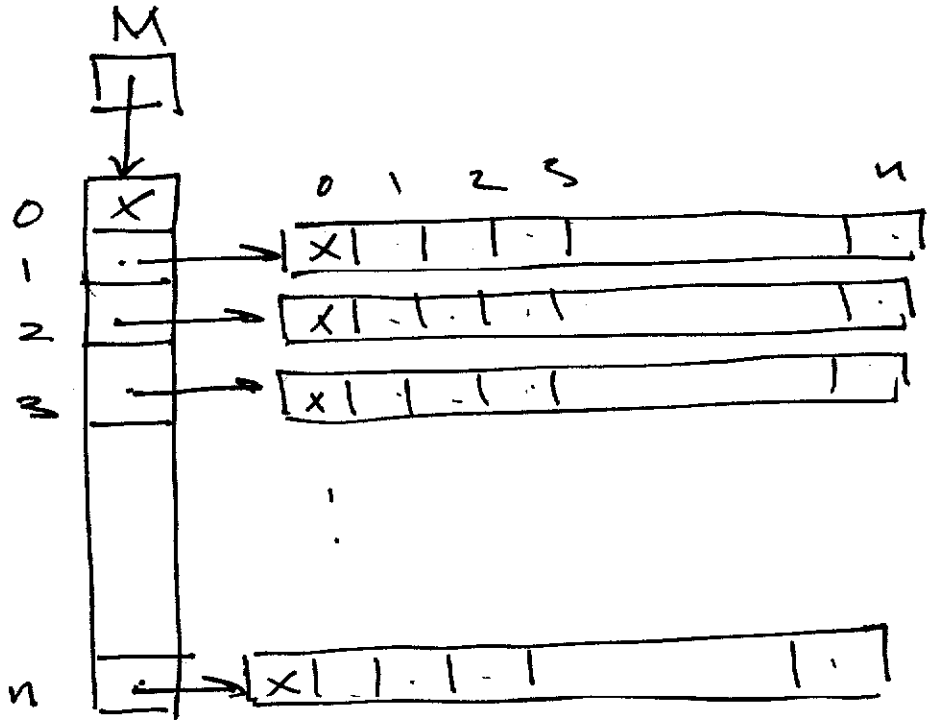
CSR 101 1-27-22

- Pa3 ext. 1 last day

Pa4:

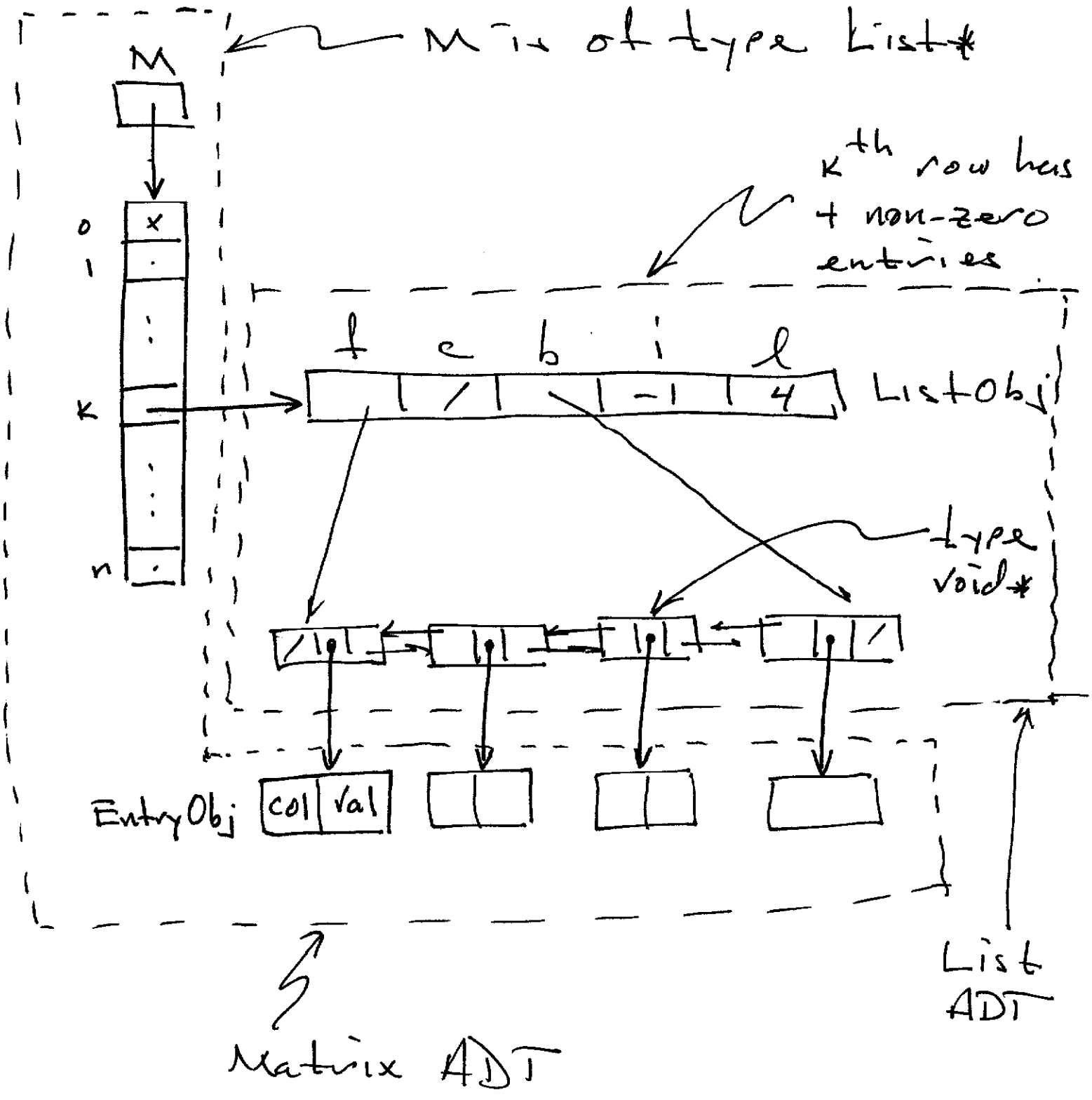
traditional matrix representation:



M would have type double**

We don't use this Matrix representation.

In Pa4 we use following sparse matrix representation



Algorithm Runtime & Efficiency

To measure runtime of an (iterative) algorithm, first establish some basic operation(s), and count # of times these ops. are performed.

Ex.

```
op1
for i = 1 to 10
    op2
    for j = 1 to 10
        op3
        for k = 1 to 10
            op4
```

op1	1
op2	10
op3	100
op4	1000

} # of executions

← count only this one

Assume OP1 .. OP4 are approx. equal cost.

(1) what if they're not equal?
(2) why not count all?

(1) re-write algorithm in terms of (1) basic OP.

new algorithm:

```
OP
for i = 1 to 10
    OP
    for j = 1 to 10
        OP
        for k = 1 to 10
            OP
```

more common to have

```
OP
for i = 1 to n
    OP
    for j = 1 to n
        OP
        for k = 1 to n
            OP
```

(# of OPs as a fcn. of input n)

$= T(n) = n^3 + n^2 + n + 1$

we propose to replace this by n^2,
i.e. only count innermost loop.

①

```
for i = 1 to n
    for j = 1 to n
        for k = 1 to n
            OP
```

Question we're interested in is how runtime grows with input size.

compare using limits:

$$\lim_{n\to\infty}\left(\frac{T(n)}{n^3}\right) = \lim_{n\to\infty}\frac{(n^3+n^2+n+1)}{n^3}$$

$$= \lim_{n\to\infty}\left(1+\frac{1}{n}+\frac{1}{n^2}+\frac{1}{n^3}\right)$$

$\frac{1}{n} \to 0$, $\frac{1}{n^2} \to 0$, $\frac{1}{n^3} \to 0$

$$= \boxed{1}$$

so we can ignore lower order terms

Consider another algorithm

(2)

```
for i = 1 to n
    for j = 1 to n
        for k = 1 to n
            OP
            OP
```

its runtime is: $2n^3$

Assume (1) and (2) accomplish the same thing, so we can compare them.

compare: n^3 to $2n^3$

n^3: twice as fast

we consider these equivalent!!

why?
we can equalize them by running (2) on a machine that is twice as fast.

It not always possible to do this. consider

(3)

```
for i = 1 to n
    for j = 1 to n
        for k = 1 to 10
            OP
```

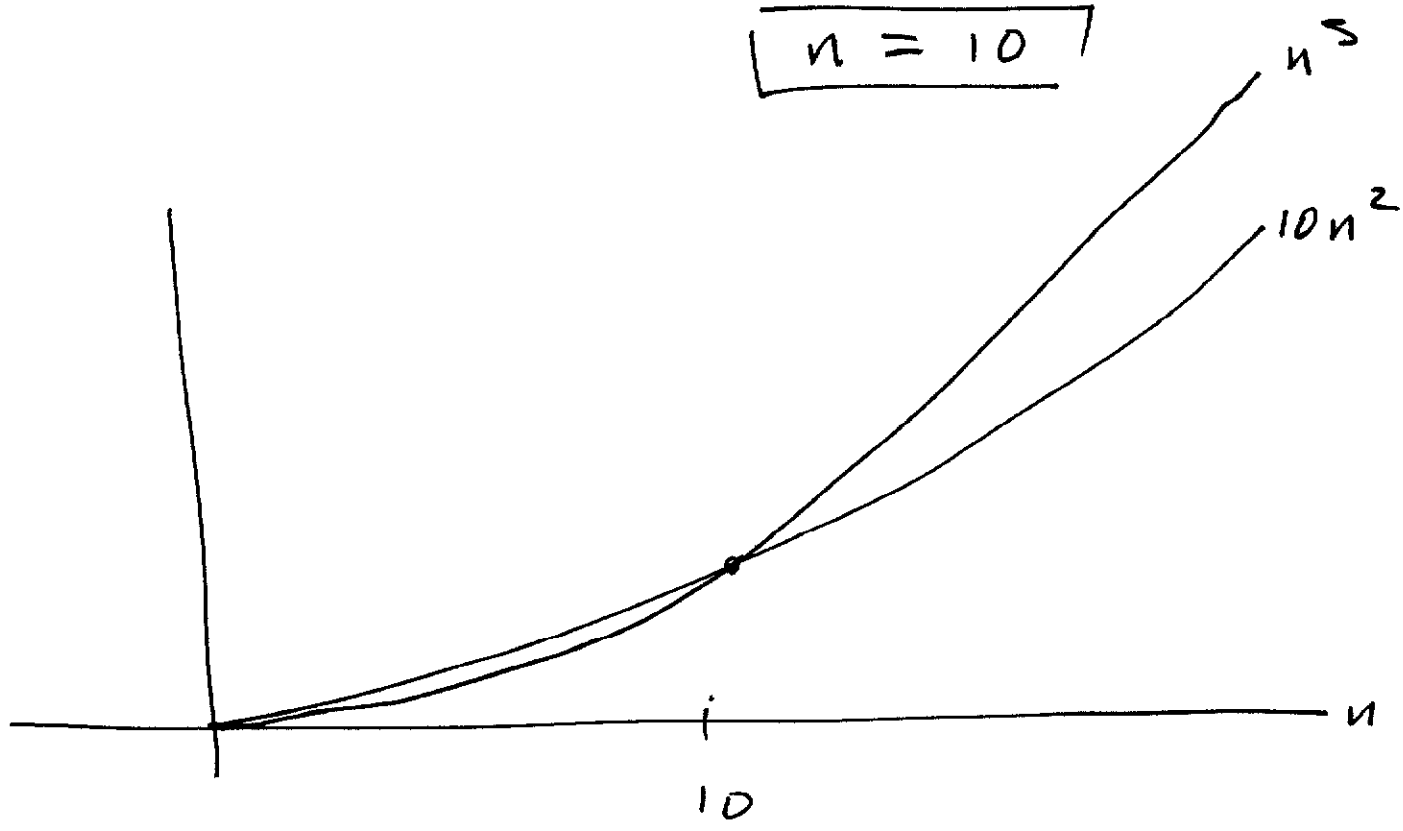
runtime = $10n^2$

Compare (1) to (3)

n^3 to $10n^2$

Crossover Point: $n^3 = 10n^2$

$n = 10$



in the limit $n \to \infty$, $10n^2$ is smaller:

$$\frac{10n^2}{n^3} = \frac{10}{n} \longrightarrow 0$$

so (B) is a superior algorithm.

Informal method :

(1) choose a basic op. inside the body of innermost loop.

(2) count # of executions of op as a function of input size n : $f(n)$

(3) simplify fcn $f(n)$ by
- drop lower order terms
- replace coeff. of highest order term by 1.

Result : asymptotic runtime

Asymptotic growth of functions

Let $f(n)$, $g(n)$ be functions.

Defn

write $\boxed{f(n) = O(g(n))}$ to mean

there exists $R > 0$ and $n_0 > 0$

such that $\frac{f(n)}{g(n)} \leq R$ for all $n \geq n_0$

We say

- $g(n)$ is an *asymptotic upper bound* for $f(n)$
- $f(n)$ is *asymptotically bounded above* by $g(n)$

means: $\boxed{f(n) \text{ grows no faster than } g(n).}$