,b_3=1,b_4=1]$
(%o92) $b(t):=\dfrac{1}{1-t}$
(%o93) $1+t+t^2+t^3+t^4+t^5+t^6+t^7+t^8+\ldots$
So we get the sequence $\{b_n\}_{n\in\mathbb{N}}=\{1_n\}_{n\in\mathbb{N}}=(1,1,1,1,1,1,\ldots)$ "inverse" of $\{a_n\}_{n\in\mathbb{N}}=(1,-1,0,0,0,0,\ldots)$. □

1.3.7 *Derivation for* $\mathcal{G}$

Let $\{a_n\}_{n\in\mathbb{N}}$ and $\{na_n\}_{n\in\mathbb{N}}$ be two sequences. We study how we can obtain $\mathcal{G}(na_n$ from $\mathcal{G}(a_n)$ with Maxima:

```
(%i207) makeSum(n):=sum (a[i]*t^i, i, 0, n)$
        makeDerSum(n):=sum (i*a[i]*t^i, i, 0, n)$
        an(n):=taylor(makeSum(n),t,0,6)$
        deran(n):=taylor(makeDerSum(n),t,0,6)$
        an(6);
        deran(6);
        t*diff(an, t);
```

(%o211) $a_0+a_1\,t+a_2\,t^2+a_3\,t^3+a_4\,t^4+a_5\,t^5+a_6\,t^6+\ldots$
(%o212) $a_1\,t+2\,a_2\,t^2+3\,a_3\,t^3+4\,a_4\,t^4+5\,a_5\,t^5+6\,a_6\,t^6+\ldots$
(%o213) $a_1\,t+2\,a_2\,t^2+3\,a_3\,t^3+4\,a_4\,t^4+5\,a_5\,t^5+6\,a_6\,t^6+\ldots$
So $\mathcal{G}(na_n)=t\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right)$.

1.3.8 *Identity Principle*

Let $\{a_n\}_{n\in\mathbb{N}}$ and $\{b_n\}_{n\in\mathbb{N}}$ be two sequences, the *Identity Principle* states:

$$\forall n\in\mathbb{N}:(a_n=b_n\leftrightarrow\mathcal{G}(a_n)=\mathcal{G}(b_n))$$

Exercise 1.3.2. *Suppose we forget the result shown in the previous section about* $\mathcal{G}(1_n)=\frac{1}{1-t}$*, how can be used the* Identity Principle *to prove that result?*

Sol. In order to use the principle we've to find two sequences which are equal for all $n\in\mathbb{N}$. Since $\{a_n\}_{n\in\mathbb{N}}=(1,1,1,1,\ldots)$ we can build another sequence $\{a_{n+1}\}_{n\in\mathbb{N}}$ using $\{a_n\}_{n\in\mathbb{N}}$. But the indices n for the former and $n+1$ for the latter doesn't change the series, also $\{a_{n+1}\}_{n\in\mathbb{N}}=(1,1,1,1,\ldots)$, hence

$\forall n \in \mathbb{N} : a_n = a_{n+1}$. By the principle $\mathcal{G}(a_n) = \mathcal{G}(a_{n+1})$ holds. We apply the derivation for $\mathcal{G}$:

$$\mathcal{G}(a_n) = \mathcal{G}(a_{n+1}) = \frac{\mathcal{G}(a_n) - a_0}{t} \qquad \text{where } a_0 = 1 \text{ by def of } \{a_n\}_{n\in\mathbb{N}}$$
$$\mathcal{G}(a_n) = \frac{\mathcal{G}(a_n) - 1}{t}$$
$$\mathcal{G}(a_n) = \mathcal{G}(1_n) = \frac{1}{1-t}$$

What we now remember! □

The exercise above let us to generalize a little. Let $\{a_n\}_{n\in\mathbb{N}}$ such that $a_n = c$. When the sequence is known a priory as in this case, it is useful to study the ratio between consecutive terms:

$$\frac{a_{n+1}}{a_n} = \frac{c}{c} \quad \to \quad a_{n+1} = a_n \quad \forall n \in \mathbb{N}$$

By *Identity Principle* follows:

$$\mathcal{G}(a_n) = \mathcal{G}(a_{n+1}) = \frac{\mathcal{G}(a_n) - a_0}{t} \qquad \text{where } a_0 = c \text{ by def of } \{a_n\}_{n\in\mathbb{N}}$$
$$\mathcal{G}(a_n) = \frac{\mathcal{G}(a_n) - c}{t}$$
$$\mathcal{G}(a_n) = \mathcal{G}(1_n) = \frac{c}{1-t}$$

1.3.9 *Composition for $\mathcal{G}$*

Let $\{a_n\}_{n\in\mathbb{N}}$ and $\{b_n\}_{n\in\mathbb{N}}$ be two sequences, we can compose their formal power series in the following way:

$$a(t) = \sum_{k\geqslant 0} a_k t^k \quad b(t) = \mathcal{G}(b_n)$$

$$\mathcal{G}(a_n) \circ \mathcal{G}(b_n) = a(b(t)) = \sum_{k=0}^{\infty} a_k b(t)^k = \sum_{k=0}^{\infty} a_k \mathcal{G}(b_n)^k$$

In the following section we'll do some example using the properties of $\mathcal{G}$.

1.3.10 $\mathcal{G}(\frac{1}{n}a_n) = \int \frac{\mathcal{G}(a_n)-a_0}{t}dt$

Let $b_n = \frac{1}{n}a_n$ be a new generic term of a sequence $\{b_n\}_{n\in\mathbb{N}}$. Follow that $a_n = nb_n$, but it isn't sufficient to apply the *Identity Principle* because in $n = 0$ the term b_0 isn't defined. So we've to augment $a_n = nb_n + a_0\delta_{n,0}$:

$$\begin{aligned}
a_n &= nb_n + a_0\delta_{n,0} \\
&\downarrow \\
\mathcal{G}(a_n) &= \mathcal{G}(nb_n) + \mathcal{G}(a_0\delta_{n,0}) \\
\mathcal{G}(a_n) &= t\frac{\partial}{\partial t}\left(\mathcal{G}(b_n)\right) + a_0\mathcal{G}(\delta_{n,0}) \\
\mathcal{G}(a_n) &= t\frac{\partial}{\partial t}\left(\mathcal{G}(\frac{1}{n}a_n)\right) + a_0 \\
\frac{\partial}{\partial t}\left(\mathcal{G}(\frac{1}{n}a_n)\right) &= \frac{\mathcal{G}(a_n)-a_0}{t} \\
\mathcal{G}(\frac{1}{n}a_n) &= \int \frac{\mathcal{G}(a_n)-a_0}{t}dt
\end{aligned}$$

1.3.11 $\mathcal{G}(\frac{1}{n+1}a_n) = \frac{1}{t}\int \mathcal{G}(a_n)dt$

Let $g_n = a_{n-1}$, where $g_0 = 0$ (otherwise a_{-1} hasn't meaning), to have:

$$\frac{1}{n+1}a_n = \frac{1}{n+1}g_{n+1}$$

We apply the *Identity Principle* directly because we've just renamed the general term of the sequence:

$$\begin{aligned}
\frac{1}{n+1}a_n &= \frac{1}{n+1}g_{n+1} \\
&\downarrow \\
\mathcal{G}(\frac{1}{n+1}a_n) &= \mathcal{G}(\frac{1}{n+1}g_{n+1}) \\
\mathcal{G}(\frac{1}{n+1}a_n) &= \frac{\mathcal{G}(\frac{1}{n}g_n)-g_0}{t} \quad \text{by translation of } \mathcal{G} \\
\mathcal{G}(\frac{1}{n+1}a_n) &= \frac{1}{t}\mathcal{G}(\frac{1}{n}g_n) \quad g_0 = 0 \\
\mathcal{G}(\frac{1}{n+1}a_n) &= \frac{1}{t}\int \frac{\mathcal{G}(g_n)-g_0}{t}dt \quad \text{by previous result} \\
\mathcal{G}(\frac{1}{n+1}a_n) &= \frac{1}{t}\int \frac{\mathcal{G}(a_{n-1})}{t}dt \\
\mathcal{G}(\frac{1}{n+1}a_n) &= \frac{1}{t}\int \mathcal{G}(a_n)dt \quad \text{by } \mathcal{G}(a_{n-1}) = t\mathcal{G}(a_n)
\end{aligned}$$

1.4 EXAMPLES OF GENERATING FUNCTIONS

In the following sections we study some interesting sequences. Every section's title shows the sequence under study in that section and its body contains the building of the relative generating function.

1.4.1 $\{c^n\}_{n\in\mathbb{N}}$

$$\begin{aligned}\mathcal{G}(c^n) = \sum_{n\geqslant 0} c^n t^n = \sum_{n\geqslant 0} 1\cdot(ct)^n = \sum_{n\geqslant 0} a_n\cdot(ct)^n \quad &\text{where } a_n = 1, \forall n \in \mathbb{N} \\ = \mathcal{G}(1_n)\circ(ct) = \frac{1}{1-ct} \quad &\text{by composition of } \mathcal{G}\end{aligned}$$

Doing a simple check with Maxima:

```
(%i226) b(t):=1/(1-c*t)$
        taylor(b(t),t,0,8);
```

(%o227) $1 + c\,t + c^2\,t^2 + c^3\,t^3 + c^4\,t^4 + c^5\,t^5 + c^6\,t^6 + c^7\,t^7 + c^8\,t^8 + ...$

1.4.2 $\{(-1)^n a_n\}_{n\in\mathbb{N}}$

$$\mathcal{G}((-1)^n a_n) = \sum_{n\geqslant 0} (-1)^n a_n t^n = \sum_{n\geqslant 0} a_n(-t)^n = a(-t)$$

1.4.3 $\{n\}_{n\in\mathbb{N}}$

$$\mathcal{G}(n) = \mathcal{G}(n\cdot 1_n) = t\frac{\partial}{\partial t}\left(\mathcal{G}(1_n)\right) = \frac{t}{(1-t)^2}$$

Doing a little check with Maxima:

```
(%i244) b(t):=1/(1-t)$
        fg(t):=t*diff(b(t),t);
        fg(t);
        taylor(fg(t),t,0,8);
```

(%o245) $fg\,(t) := t\,diff\,(b\,(t)\,,t)$

(%o246) $\dfrac{t}{(1-t)^2}$

(%o247) $t + 2\,t^2 + 3\,t^3 + 4\,t^4 + 5\,t^5 + 6\,t^6 + 7\,t^7 + 8\,t^8 + ...$

1.4.4 $\{n^2\}_{n\in\mathbb{N}}$

Using the result of the previous exercise:

$$\mathcal{G}(n^2) = \mathcal{G}(nn) = t\frac{\partial}{\partial t}(\mathcal{G}(n)) = -\frac{t^2+t}{t^3-3t^2+3t-1}$$

Just a little check:

```
(%i252) snder(t):=t*diff(fg(t),t)$
        ratsimp(snder(t));
        taylor(snder(t),t,0,8);
```

$$(\%o253)\ -\frac{t^2+t}{t^3-3t^2+3t-1}$$
$$(\%o254)\ t+4t^2+9t^3+16t^4+25t^5+36t^6+49t^7+64t^8+\ldots$$

1.4.5 $\{a_n = 1 \text{ if } n \text{ is even }, 0 \text{ otherwise}\}_{n\in\mathbb{N}}$

We compare $\mathcal{G}(a_n)$ with $\mathcal{G}(1_n)$:

$$\mathcal{G}(a_n) = 1+t^2+t^4+t^6+\ldots$$
$$\mathcal{G}(1_n) = 1+t^1+t^2+t^3+\ldots$$

So we can compose $\mathcal{G}(1_n)$ which is known with the function t^2 to obtaining $\mathcal{G}(a_n)$:

$$\mathcal{G}(a_n) = \frac{1}{1-t}\circ t^2 = \frac{1}{1-t^2}$$

Simple check:

```
(%i8)   b(t):=1/(1-t)$
        taylor(b(t^2),t,0,10);
```

$$(\%o9)\quad 1+t^2+t^4+t^6+t^8+t^{10}+\ldots$$

1.4.6 $\{a_n = (1,0,0,1,0,0,1,0,0,1,\ldots)\}_{n\in\mathbb{N}}$

We compare $\mathcal{G}(a_n)$ with $\mathcal{G}(1_n)$ again:

$$\mathcal{G}(a_n) = 1+t^3+t^6+t^9\ldots$$
$$\mathcal{G}(1_n) = 1+t^1+t^2+t^3+\ldots$$

So we can compose $\mathcal{G}(1_n)$ which is known with the function t^3 to obtaining $\mathcal{G}(a_n)$:

$$\mathcal{G}(a_n) = \frac{1}{1-t}\circ t^3 = \frac{1}{1-t^3}$$

Simple check:

```
(%i10) taylor(b(t^3),t,0,10);
```

(%o10) $1 + t^3 + t^6 + t^9 + \ldots$

1.4.7 $\{a_n = (1,0,0,2,0,0,4,0,0,8,\ldots)\}_{n\in\mathbb{N}}$

We compare $\mathcal{G}(a_n)$ with $\mathcal{G}(2^n)$ again:

$$\mathcal{G}(a_n) = 1 + 2t^3 + 4t^6 + 8t^9 \ldots$$
$$\mathcal{G}(2^n) = 1 + 2t^1 + 4t^2 + 8t^3 + \ldots$$

So we can compose $\mathcal{G}(2^n)$ which is known with the function t^3 to obtaining $\mathcal{G}(a_n)$:

$$\mathcal{G}(a_n) = \frac{1}{1-2t} \circ t^3 = \frac{1}{1-2t^3}$$

Simple check:

```
(%i13) b(t):=1/(1-2*t)$
       taylor(b(t^3),t,0,15);
```

(%o14) $1 + 2\,t^3 + 4\,t^6 + 8\,t^9 + 16\,t^{12} + 32\,t^{15} + \ldots$

1.4.8 $\{a_n = \frac{1}{n}\}_{n\in\mathbb{N}} \wedge a_0 = 0$

We try to understand the generating function $\mathcal{G}(a_n)$:

$$\begin{aligned} \mathcal{G}(\frac{1}{n}) &= t + \frac{1}{2}t^2 + \frac{1}{3}t^3 + \frac{1}{4}t^4 + \ldots \\ \frac{a_{n+1}}{a_n} &= \frac{n}{n+1} \quad \text{by ratio of consecutive terms} \\ (n+1)a_{n+1} &= na_n \end{aligned}$$

In order to apply the identity principle, it is required to check the equivalence holds for all n. But this isn't the case for $n = 0 : a_1 = 1 \neq 0 = a_0$. So we've to augment the second member with the term $\delta_{n,k}$ such that $\delta_{n,k} = 1$ if $n = k, 0$ otherwise. Augmenting we obtain $(n+1)a_{n+1} = na_n + \delta_{n,0}, \forall n \in \mathbb{N}$, in particular for $n = 0 : a_1 = 1 = 1 = a_0 + \delta_{0,0}$. Now it is possible to apply the identity principle, calling $b_{n+1} = (n+1)a_{n+1}$ (hence $b_n = na_n$ and $b_0 = 0$):

$$
\begin{aligned}
b_{n+1} &= na_n + \delta_{n,0} \\
&\downarrow \\
\mathcal{G}(b_{n+1}) &= \mathcal{G}(na_n) + \mathcal{G}(\delta_{n,0}) \\
\frac{\mathcal{G}(b_n) - b_0}{t} &= \mathcal{G}(na_n) + \mathcal{G}(\delta_{n,0}) \\
\frac{\mathcal{G}(na_n)}{t} &= \mathcal{G}(na_n) + \mathcal{G}(\delta_{n,0}) \\
\frac{t\frac{\partial}{\partial t}(\mathcal{G}(a_n))}{t} &= \mathcal{G}(na_n) + \mathcal{G}(\delta_{n,0}) \\
\frac{\partial}{\partial t}(\mathcal{G}(a_n)) &= t\frac{\partial}{\partial t}(\mathcal{G}(a_n)) + 1 \\
\frac{\partial}{\partial t}(\mathcal{G}(a_n)) &= \frac{1}{1-t} \\
\mathcal{G}(a_n) &= \log\frac{1}{1-t}
\end{aligned}
$$

Simple check:

```
(%i15) b(t):=log(1/(1-t))$
       taylor(b(t),t,0,13);
```

(%o16) $t + \frac{t^2}{2} + \frac{t^3}{3} + \frac{t^4}{4} + \frac{t^5}{5} + \frac{t^6}{6} + \frac{t^7}{7} + \frac{t^8}{8} + \frac{t^9}{9} + \frac{t^{10}}{10} + \frac{t^{11}}{11} + \frac{t^{12}}{12} + \frac{t^{13}}{13} + \ldots$

1.4.9 $\{H_n = \sum_{k=1}^{n} \frac{1}{k}\}_{n\in\mathbb{N}}$

$$\mathcal{G}(\sum_{k=1}^{n} \frac{1}{k}) = \mathcal{G}(\sum_{k=1}^{n} \frac{1}{k}1_k) = \mathcal{G}(\frac{1}{k})\mathcal{G}(1_k) = \left(\log\frac{1}{1-t}\right)\frac{1}{1-t} \quad \text{by convolution of } \mathcal{G}$$

Simple check:

```
(%i66) harmonic(n) := sum(1/k,k,1,n);
       b(t):=log(1/(1-t))*(1/(1-t));
       map(harmonic, makelist(n,n,1,10));
       taylor(b(t),t,0,10);
```

(%o66) $\operatorname{harmonic}(n) := \sum_{k=1}^{n} \frac{1}{k}$

(%o67) $b(t) := \log\left(\frac{1}{1-t}\right)\frac{1}{1-t}$

(%o68) $[1, \frac{3}{2}, \frac{11}{6}, \frac{25}{12}, \frac{137}{60}, \frac{49}{20}, \frac{363}{140}, \frac{761}{280}, \frac{7129}{2520}, \frac{7381}{2520}]$

(%o69) $t + \frac{3\,t^2}{2} + \frac{11\,t^3}{6} + \frac{25\,t^4}{12} + \frac{137\,t^5}{60} + \frac{49\,t^6}{20} + \frac{363\,t^7}{140} + \frac{761\,t^8}{280} + \frac{7129\,t^9}{2520} + \frac{7381\,t^{10}}{2520} + \ldots$

1.4.10 $\{F_{n+2} = F_{n+1} + F_n\}_{n\in\mathbb{N}} \wedge F_1 = 1 \wedge F_0 = 0$

It is possible to apply the *Identity Principle* directly because the equality $F_{n+2} = F_{n+1} + F_n$ holds $\forall n \in \mathbb{N}$ (would be problem if we formulate the same as $F_n = F_{n-1} + F_{n-2}$ for $n = 0$):

$$\begin{aligned}
F_{n+2} &= F_{n+1} + F_n \\
&\downarrow \\
\mathcal{G}(F_{n+2}) &= \mathcal{G}(F_{n+1}) + \mathcal{G}(F_n) \\
\frac{\mathcal{G}(F_n) - F_0 - F_1 t}{t^2} &= \frac{\mathcal{G}(F_n) - F_0}{t} + \mathcal{G}(F_n) \qquad \text{by translation of } \mathcal{G} \\
\frac{\mathcal{G}(F_n) - t}{t^2} &= \frac{\mathcal{G}(F_n)}{t} + \mathcal{G}(F_n) \\
\mathcal{G}(F_n) - t &= t\mathcal{G}(F_n) + t^2\mathcal{G}(F_n) \\
\mathcal{G}(F_n) &= -\frac{t}{t^2 + t - 1} = \frac{1}{1 - t - t^2}
\end{aligned}$$

Simple check:

```
(%i106) b(t):=(1/(1-t-t^2))$
        map(fib, makelist(n,n,1,10));
        taylor(b(t),t,0,9);
```

(%o108) $[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]$
(%o109) $1 + t + 2\,t^2 + 3\,t^3 + 5\,t^4 + 8\,t^5 + 13\,t^6 + 21\,t^7 + 34\,t^8 + 55\,t^9 + \ldots$

1.4.11 $\{T_{n+3} = T_{n+2} + T_{n+1} + T_n\}_{n\in\mathbb{N}} \wedge T_2 = 1 \wedge T_1 = 1 \wedge T_0 = 0$

It is possible to apply the *Identity Principle* directly because the equality $T_{n+3} = T_{n+2} + T_{n+1} + T_n$ holds $\forall n \in \mathbb{N}$:

$$\begin{aligned}
T_{n+3} &= T_{n+2} + T_{n+1} + T_n \\
&\downarrow \\
\mathcal{G}(T_{n+3}) &= \mathcal{G}(T_{n+2}) + \mathcal{G}(T_{n+1}) + \mathcal{G}(T_n) \\
\frac{\mathcal{G}(T_n) - T_0 - T_1 t - T_2 t^2}{t^3} &= \frac{\mathcal{G}(T_n) - T_0 - T_1 t}{t^2} + \frac{\mathcal{G}(T_n) - T_0}{t} + \mathcal{G}(T_n) \quad \text{by translation of } \mathcal{G} \\
\frac{\mathcal{G}(T_n) - t - t^2}{t^3} &= \frac{\mathcal{G}(T_n) - t}{t^2} + \frac{\mathcal{G}(T_n)}{t} + \mathcal{G}(T_n) \\
\mathcal{G}(T_n) - t - t^2 &= t(\mathcal{G}(T_n) - t) + t^2\mathcal{G}(T_n) + t^3\mathcal{G}(T_n) \\
\mathcal{G}(T_n) &= \frac{1}{1 - t - t^2 - t^3}
\end{aligned}$$

Simple check:

```
(%i9)   tribonacci(n):=if is(n=0) then 0 else if is(n=1) then 1 else if
        is(n=2) then 1 else tribonacci(n-1)+tribonacci(n-2)+tribonacci(n-3)$
        b(t):=(1/(1-t-t^2-t^3));
        map(tribonacci, makelist(n,n,1,10));
        taylor(b(t),t,0,9);
```

(%o10) $b(t) := \frac{1}{1 - t - t^2 - t^3}$
(%o11) $[1, 1, 2, 4, 7, 13, 24, 44, 81, 149]$
(%o12) $1 + t + 2\,t^2 + 4\,t^3 + 7\,t^4 + 13\,t^5 + 24\,t^6 + 44\,t^7 + 81\,t^8 + 149\,t^9 + \ldots$

1.4.12 $\{a_n = \binom{p}{n}\}_{n \in \mathbb{N}}$

There isn't a property ready for direct application, so we study the ratio between consecutive terms:

$$\begin{aligned}
\frac{a_{n+1}}{a_n} = \frac{\binom{p}{n+1}}{\binom{p}{n}} &= \frac{p!}{(n+1)!(p-n-1)!} \frac{n!(p-n)!}{p!} = \frac{p-n}{n+1} \\
&\downarrow \\
(n+1)a_{n+1} &= (p-n)a_n
\end{aligned}$$

Before applying the *Identity Principle* we check if the equivalence holds for all n, in particular for $n = 0 : a_1 = p = p = pa_0$. So we can apply the principle, calling $b_n = na_n$:

$$\begin{aligned}
b_{n+1} &= (p-n)a_n \\
&\downarrow \\
\mathcal{G}(b_{n+1}) &= \mathcal{G}((p-n)a_n) \\
\frac{\mathcal{G}(b_n)-b_0}{t} &= \mathcal{G}((p-n)a_n) \\
\frac{\mathcal{G}(b_n)}{t} &= \mathcal{G}((p-n)a_n) \\
\frac{\mathcal{G}(na_n)}{t} &= \mathcal{G}((p-n)a_n) \\
\frac{t\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right)}{t} &= \mathcal{G}((p-n)a_n) \\
\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right) &= p\mathcal{G}(a_n) - \mathcal{G}(na_n) \\
\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right) &= p\mathcal{G}(a_n) - t\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right) \\
(1+t)\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right) &= p\mathcal{G}(a_n) \\
\frac{a'(t)}{a(t)} &= \frac{p}{1+t} \quad \text{by calling } a(t) = \mathcal{G}(a_n) \\
\log a(t) &= \int \frac{p}{1+t}dt = p\log(1+t) + c
\end{aligned}$$

We compute the constant c for $t = 0 : \log 1 = p\log 1 + c$, with $a(0) = a_0 + 0t + 0t^2 + 0t^3 + \ldots = 1$, hence $c = 0$.

$$\begin{aligned}
\log a(t) &= \log{(1+t)^p} \\
&\downarrow \\
a(t) &= (1+t)^p
\end{aligned}$$

Simple check:

```
(%i45) b(t):=p/(1+t);
       integrate(b(t),t);
       a(t) := (1+t)^p$
       curriedBinomial(n):=binomial(p,n)$
       map(ratsimp, map(curriedBinomial, makelist(n,n,0,5)));
       taylor(a(t),t,0,5);
```

(%o45) $b\left(t\right) := \frac{p}{1+t}$

(%o46) $p\log\left(t+1\right)$

(%o49) $[1, p, \frac{p^2-p}{2}, \frac{p^3-3p^2+2p}{6}, \frac{p^4-6p^3+11p^2-6p}{24}, \frac{p^5-10p^4+35p^3-50p^2+24p}{120}]$

(%o50) $1+p\,t+\frac{\left(p^2-p\right)\,t^2}{2}+\frac{\left(p^3-3\,p^2+2\,p\right)\,t^3}{6}+\frac{\left(p^4-6\,p^3+11\,p^2-6\,p\right)\,t^4}{24}+\frac{\left(p^5-10\,p^4+35\,p^3-50\,p^2+24\,p\right)\,t^5}{120}+\ldots$

1.4.13 $\{a_n = \binom{2n}{n}\}_{n\in\mathbb{N}}$

There isn't a property ready for direct application, so we study the ratio between consecutive terms:

$$\frac{a_{n+1}}{a_n} = \frac{\binom{2n+2}{n+1}}{\binom{2n}{n}} = \frac{(2n+2)!}{(n+1)!(n+1)!}\frac{n!n!}{(2n)!} = \frac{(2n+2)(2n+1)}{(n+1)^2} = \frac{2(2n+1)}{n+1}$$

$$\downarrow$$

$$(n+1)a_{n+1} = 2(2n+1)a_n$$

Before applying the *Identity Principle* we check if the equivalence holds for all n, in particular for $n = 0 : a_1 = 2 = 2 = 2a_0$ and $a_0 = 1$. So we can apply the principle, calling $b_n = na_n$:

$$\begin{aligned}
b_{n+1} &= 2(2n+1)a_n \\
&\downarrow \\
\mathcal{G}(b_{n+1}) &= \mathcal{G}(2(2n+1)a_n) \\
\frac{\mathcal{G}(b_n)-b_0}{t} &= \mathcal{G}(2(2n+1)a_n) \\
\frac{\mathcal{G}(b_n)}{t} &= \mathcal{G}(2(2n+1)a_n) \\
\frac{\mathcal{G}(na_n)}{t} &= \mathcal{G}(2(2n+1)a_n) \\
\frac{t\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right)}{t} &= \mathcal{G}(2(2n+1)a_n) \\
\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right) &= 4\mathcal{G}(na_n) + 2\mathcal{G}(a_n) \\
\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right) &= 4t\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right) + 2\mathcal{G}(a_n) \\
(1-4t)\frac{\partial}{\partial t}\left(\mathcal{G}(a_n)\right) &= 2\mathcal{G}(a_n) \\
\frac{a'(t)}{a(t)} &= \frac{2}{1-4t} \qquad \text{by calling } a(t) = \mathcal{G}(a_n) \\
\log a(t) &= 2\int \frac{1}{1-4t}dt = -\frac{1}{2}\log(1-4t) + c
\end{aligned}$$

We compute the constant c for $t = 0 : \log 1 = -\frac{1}{2}\log 1 + c$, with $a(0) = a_0 + 0t + 0t^2 + 0t^3 + \ldots = 1$, hence $c = 0$.

$$\begin{aligned} \log a(t) &= \log \frac{1}{\sqrt{1-4t}} \\ &\downarrow \\ a(t) &= \frac{1}{\sqrt{1-4t}} \end{aligned}$$

Simple check:

```
(%i63) b(t):=2/(1-4*t);
       integrate(b(t),t);
       a(t) := 1/sqrt(1-4*t)$
       curriedBinomial(n):=binomial(2*n,n)$
       map(ratsimp, map(curriedBinomial, makelist(n,n,0,5)));
       taylor(a(t),t,0,5);
```

(%o63) $b(t) := \dfrac{2}{1-4t}$

(%o64) $-\dfrac{\log(1-4t)}{2}$

(%o67) $[1, 2, 6, 20, 70, 252]$

(%o68) $1 + 2t + 6t^2 + 20t^3 + 70t^4 + 252t^5 + \ldots$

1.4.14 $\{a_n = \frac{1}{n+1}\binom{2n}{n}\}_{n\in\mathbb{N}}$

$$\mathcal{G}(\frac{1}{n+1}\binom{2n}{n}) = \frac{1}{t}\int \mathcal{G}(\binom{2n}{n})dt = \frac{1}{t}\int \frac{1}{\sqrt{1-4t}}dt =$$

Calling $\sqrt{1-4t} = y$ follows $t = \frac{1-y^2}{4}$ and $dt = -\frac{1}{2}y\,dy$, so:

$$\begin{aligned} &= \frac{1}{t}\int_0^t \frac{1}{y}\left(-\frac{1}{2}y\right)dy = -\frac{1}{2t}\int_0^t dy = -\frac{1}{2t}[y]_0^t = -\frac{1}{2t}\left[\sqrt{1-4t}\right]_0^t \\ &= -\frac{1}{2t}(\sqrt{1-4t}-1) = \frac{1-\sqrt{1-4t}}{2t} \end{aligned}$$

Simple check:

```
(%i167) b(t):=(1-sqrt(1-4*t))/(2*t)$
        catalan(n):=binomial(2*n,n)/(n+1)$
        map(catalan, makelist(n,n,0,10));
        taylor(b(t),t,0,10);
```

(%o169) $[1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796]$

(%o170) $1 + t + 2t^2 + 5t^3 + 14t^4 + 42t^5 + 132t^6 + 429t^7 + 1430t^8 + 4862t^9 + 16796t^{10} + \ldots$

2 SEQUENTIAL SEARCH SIMULATION

We've repeated the simulation for the *sequential search* problem as did during the class. We want to study the mean of checks performed by the sequential search algorithm before the desired element is found in the given permutation. After we compare the mean of checks obtained by simulation with the theoretical mean value for the same input dimension.

The sequential search algorithm cited above is a very simple searching method: it consume a permutation of integer of a fixed dimension n and a target integer; produce true if the target belongs to the given permutation, else otherwise (in our case we augment the information returned with the number of checks needed in the run). The searching strategy consists of starting from the very left and moving one step to right every time the target is missed.

2.1 IMPLEMENTATION

We don't report here the description for the implementation of the sequential search algorithm because is very simple. Instead, we focus the explanation on the main simulation procedure.

The simulation function consume three parameters:

- $numdimensions$, the number of dimensions that we would like to test;
- $interval$, the multiplier to build the permutation vector to be used during the search;
- $attempts$, the number of application of the sequential search algorithm to a given permutation.

We use the number of dimensions $numdimensions$ to build a vector $dimensions$ such that both of the following hold:

$$\begin{aligned} length(dimensions) &= numdimensions \\ dimensions[i] &= i * interval \quad \forall i \in \{1, \dots, numdimensions\} \end{aligned}$$

For a given dimension $n \in \{interval, 2 * interval, \dots, numdimensions * interval\}$, we sample a permutation of integers from $\{1, \dots, n\}$ using the uniform distribution. For each permutation we apply the sequential search algorithm: the number of application is proportional to n (specifically $2n$) and after every application we record the number of checks needed to hit the target. When we run out all the iterations we compute the mean and the variance of the checks, storing them into auxiliary vectors.

2.2 RESULTS

In Table 3 we report a summary table for a simulation invoked with arguments $numdimensions = 50, interval = 20, attempts = 50$. Observing the table we can see that the simulation is quite correct, the theoretical values matches the simulated ones with little differences.

We've used the standardized means to plot them against the normal distribution in order to verify the Central Limit Theorem in Figure 1. The dotted curve represent the normal distribution, the blue one represent the sampling means distribution instead. Observing the plot we can say that the approximation is quite good, however the max dimension is $n = 50$, this allow us to apply the theorem but we could gain precision if we'll repeat our simulation for a greater value of n.

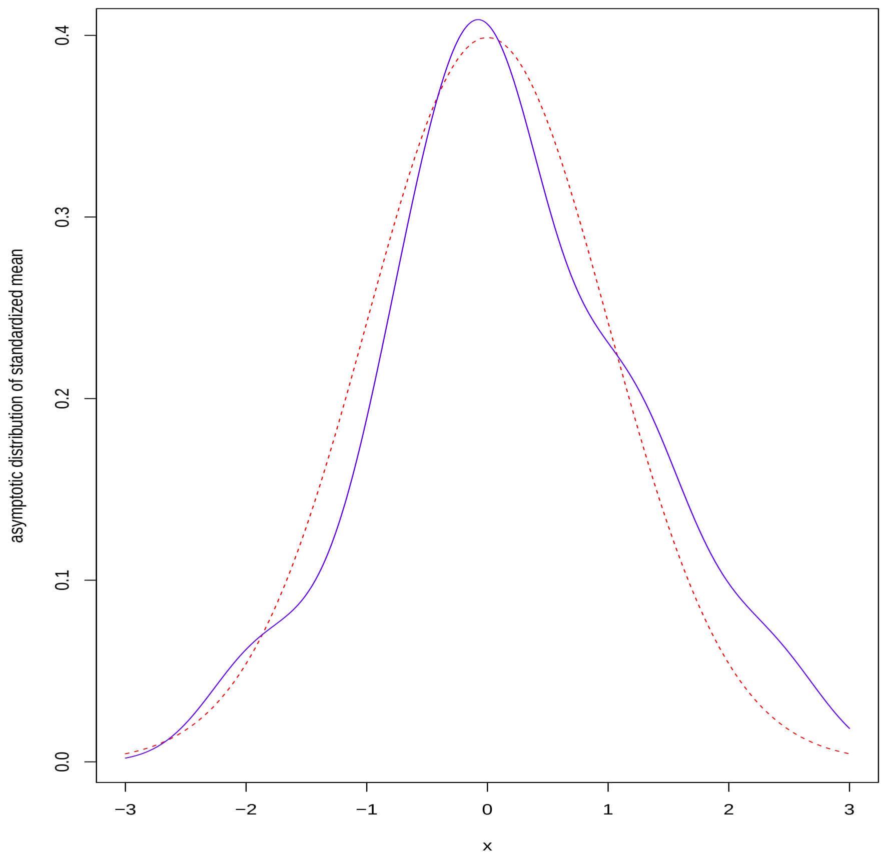

Figure 1: Plot of standardized means

Also we've build a second plot with a regression of the sampling means in Figure 2. The red line is drawn after computing the intercept and the coefficient using an hybrid model. Our attempt to explain the data consists of a mixture up to the second degree, as follow:

$$
\begin{aligned}
dims &= \sum_{i}^{n} dimensions[i] \\
means &= \sum_{i}^{n} mean[i] \\
dim\_squares &= \sum_{i}^{n} dimensions[i]^2 \\
mean\_squares &= \sum_{i}^{n} means[i]^2 \\
dim\_mean &= \sum_{i}^{n} means[i] * dimensions[i] \\
intercept &= \frac{dim\_squares * means - dims * dim\_mean}{n * dim\_squares - dims^2} \\
coefficient &= \frac{n * dim\_mean - dims * means}{n * dim\_squares - dims^2}
\end{aligned}
$$

where n is our $numdimensions$ cited above, hence the red line is defined as $means = coefficient * dimensions + intercept$. Using an R interpreter we can see our numerical output:

```
coefficient = 0.49, intercept = 0.66
square of correlation index = 0.999
```

The regression is almost perfect, hence there exists a strong linear relation between means and distribution (from a statistical point of view we can conclude with the following observation: for an increment of the dimensions of one unit we get an increment of about a half unit on the number of checks).

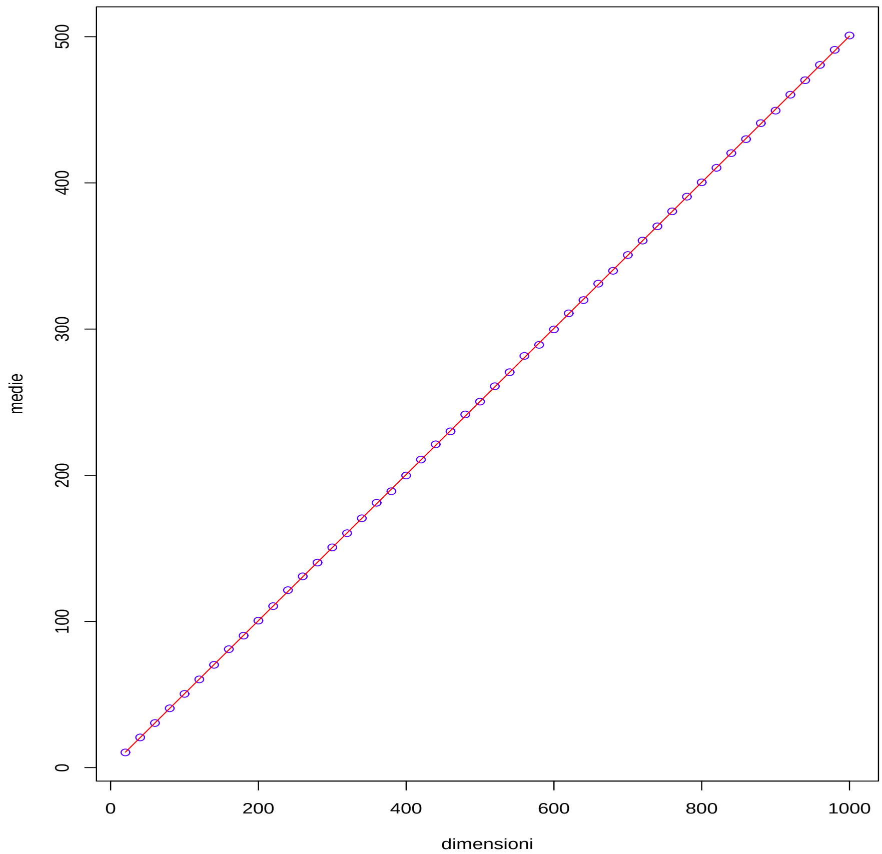

Figure 2: Plot of means regression

Table 3: Sequential search summary

	dimensions	theo means	means	theo vars	vars	theo var of vars	var of vars	stand. means	stand. vars
1	20.00	10.50	10.66	33.25	34.34	877.80	975.55	1.22	1.64
2	40.00	20.50	20.65	133.25	135.02	14177.80	14994.21	0.82	0.94
3	60.00	30.50	30.48	299.92	302.46	71900.02	74572.74	-0.10	0.74
4	80.00	40.50	41.09	533.25	531.28	227377.80	221994.03	2.28	-0.37
5	100.00	50.50	50.52	833.25	833.01	555277.80	553196.84	0.07	-0.03
6	120.00	60.50	60.13	1199.92	1189.70	1151600.02	1122368.13	-1.18	-1.04
7	140.00	70.50	70.42	1633.25	1621.51	2133677.80	2078575.00	-0.22	-0.95
8	160.00	80.50	79.80	2133.25	2155.97	3640177.80	3782358.53	-1.92	1.51
9	180.00	90.50	91.06	2699.92	2696.73	5831100.02	5837985.82	1.44	-0.18
10	200.00	100.50	100.26	3333.25	3315.86	8887777.80	8760281.48	-0.59	-0.82
...									
41	820.00	410.50	409.84	56033.25	56005.31	2511768877.80	2520823155.84	-0.80	-0.16
42	840.00	420.50	419.93	58799.92	59131.53	2765932400.02	2833053962.68	-0.68	1.83
43	860.00	430.50	429.49	61633.25	61801.47	3038913677.80	3061925029.95	-1.20	0.89
44	880.00	440.50	441.59	64533.25	64686.44	3331619377.80	3349812105.46	1.27	0.79
45	900.00	450.50	448.81	67499.92	67476.72	3644977500.02	3620068539.03	-1.95	-0.12
46	920.00	460.50	461.97	70533.25	70360.59	3979937377.80	3963315767.79	1.68	-0.83
47	940.00	470.50	470.75	73633.25	73791.33	4337469677.80	4381479940.27	0.28	0.74
48	960.00	480.50	481.49	76799.92	76684.82	4718566400.02	4704539498.17	1.11	-0.52
49	980.00	490.50	490.41	80033.25	80176.91	5124240877.80	5142289190.74	-0.10	0.63
50	1000.00	500.50	500.43	83333.25	83210.10	5555527777.80	5524755813.88	-0.08	-0.52

3 GENERATING BINARY TREES AT RANDOM

In this section we'll explain our work on generation of binary trees at random. We're interested to setup a simulation to study the means of the number of leaves in trees with n nodes and comparing the result of the simulation against theoretical results.

In order to do this we've implemented an algorithm to generate binary trees: it consume $n \in \mathbb{N}$ and produce a binary tree with n nodes.

We've repeated the application of that algorithm $k(>> n)$ times in order to check if the algorithm is an *uniform* binary tree generator, that is, if the generator would be perfect, each tree with n nodes should have $\frac{1}{\frac{1}{n+1}\binom{2n}{n}}$ probability to be generated.

The last point of our work is to check a theoretical result about the mean of number of leaves among binary trees with n nodes.

3.1 ATKINSON AND SACK ALGORITHM

The set of binary trees with n nodes is in one-to-one correspondence with many other sets of combinatorial objects, one of them is the set of well-formed bracket sequences with n pairs of brackets. The Atkinson and Sack algorithm focus on generating those sequences: in this section we'll explain in our words their solution (in chapter 4 is reported the original article).

3.1.1 *Definitions*

A sequence is a word $w \in \{(,)\}^*$. In the following we draw a sequence as a zigzag walk which starts from a base line (represented as - - -) and for each $(\in w$ we draw the character slash, for each $) \in w$ we draw the character backslash. For instance, the sequence of brackets $())()((())$ is drawn as:

```
           /\
 /\       /  \
--------------
    \/\/
```

A word is *balanced* if it contains an equal number of (and), as in the sequence above. Graphically speaking, balanced words are sequences that start from the base line and, at right-most, return to the base line (in the middle it is possible to cross zero, one or more time the base line).

A word w is *reducible* if $w = w_1 w_2$ where w_1 and w_2 are *balanced* and *non empty*. For instance the sequence reported above is reducible in four words while the following one isn't:

```
   /\
  /  \/\
 /      \
----------
```

A *balanced* word w has *defect* i if w has i bracket pairs under the base line. Words with *defect* 0 are called *well-formed*. The previous word has *defect* 0, while the following one has *defect* 3:

```
 /\        /\
----------------
    \/\  /
       \/
```

A word w^* is the complement of a word w if $\forall i \in \{0, \dots, |w|\} : w[i] = (\leftrightarrow w^*[i] =)$, for instance the complement of the previous word is:

```
       /\
    /\/  \
--------------
 \/        \/
```

If a *balanced* word w is *non reducible* then either w or w^* is *well-formed*. We can see this result assuming w be one of the following words:

```
              ----------
   /\          \      /
  /  \/\        \  /\/
 /      \        \/
----------
```

Both of them are *balanced* and *non reducible*: if w is the left-most one then w is *well-formed*, if w is the right-most one then w^* is *well-formed*.

3.1.2 *Splitting a* reducible *word*

How can we decide if a word w is *reducible* or not? We can do a cumulative summation on w where we consider each (as 1 and each) as -1. The following word has the cumulative sums $(0, -1, 0, 1, 0, 1, 2, 1, 0, -1, 0)$:

```
       /\
    /\/  \
--------------
 \/        \/
```

The following words have cumulative sums $(0,1,2,3,2,1,2,1,0)$ and $(0,-1,-2,-3,-2,-1,-2,-1,0)$ respectively:

```
             ----------
   /\         \      /
  /  \/\       \  /\/
 /      \       \/
----------
```

Let w be a word and $s = (0, s_1, \ldots, s_n)$ be the sequence of cumulative sums respect of w, where $n = |w|$. We split $w = uv$ such that $|u| > 0 \wedge |v| \geqslant 0$ using the following strategy:

$$
\begin{aligned}
i &= \min\{k \in \{1, \ldots, n\} : s_k = 0\} \\
u &= (w_1, \ldots, w_i) \quad v = (w_{i+1}, \ldots, w_n)
\end{aligned}
$$

For instance, the following sequence as cumulative sums $(0,1,0,1,2,1,0)$, hence $i = 2$ so $u = ()$ and $v = (())$:

```
    /\
 /\/  \
--------
```

While the following one as cumulative sums $(0,1,2,3,2,1,2,1,0)$, hence $i = 8$ so $u = ((())())$ and v empty:

```
   /\
  /  \/\
 /      \
----------
```

3.1.3 *Generating a* balanced *word*

In order to generate a *balanced* word of length $2n$ (not necessary *well-formed*) we use the follow strategy:

1. generate a uniform sample L of length n from $\{1, \ldots, 2n\}$;
2. let $w = w_1 w_2 \ldots w_{2n}$ a word such that $w_i = (\leftrightarrow i \in L$;

3.1.4 *Transform a* balanced *word in a* well-formed *word*

Let $w = uv$ a *balanced* word obtained using the strategy described in the previous section. To get a *well-formed* word from w we define a function $\phi : \{(,)\}^* \to \{(,)\}^*$ inductively as follow (ϵ represent the empty string):

$$\phi(\epsilon) = \epsilon$$
$$\phi(w) = u\phi(v) \quad \text{if } u \text{ is } \textit{well-formed}$$
$$\phi(w) = (\phi(v))t^* \quad \text{if } u =)t(\text{ is } \textit{not well-formed}$$

In order to recognize if a word w is *well-formed* it is sufficient to compute the cumulative sequence of sums s and check that $\forall i : s_i \geqslant 0$. Let's do some examples:

```
        /\
----------
 \   /
  \/
```

The previous sequence represents the word $w =))((()$. It is *balanced* so applying ϕ we get $\phi(w) = (\phi(v))()$ where $u =)t($, $t =)($ and $v = ()$. Now $\phi(v) = ()$ because $v = ()$ is *well-formed*, hence $\phi(w) = (())()$.

Another limit example:

```
----------
 \       /
  \   /\/
   \/
```

The previous sequence represents the word $w =)))(()(($. It is *balanced* so applying ϕ we get $\phi(w) = (\phi(v))(())()$ where $u =)t($, $t =))(()($ and $v = \epsilon$. Now $\phi(v) = \epsilon$ hence $\phi(w) = ()(())()$, graphically:

```
     /\
 /\/   \/\
-----------
```

3.1.5 *Complexity and Space*

Let w a *balanced* word and $T(n)$ the number of operation required. The decomposition $w = uv$ can be computed in $T(r)$ where $n \geqslant r = |u|$ by cumulative summations, remaining $T(n-r)$ operation for v. Hence $T(n) = O(r) + T(n-r) = O(n)$ a linear time.

For a space analysis, the algorithm use integers up to $2n$.

3.2 IMPLEMENTATION USING R

```r
generate.tree <- function(number_of_nodes){
  word_dimension <- 2 * number_of_nodes
  universe <- 1:word_dimension
  sample <- sample(universe, size=number_of_nodes)
  w = rep(0, word_dimension)
  for (i in 1:word_dimension) {
    w[i] <- ifelse(any(sample == i), 1, -1)
  }
  phi=phi(w)
  list(word=w, phi=phi, as_brackets = brackets_of_word(phi))
}

split.word <- function(w){
  if(length(w) == 0){
    return(list(u=c(), v=c()))
  }
  u_index_set <- 1:match(0, cumsum(w))
  list(u=w[u_index_set], v=w[-u_index_set])
}

phi <- function(w){
  if(length(w) == 0){
    return(w)
  }
  split <- split.word(w)
  if(all(cumsum(split$u) > -1)){
    return (c(split$u, phi(split$v)))
  }
  else{
    t = split$u[-c(1, length(split$u))]
    return (c(1, phi(split$v), -1, -t))
  }
}
```

3.3 CHECKING RANDOMNESS

In order to establish if the algorithm is an *uniform random* generator, we perform a χ^2 test on the generated trees. We've implemented that process in R and we're going to comment the result obtained.

In Table 4 we report the p-values obtained performing a simulation on trees with $4, 5, 6, 8, 10$ nodes (in columns) and for each dimension, we build $1000, 2000, 5000, 10000, 20000, 50000$ trees (in rows). In order to be considered an *uniform random* generator, each p-value ν should be $.1 \leqslant \nu \ \leqslant .9$. Our simulation

	4	5	6	8	10
1000	0.60	0.85	0.54	1.00	1.00
2000	0.99	0.98	0.13	1.00	1.00
5000	0.36	0.85	0.28	1.00	1.00
10000	0.69	0.73	0.11	0.65	1.00
20000	0.63	0.23	0.50	0.16	1.00
50000	0.09	0.74	0.63	0.90	1.00

Table 4: p-values per trees and nodes

is quite good, there are two strange p-values for dimension 2000 and trees $4, 5$ where we find $.99$ and $.98$. Maybe the chance has played its role at the running time. Instead, the cell containing 1s means that the number of trees isn't sufficient to perform a χ^2 test (there are 16796 different trees with 10 nodes, with all simulated dimension isn't possible to test randomness).

In table Table 5 we report the empirical number of trees calculated for trees with some nodes, each a different number of times (as in the case above). We can

	4	5	6	8	10
1000	11.08	31.69	128.86	1456.74	16770.17
2000	4.39	24.36	149.75	1454.88	17046.66
5000	14.18	31.62	140.13	1378.37	16478.72
10000	10.09	35.05	151.54	1414.26	17028.12
20000	10.75	47.34	130.37	1483.03	16813.47
50000	20.26	34.83	125.10	1359.54	16778.21

Table 5: Empirical number of trees per nodes

see a correspondence of "strange" values in p-values table and in the empirical number of trees table: where the p-values ν indicate a non random sequence we've a corresponding empirical value very different respect the others in the same column.

It is possible to compare the values contained in table Table 5 with those (theoretically correct) contained in **??**.

number of nodes	4	5	6	8	10
number of trees	14	42	132	1430	16796

Table 6: Number of trees per nodes

3.4 MEANS OF LEAVES AND HEIGHTS

In this section we study the means of the number of leaves and of heights of all different binary trees with n nodes.

From theory we know that the mean of leaves of trees with n nodes is:

$$\frac{n\,(n+1)}{2\,(2\,n-1)}$$

Using Maxima to have the exacted means for some nodes dimensions:

```
(%i178) fpprintprec:4$
        leaves(n):=(n*(n+1))/(2*(2*n-1));
        combineResult(n):=[n,leaves(n)]$
        map(combineResult, makelist(n,n,1,10)),numer;
```

(%o179) $\mathrm{leaves}\,(n) := \frac{n\,(n+1)}{2\,(2\,n-1)}$

(%o181) $[[1,1],[2,1],[3,1.2],[4,1.429],[5,1.667],[6,1.909],[7,2.154],[8,2.4],[9,2.647],[10,2.895]]$

In Table **??** we report the results obtained with our implementation in R: the second column report the number of trees generated with the correspondent number of nodes.

3.5 STANDARDIZED MEANS AND ASYMPTOTIC DISTRIBUTIONS

In this section we study the asymptotic distribution of the means for the leaves and heights of trees with n nodes. We use the result of the *Central Limit Theorem* to check if the two means under study behave like a normal distribution in repeated sampling.

Each of the following plots are obtained repeating 1000 times the generation of 500 trees each with 5 nodes. In each plot the red dotted curve is the nor-

Table 7: Theoretical and empirical means for leaves and height

nodes	dimensions	theo.mean.leaves	theo.mean.height	emp.mean.leaves	emp.mean.height
3	100	1.20	2.80	1.16	2.84
4	200	1.43	3.57	1.35	3.65
5	300	1.67	4.24	1.70	4.23
6	1000	1.91	4.88	1.91	4.87
7	10000	2.15	5.47	2.15	5.46
8	10000	2.40	6.03	2.41	6.02
9	50000	2.65	6.56	2.65	6.56
10	100000	2.89	7.07	2.90	7.07

mal distribution (out "target"), while the blue continue curve is the inferred distribution.

In Figure 3 we report the standardized mean of leaves.

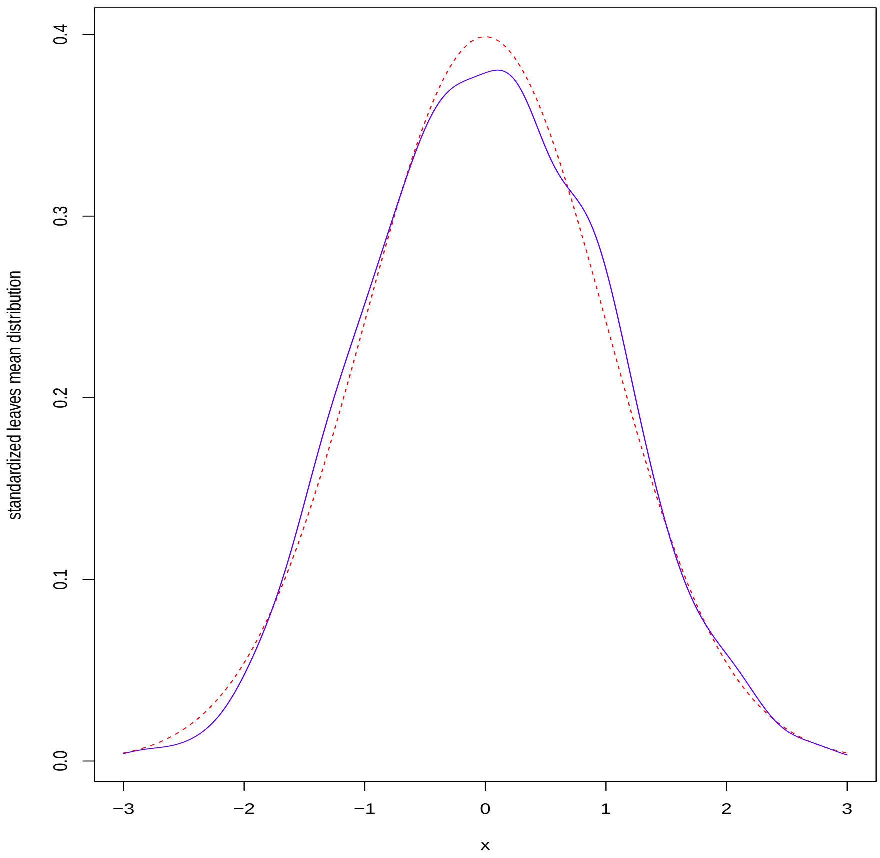

Figure 3: Standardized leaves mean distribution

In Figure 4 we report the standardized mean of leaves.

3.6 DRAWING TREES

Just for fun, our implementation draw all different trees with n nodes (this implementation is done in OCaml). In Figure 5 we report the image with all binary trees with 4 nodes.

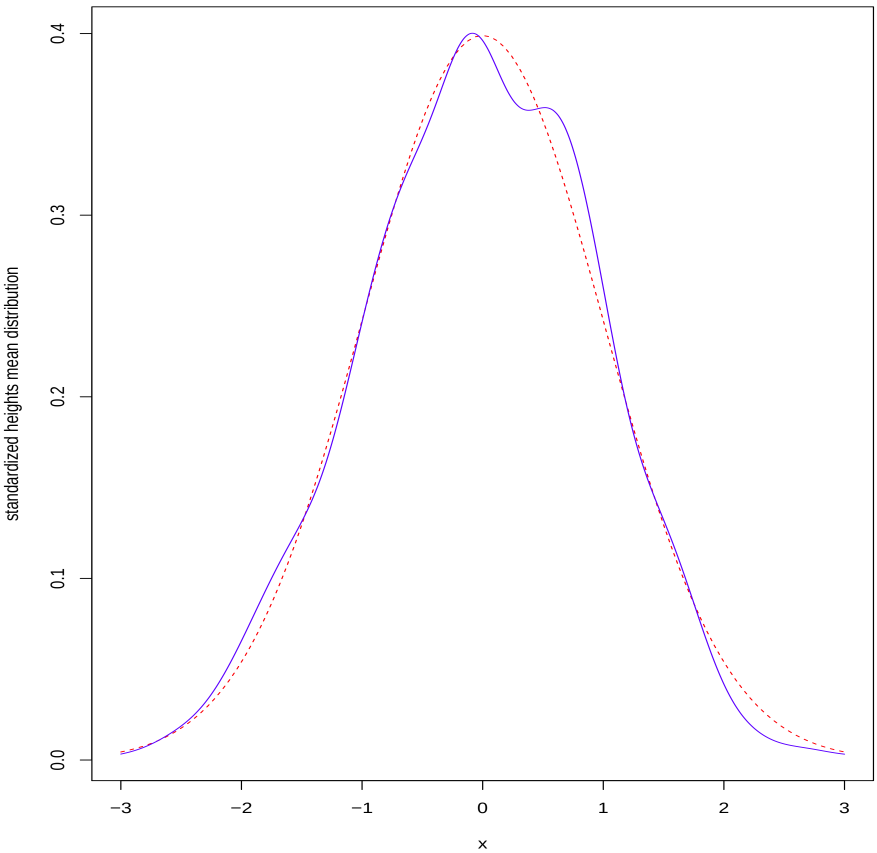

Figure 4: Standardized height mean distribution

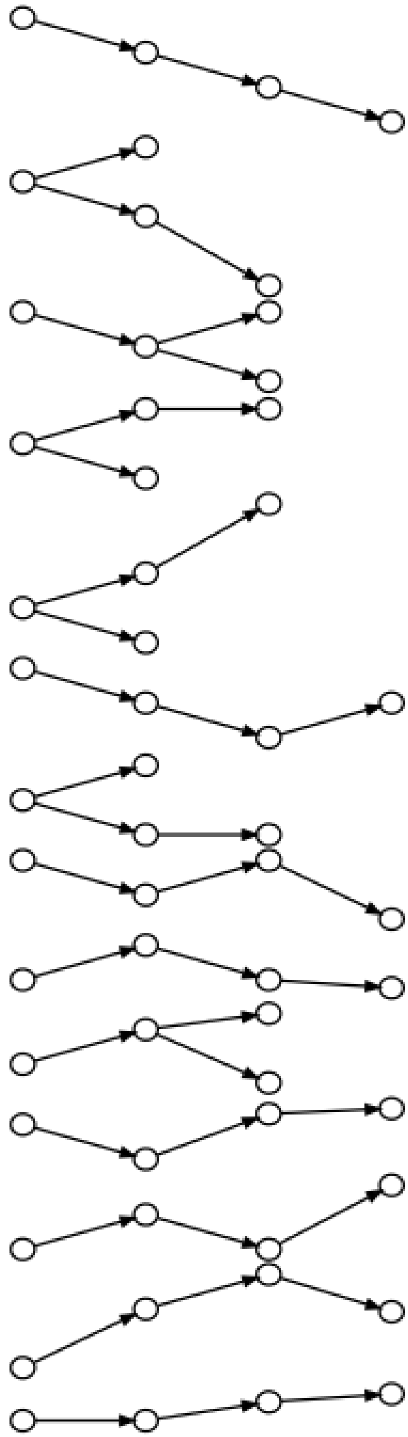

Figure 5: Binary trees with four nodes

4 APPENDIX

4.1 PROJECT DESCRIPTION

In this section we explain some details "under the hood" of the implementation of this project.

We've used the statistical environment R to use its very efficient functions to perform uniform sampling (required by the Atkinson and Sack algorithm) and χ^2 test in order to check the randomness of the generator. To produce the images with all different trees with n nodes, we've used the OCaml language to use its functional power.

The two programs works together: the entry point for the user are the R functions (which can be loaded in any R interpreter). During their computation, they invoke an OCaml program giving their argument in form a file written on the file system. The OCaml program parse the input file, compute the representation of trees and write them back in another file. The control return to R functions, which invoke the *graphviz* programs to build a representation of trees in image form.

4.2 ORIGINAL ARTICLE ABOUT RANDOM BINARY TREES GENERATION

In the last pages of this document we report the original article about the algorithm for random trees generation written by Atkinson and Sack (this isn't our work, we report it here because it is difficult to find and it is just for getting the reading experience of this document more complete).

Information Processing Letters 41 (1992) 21–23
North-Holland 21 January 1992

Generating binary trees at random

M.D. Atkinson and J.-R. Sack
School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6

Communicated by S.G. Akl
Received 21 December 1990
Revised 17 September 1991

Abstract

Atkinson, M.D. and J.-R. Sack, Generating binary trees at random, Information Processing Letters 41 (1992) 21–23.

We give a new constructive proof of the Chung–Feller theorem. Our proof provides a new and simple linear-time algorithm for generating random binary trees on n nodes; the algorithm uses integers no larger than $2n$.

Keywords: Analysis of algorithms, binary trees, bracket sequences, data structures

1. Introduction

Methods for generating binary trees on n nodes have been considered by several authors (see [4,8] and [6] also for additional references). In most cases the focus has been on generating all binary trees in some order or on ranking and unranking them. The number of binary trees on n nodes is the Catalan number $\binom{2n}{n}/(n+1)$ which is exponential in n ($\approx 4^n$) and so these computations cannot be carried out very easily unless n is small. Unranking algorithms allow binary trees to be generated uniformly at random and this is often more useful than being able to list all the possible trees. Unfortunately, numbers which are exponential in n enter these calculations and this makes them impracticable unless n is small.

The problem of generating binary trees uniformly at random without introducing exponentially large numbers was overcome by Arnold and Sleep [1] and Martin and Orr [6]; they gave linear time algorithms which used integers of size $O(n)$ for generating a random binary tree. We shall present a new and simpler solution having the same advantages. Our solution is based on a constructive version of the Chung–Feller theorem on coin-tossing, which has not previously been applied to this area. Our treatment also provides a new proof of the Chung–Feller theorem.

The set of binary trees on n nodes is well known to be in one-to-one correspondence with many other sets of combinatorial objects including rooted (ordered) trees with n branches, triangulations of a convex $(n+2)$-gon, lattice paths from $(0, 0)$ to (n, n) which do not cross the diagonal, and well-formed bracket sequences with n pairs of brackets. The one-to-one correspondences are explicit and efficient to compute in linear time; thus a uniform random generator for binary trees gives rise to a uniform random generator for all these other objects, and vice versa. We shall focus on generating well-formed sequences of brackets.

2. Terminology

We begin with some terminology. It is convenient to denote the left and right bracket symbols by λ and ρ respectively. Thus a bracket sequence (well formed or not) corresponds to a word over

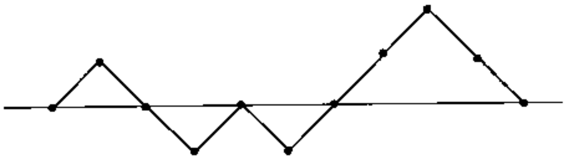

Fig. 1. The zigzag diagram corresponding to the word $\lambda\rho\rho\lambda\rho\lambda\lambda\lambda\rho\rho$.

the alphabet $\{\lambda, \rho\}$. A word such as $\lambda\rho\rho\lambda\rho\lambda\lambda\lambda\rho\rho$ may be pictured as a zigzag diagram drawn from some base line where each upward edge represents λ and each downward edge represents ρ (see Fig. 1).

A word is said to be *balanced* if it contains equal numbers of λ's and ρ's. Balanced words are precisely those whose diagram returns to the base line. A balanced word w is said to be *reducible* if it may be written $w = w_1w_2$ with w_1, w_2 each balanced and nonempty, otherwise w is *irreducible*.

A balanced word is defined to have *defect* i if its diagram has precisely $2i$ links below its base line. Defect 0 words are called *well-formed* and corresponds to well-nested bracket sequences. Observe that the defect of a word is easily found by a summation: we scan the word of length k from left to right regarding each λ as $+1$, each ρ as -1, and computing the partial sums 0, $s_1, \ldots, s_k$; the final sum is zero and the number of negative interim sums s_j at odd indices j is the defect. We call this calculation *partial summation*.

For any word w let w^* denote the result of replacing all occurrences of λ by ρ and ρ by λ. The following two results are immediate consequences of these definitions.

Lemma 1. *If a balanced word w is irreducible, then one of w and w^* is well-formed; in fact, $w = \lambda u \rho$ where u is well-formed, or $w = \rho u \lambda$ where u^* is well-formed. If a balanced word w is well-formed, then $\lambda w \rho$ is irreducible.*

Lemma 2. *A balanced word w has a unique factorisation as $w = w_1w_2 \cdots w_k$, where each w_i is irreducible. If w is well-formed, so is each w_i.*

3. The algorithm

Let B_n denote the set of $\binom{2n}{n}$ balanced words of length $2n$, and let B_{ni} denote the subset of balanced words of defect i. Clearly B_n is the disjoint union of $B_{n0}, B_{n1}, \ldots, B_{nn}$. The Chung–Feller Theorem, see [2, Theorem 2A] and [3, p.94], states that these subsets all have the same size. The central idea of our algorithm is to use a new constructive proof of this theorem which depends on explicit 1–1 correspondences between these sets.

Our algorithm has the following form:

Algorithm RANDOM BRACKET SEQUENCE

Input: An integer n.

Output: A well-formed word of length $2n$ over the alphabet $\{\lambda, \rho\}$.

1. Generate a uniformly random combination L of n integers from $\{1, 2, \ldots, 2n\}$
2. Define a random member $x = (x_1x_2 \cdots x_{2n})$ of B_{2n} by the rule $x_i = \lambda$ if $i \in L$, $x_i = \rho$ if $i \notin L$.
3. Return the well-formed member of B_{2n} to which x corresponds.

Steps 1 and 2 of the algorithm are straightforward. To generate a combination of n integers from $\{1, 2, \ldots, 2n\}$ uniformly at random, and hence a member of B_n, we may use the technique described, for example, in [5, p.137] (see also [7, pp. 189–198] for more discussion on this topic). This technique takes only linear time and uses only integers less than $2n$.

To implement step 3 we need to define some suitable correspondences. We denote by $|w|$ the length of w and by card(S) the cardinality of a set S. We now define a map $\Phi_n : B_n \to B_{n0}$. The definition is inductive. For $n = 0$, we define Φ_0 in the only way possible: it maps the empty string to the empty string. For $n > 0$ and $w \in B_n$ we begin by expressing w as $w = uv$, where u is irreducible, $|u| = r > 0$, $|v| = s \geqslant 0$; then we define Φ_n by the rules

$$\Phi_n(w) = u\Phi_s(v) \qquad \text{if } u \text{ is well-formed,}$$
$$\Phi_n(w) = \lambda\Phi_s(v)\rho t^* \qquad \text{if } u = \rho t \lambda \text{ is not well-formed.}$$

Theorem 3. *Φ_n is $n+1$ to 1 onto B_{n0} and is bijective on each B_{ni}.*

Corollary 4. (Chung and Feller) card(B_{ni}) = card(B_{n0}).

Corollary 5. *If w is a random variable distributed uniformly in B_n, then $\Phi_n(w)$ is distributed uniformly in B_{n0}.*

Proof. It is sufficient to show that Φ_n is a bijection $B_{ni} \to B_{n0}$ for each i. Suppose that $w_1, w_2 \in B_{ni}$ and that they have the same image under Φ_n. For $k = 1,2$ put $w_k = u_k v_k$, where u_k irreducible, and let $|u_k| = r_k$, $|v_k| = s_k$. There are four possibilities:

(1) u_1, u_2 are each well-formed. Then $v_1 \in B_{s_1,i}$, $v_2 \in B_{s_2,i}$, and $u_1\Phi_{s_1}(v_1) = \Phi_n(w_1) = \Phi_n(w_2) = u_2\Phi_{s_2}(v_2)$. By Lemma 2, $u_1 = u_2$, $\Phi_{s_1}(v_1) = \Phi_{s_2}(v_2)$ and so $v_1 = v_2$ by induction.

(2) Neither of u_1 and u_2 are well-formed, say $u_1 = \rho t_1 \lambda$ and $u_2 = \rho t_2 \lambda$. Then $v_1 \in B_{s_1,i-r_1}$, $v_2 \in B_{s_2,i-r_2}$, and $\lambda\Phi_{s_1}(v_1)\rho t_1^* = \lambda\Phi_{s_2}(v_2)\rho t_2^*$. The leading subwords $\lambda\Phi_{s_1}(v_1)\rho t_1^*$ and $\lambda\Phi_{s_2}(v_2)\rho t_2^*$ are irreducible, therefore equal. Therefore, by induction, $v_1 = v_2$ and $t_1 = t_2$, so $u_1 = u_2$.

(3) u_1 is well-formed and u_2 is not well-formed, say $u_2 = \rho t_2 \lambda$. Then $v_1 \in B_{s_1,i}$, $v_2 \in B_{s_2,i-r_2}$, and $u_1\Phi_{s_1}(v_1) = \lambda\Phi_{s_2}(v_2)\rho t_2^*$. By Lemma 2 again, $u_1 = \lambda\Phi_{s_2}(v_2)\rho$ and, taking lengths, $r_1 = s_2 + 2$ and $s_1 = r_2 - 2$. But $r_2 \leqslant i$ since $v_2 \in B_{s_2,i-r_2}$ and $i \leqslant s_1$ since $v_1 \in B_{s_1,i}$ from which it follows that $i \leqslant s_1 \leqslant r_2 - 2 < r_2 \leqslant i$, a contradiction.

(4) u_1 is not well-formed and u_2 is well-formed. This case is impossible for the same reasons as case 3.

This proves that Φ_n is one-to-one on B_{ni}. To prove that it maps B_{ni} onto B_{n0} it is enough to show that these sets have the same size. But we have seen that card $(B_{ni}) \leqslant$ card(B_{n0}) for each i and if any of these inequalities were strict we would have the contradiction

$$\binom{2n}{n} = \text{card}(B_n) = \sum_{i=0}^{n} \text{card}(B_{ni})$$
$$< (n+1)\text{card}(B_{n0}) = \binom{2n}{n}. \qquad \square$$

Lemma 6. *If $w \in B_n$, $\Phi_n(w)$ can be determined in $O(n)$ time.*

Proof. We follow the inductive definition of $\Phi_n(w)$. Let $T(n)$ be the total number of operations required. The decomposition $w = uv$, where u is irreducible, can be found in $O(r)$ steps, where $r = |u|$ by partial summation; the first partial sum equal to zero defines u. Then $\Phi_{n-r}(v)$ must be calculated and so we obtain the recurrence $T(n) = O(r) + T(n-r) = O(n)$. $\square$

Note that the computation of $\Phi_n(w)$ requires no integers larger than $2n$. Thus step 3 of our algorithm can be implemented in linear time, with integers of size $O(n)$, by applying the function Φ_n.

The above discussion provides the proof of the following theorem:

Theorem 7. *Algorithm RANDOM BRACKET SEQUENCE is an unbiased random binary tree generator. It is executes in linear time and uses integers of size $O(n)$.*

Acknowledgment

We acknowledge helpful discussions with D. Horrocks, A. Knight and U. Martin.

References

[1] D.B. Arnold and M.R. Sleep, Uniform random number generation of n balanced parenthesis strings, *ACM Trans. Programming Languages Systems* **2** (1980) 122–128.

[2] K.L. Chung and W. Feller, Fluctuations in coin-tossing, *Proc. Nat. Acad. Sci. U.S.A.* **35** (1949) 605–608.

[3] W. Feller, *An Introduction to Probability Theory and its Applications*, Vol. 1 (Wiley, New York, 3rd ed., 1968).

[4] G.D. Knott, A numbering system for binary trees, *Comm. ACM* **20** (1977) 113–115.

[5] D.E. Knuth, *Semi-numerical Algorithms, The Art of Computer Programming*, Vol. 2 (Addision-Wesley, Reading, MA, 2nd ed., 1981).

[6] H.W. Martin and B.J. Orr, A random binary tree generator, in: *Computing Trends in the 1990's, ACM Seventeenth Computer Science Conf.*, Louisville, KY (ACM, New York, 1989) 33–38.

[7] E.M. Reingold, J. Nievergelt and N. Deo, *Combinatorial Algorithms: Theory and Practice* (Prentice Hall, Englewood Cliffs, NJ, 1977).

[8] M. Solomon and R.A. Finkel, A note on enumerating binary trees, *J. ACM* **27** (1980) 3–5.