Learning objectives:
- ☐ simplify summations into nicer expressions using the perturbation method,
- ☐ analyze how many operations are performed in some simple algorithms.

1 Methods for computing sums

Consider the following example.

Example 1:
Say I told you that I will give you either a million dollars today, or \$50,000 every year for the rest of your life. Which would you pick? You may assume a fixed annual interest rate of $p = 8\%$.

Solution:
We can compare the two options by calculating the *present value* of both options - how much is each option *currently* worth? Clearly, the one million dollar option is worth one million dollars today. But how much is \$50,000 every year for the rest of your life worth today? Well, \$50,000 in 1 year is worth $\$50,000/(1+p)$ today. Similarly, \$50,000 in two years is worth $\$50,000/(1+p)^2$ today. The total value today is the sum of the current value of these yearly \$50,000 payments after n years is:

$$\text{present value} = \$50,000 + \frac{\$50,000}{1+p} + \frac{\$50,000}{(1+p)^2} + \cdots = \sum_{i=0}^{n} \frac{\$50,000}{(1+p)^i}.$$

Okay, well to completely solve this, we need a way to compute this sum.

1.1 Perturbation method

The last example involved a sum of the form

$$\sum_{i=0}^{n} r^i = 1 + r + r^2 + r^3 + \cdots + r^n \tag{1}$$

which is called a *geometric series*. Convince yourself that our formula for the present value of an annuity is in this form with $r = \frac{1}{1+p}$. Note that since $p > 0$, then $r < 1$.

To compute Equation 1 we can use the *perturbation method*, which involves "perturbing" it and seeing if we can observe any interesting structure. Let's perturb Equation 1 by multiplying it by r and subtracting the result from Equation 1:

$$\begin{aligned} S &= 1 + r + r^2 + r^3 + \cdots + r^n \\ -rS &= \quad - r - r^2 - r^3 - \cdots - r^n - r^{n+1}. \end{aligned}$$

When doing the subtraction, a lot of terms cancel, and we are left with

$$S - rS = 1 - r^{n+1} \quad \rightarrow \quad S = \boxed{\sum_{i=0}^{n} r^i = \frac{1 - r^{n+1}}{1 - r}}. \tag{2}$$

Say you live to $n = 100$ years and the interest rate (which you lock in at year 0) is $p = 0.08$. Then, the present value of the annuity (by plugging into Equation 2) is

Take the one million dollars!

$$\text{present value} = \$50,000 \times \frac{1 - \left(\frac{1}{1.08}\right)^{101}}{\left(1 - \frac{1}{1.08}\right)} \approx \$674,693.$$

Now, what happens as $n \to \infty$ (i.e. you live forever)? Under the assumption that $|r| < 1$ and taking the limit as $n \to \infty$ in Equation 2 gives

$$\lim_{n \to \infty} \frac{1 - r^{n+1}}{1 - r} = \frac{1}{1 - r}. \tag{3}$$

$|r| < 1$**?**

This assumption is very important because it means that $r^{n+1} \to 0$ as $n \to \infty$.

1.2 Derivative method (optional)

This section can be skipped! I really just want to provide you with another method for computing sums in case you're interested, however, I will not ask you to compute any sums that require this method. If you are interested, consider the following sum:

$$\sum_{i=0}^{n} ir^i = r + 2r^2 + 3r^3 + \cdots + nr^n. \tag{4}$$

This looks a lot like a geometric series, but the factors in front of each terms messes everything up. However, if you take the derivative of Equation 2 *with respect to r*. Then you can recover something that looks like Equation 4:

$$\frac{\mathrm{d}}{\mathrm{d}r} \sum_{i=0}^{n} r^i = \frac{\mathrm{d}}{\mathrm{d}r}(1 + r + r^2 + r^3 + \cdots + r^n) = 1 + 2r + 3r^2 + \cdots + nr^{n-1} = \frac{\mathrm{d}}{\mathrm{d}r}\left(\frac{1 - r^{n+1}}{1 - r}\right) = \frac{1 - (n+1)r^n + nr^{n+1}}{(1 - r)^2}.$$

We can multiply this result by r to recover the sum in Equation 4:

$$\sum_{i=0}^{n} ir^i = r + 2r^2 + 3r^3 + \cdots + nr^n = r\left(\frac{1 - (n+1)r^n + nr^{n+1}}{(1 - r)^2}\right).$$

2 Double sums

Computing sums is useful if we want to count how many operations are performed in an algorithm, which is particularly important if you are estimating the run-time of your algorithm. Consider the following algorithm for calculating the total number of edges in a graph $G = (V, E)$ from the adjacency matrix A.

countEdges

```
   input: adjacency matrix $A$ for graph $G = (V, E)$
   output: number of edges $|E|$
1  $s \leftarrow 0$
2  for $i = 1 \to |V|$
3      for $j = 1 \to i$
4          $s \leftarrow s + a_{i,j}$
5  return $s$
```

Algorithm 1: Algorithm for calculating the number of edges in a graph from its adjacency matrix.

Example 2:
How many operations are performed in Algorithm 1?

Solution:
We have a single operation on Line 1 when we initialize s. We also have a single addition being performed on Line 4 so we need to count how many times this line gets hit. The outer loop gets executed $|V|$ times, but the inner loop gets executed i times. Expanding this gives

$$\text{number of operations} = 1 + \sum_{i=1}^{|V|} \sum_{j=1}^{i} 1.$$

Note that the inner sum is just $i \times 1$ since the thing we are summing doesn't depend on j. We are therefore computing

$$\sum_{i=1}^{|V|} i = 1 + 2 + \cdots + |V| = \frac{|V|(|V|+1)}{2}.$$

The total number of operations (including the one on Line 1) is then $1 + \frac{|V|(|V|+1)}{2}$.

Now if $|V|$ is really big, then we don't really care about the $+1$'s or the $/2$. We care more about the $|V|^2$ part, which will lead into our discussion about *asymptotic* notation (soon).

Arithmetic series?

See if you can get this result using the perturbation method!

3 Matrix multiplication

In your electives (and in real life), you will very often need to write code that manipulates matrices. Though it is very unlikely that you will need to write your own matrix-multiplication function (because somebody else likely wrote a very optimized version), you should still be able to analyze the run-time complexity of the following algorithms.

When will I need this?

You will often need handle matrices in courses such as Computer Graphics, Machine Learning, Computer Vision and many many more! Pretty much any time you have a set of equations and unknowns, or need to transform vectors.

Given a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ (that is, a real $m \times n$ matrix), and a vector $\vec{x} \in \mathbb{R}^n$ (that is, an n-dimensional vector), then we can compute the m-dimensional vector $\vec{b} = \mathbf{A}\vec{x}$ by multiplying every row of A with $\vec{x}$ to form the entries of $\vec{b}$. This procedure is outlined in Algorithm 2.

Algorithm 2: Matrix-vector multiplication.

```
matrixVectorMultiplication

    input: $m \times n$ matrix $\mathbf{A}$ and $n$-dimensional vector $\vec{x}$
    output: $m$-dimensional vector $\vec{b} = \mathbf{A}\vec{x}$.
1   $\vec{b} \leftarrow \vec{0}$
2   for $i = 1 \rightarrow m$
3       for $j = 1 \rightarrow n$
4           $b_i = b_i + a_{i,j} \times x_j$
5   return $\vec{b}$
```

Example 3:
How many operations are performed in the matrix-vector multiplication procedure of Algorithm 2?

Solution:
First, Line 1 incurs m operators to assign zero to every element of $\vec{b}$. Next, there are two loops to consider, one ranging from 1 to m, the other ranging from 1 to n. There are two operations (one addition and one multiplication) being performed on Line 4 in Algorithm 2, which gives $2mn + m$ operations in total.

In reality, you probably don't care about the 2 in front of the mn, nor do you care about the extra m from initializing $\vec{b}$. It's more important to consider the fact that we have two loops, which gives *roughly* mn operations. We will consider this type of *asymptotic* analysis soon.