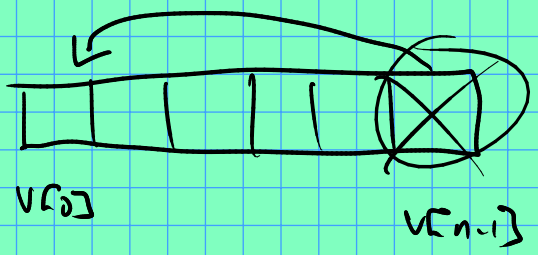
Push_front + pop_front ?

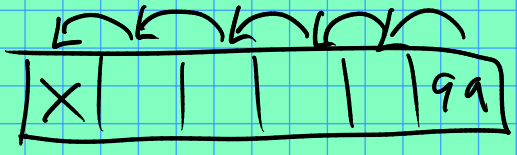
If order doesn't matter:



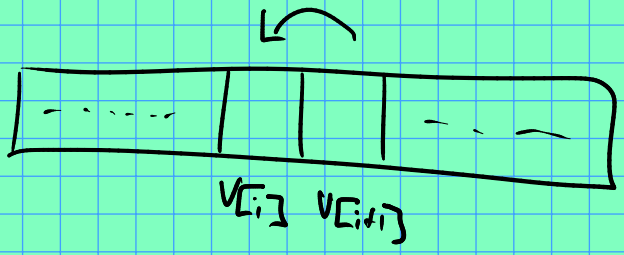
(n = v.size())

```
v[0] = v[ v.size()-1 ];
v.pop_back();
```

But if we do care about order, this is annoying:

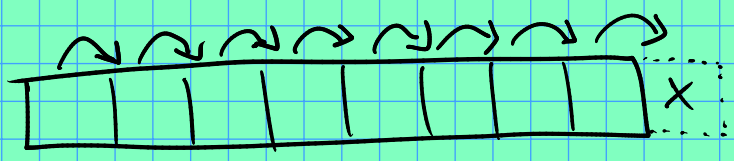



```
for(i=0; i < v.size()-1; i++) {
    v[i] = v[i+1];
}
v.pop_back();
```



Note: this takes $\approx$ n steps, where n = v.size() !

what about push_front ?




```
V.resize(V.size()+1);
for(i=1; i<V.size(); i++)
    V[i] = V[i-1];

V[0] = x;
```

```
void push_front(vector<int>& V, int x);
```

Note: passing vectors by value (no &) leads to a copy of the whole thing being created! You can "fake" by value efficiently like this:

```
void f(const vector<int>& V);
```

const: no modifying!

&: No copying! (efficient!)

Example: polynomial evaluation.

input, $f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n$ $\left(= \sum_{i=0}^{n} a_i x^i\right)$

and a value x_0.

output: $f(x_0)$

How to represent f? List of coefficients will suffice.

Prototype: `int poly_eval(const vector<int>& a, int x);`

// a[i] ≡ a_i above.

Sketch:

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

Note: calling pow(x,i) over + over seems wasteful: if I know x^i, x^{i+1} is easy to compute! $x^{i+1} = x^i \cdot x$.

Exercise: try to finish this!