Analyzing the two options: how much is each option worth today?

option 1: $1,000,000 today

assume interest rate of $p\%$ $\quad p>0$

option 2: year 0 | year 1 | year 2: | year 3: | ..... | year n:

$$\text{present value} = 50K + \frac{50K}{(1+p)} + \frac{50K}{(1+p)^2} + \frac{50K}{(1+p)^3} + \dots + \frac{50K}{(1+p)^n}$$

$$= 50K\left[1 + \frac{1}{(1+p)} + \frac{1}{(1+p)^2} + \dots + \frac{1}{(1+p)^n}\right]$$

let $r = \frac{1}{1+p}$

$$= 50K\left[1 + r + r^2 + r^3 + \dots + r^n\right]$$

geometric series. $\quad r<1$

upper bound

$$= 50K \sum_{i=0}^{n} r^i$$

summation

lower bound

Analyzing the two options: how much is each option worth today?

$$\sum_{i=0}^{n} r^i = S = 1 + r + r^2 + \dots + r^{n-1} + r^n$$

$$-rS = -r - r^2 - r^3 + \dots - r^n - r^{n+1}$$

multiply by $-r$ then add two equations

$$S - rS = 1 - r^{n+1}$$

$$\boxed{S = \frac{1-r^{n+1}}{1-r}} = \sum_{i=0}^{n} r^i$$

take $1M!

let $p = 8\%$

$n = 100$

option 2: $50k\left[\frac{1-r^{101}}{1-r}\right] \simeq 675K$

$r = \frac{1}{1+0.08} = \frac{1}{1.08}$

like infinity

what if n is huge?

$r^{n+1} \to 0$

we have $\frac{1}{1-r}$

Counting the number of operations in matrix multiplication.

In general, we care about arithmetic (+, -, *, /) and comparison operators (>, <).

matrix $m \times n$

vector n

```
def multiply_matrix_vector(
        A: list[list[float]], x: list[float]) -> list[float]:
    """Computes the matrix-vector multiplication A * x = b"""
    m = len(A) # number of rows, also num. entries in b
    n = len(x) # number of colums of A, also num. entries in x
    b = [0 for _ in range(m)] # initialize b to zero
    for i in range(m):
        assert len(A[i]) == n
        for j in range(n):
            b[i] = b[i] + A[i][j] * x[j]
    return b
```

a

→ 2 operations

$\underbrace{\sum_{j=1}^{n} 2}_{a}$

$\sum_{i=1}^{m} a$

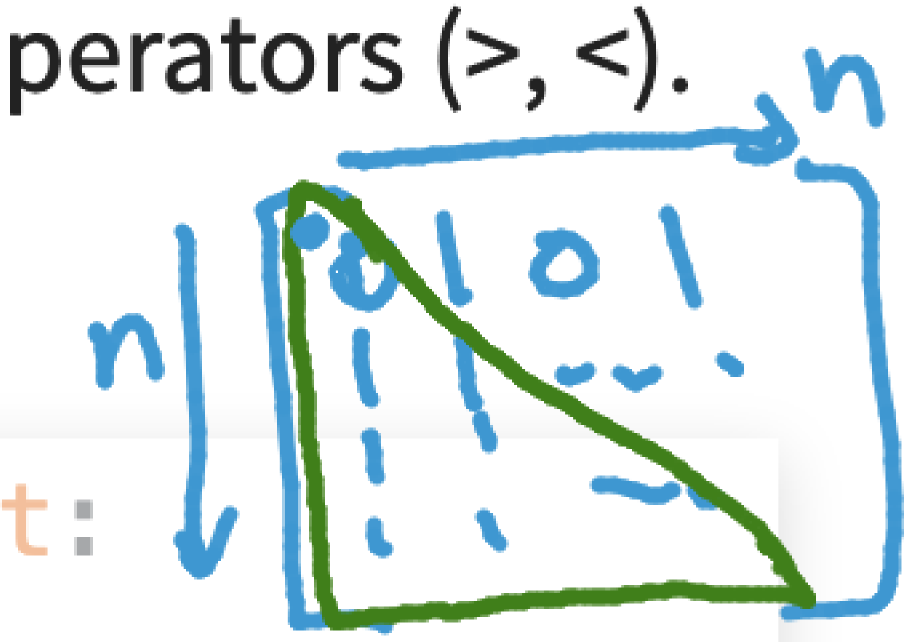
$$\#\text{operations} = \sum_{i=1}^{m} \underbrace{\sum_{j=1}^{n} (2)}_{\text{adding 2 n times}} = \sum_{i=1}^{m} 2n = 2n \sum_{i=1}^{m} 1$$

$$\boxed{= 2mn}$$

when m, n get very big, we might not care about "2"

Counting the number of operations in a graph algorithm.

In general, we care about arithmetic (+, -, *, /) and comparison operators (>, <).



```python
def count_edges(adj_matrix: list[list[int]]) -> int:
  """Counts the number of edges in a graph (V, E)
     given it's adjacency matrix."""
  n = len(adj_matrix) # this is |V|
  s = 0
  for i in range(n):
      for j in range(i):
          s = s + adj_matrix[i][j]
  return s
```

$i=0$ 1

$i=1$ 2

$i=3$ 3

$\sum_{j=1}^{i} 1$

arithmetic series

#operations $\sum_{i=1}^{n} \underbrace{\sum_{j=1}^{i} 1}_{\underbrace{1+1+\dots+1}_{i \text{ times}}}$

$S = \boxed{\sum_{i=1}^{n} i}$

$= 1 + 2 + 3 + \dots + (n-1) + n$

$\quad n + (n-1) + (n-2) + \dots + 2 + 1$

$(n+1) + (n+1) + (n+1) + \dots + (n+1) + (n+1)$

$= n(n+1) = 2S \qquad \boxed{S = \frac{n(n+1)}{2}}$

Exercise 1:

hint#2

Compute a closed-form expression for: $\sum_{i=1}^{n}\sum_{j=1}^{n}(i+j)$ $= \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n} i}_{1} + \underbrace{\sum_{i=1}^{n}\sum_{j=1}^{n} j}_{2}$

Hint: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$.

$$① \quad \sum_{i=1}^{n}\sum_{j=1}^{n} i = \sum_{i=1}^{n}(\underbrace{i+i+i+i+\cdots+i}_{n \text{ times}}) = \sum_{i=1}^{n} n\cdot i = n\sum_{i=1}^{n} i = \frac{n^2(n+1)}{2}$$

$$② \quad \sum_{i=1}^{n}\sum_{j=1}^{n} j = \sum_{i=1}^{n}\frac{n(n+1)}{2} = \frac{n(n+1)}{2}\underbrace{\sum_{i=1}^{n} 1}_{n} = \frac{n^2(n+1)}{2}$$

$$① + ② = n^2(n+1)$$