# Polynomial evaluation

Say $f(x) = \sum_{i=0}^{d} a_i x^i = \underline{a_0 + a_1 x + \cdots a_d x^d}$

Goal: write a function that accepts $f, x$ as input & returns $f(x)$.

Example: Say $\boxed{f(x) = x^2 - 1}$. $(= \underline{-1 \cdot x^0 + 0x + 1 \cdot x^2})$

Then $a_0 = -1$, $a_1 = 0$, $a_2 = 1$

$-1x^0 \quad + \quad 0x \quad + 1x^2$

input: $f$ (as coeff list), $x = \_$

output $f(x)$.

```
int polyEval(const vector<int>& a, int x)
{
    int sum = 0;
    for (int i = 0; i < a.size(); i++) {   <--
        sum += a[i] * pow(x,i);   <--
    }
    return sum;
}
```

$\underbrace{\texttt{const vector<int>\& a}}_{\equiv f}$, $\underbrace{\texttt{int x}}_{\equiv x\text{, the eval. pt.}}$

pow: from <cmath>

This works, but it's kind of slow. What is the cost of naively computing pow(x,i)?

Might take $\approx$ i multiplications! (There are more clever ways though...)

Then what's the total # of mult's for polyEval?

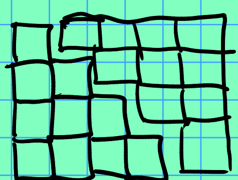
```
for (i=0; i<a.size(); i++) {
    sum += a[i] * pow(x,i);
}
```

$1 + i$ mults.

total: $1 + 2 + 3 + \cdots + d+1$ = degree (f).

$$\frac{(d+1)(d+2)}{2} \approx d^2$$

1+2+3+4 = area →



$$\frac{d(d+1)}{2}$$

area of rectangle $= 2 *$ answer $= d(d+1)$

So answer $= \dfrac{d(d+1)}{2}$

Question: can we do better than $d^2$ mults?

Yes:

```
int polyEval(a, x)
{
    int sum = 0;   // sum_{k=0}^{i} a_k x^i
    int xi = 1;    // store x^i
    for (i=0; i<a.size(); i++) {
        sum += a[i] * xi;
        xi *= x;
    }
    return sum;
}
```

How many mults? if deg (f) = d, we need $2(d+1)$

Can we do better than $2(d+1)$?

Again, yes! Try this:

$a_d$

$a_d x + a_{d-1}$

$(a_d x + a_{d-1})x + a_{d-2}$

$((a_d x + a_{d-1})x + a_{d-2})x + a_{d-3}$

$\vdots$

$= f(x)$

Note! uses only $\underline{1}$ mult / step!

(This is called Horner's Rule, btw.)