C2 cont.

1 Example of decision sort tree for Bubble Sort for an array of size 3 : v1, v2, v3 : https://condor.depaul.edu/glancast/403class/docs/slides/Nov14/slide4.html

In the example symbol ":" is used instead of "?" .
Also, there is a difference of notation (in the course examples only indices are used, in this example the concrete values are used; e.g., 1 vs v1)

2. Turing/ Cook Reduction

A problem $P$ polynomially reduces to (a solvable problem) $Q$, write $P \propto Q$, if we can design an algorithm for $P$ as follows:

1. let $p$ be an instance of $P$;
2. preprocesss in polynomial time the input $p$ to obtain an instance (or instances) of $Q$;
3. call an algorithm for $Q$, possible of several times (but of polynomial times);
4. postprocess the output(s) given by $Q$ in polynomial time to get the answer $P(p)$.

If the (pre+post)processing time requires $O(g(n))$ time, then we write $P \propto_{g(n)} Q$.

$$p \xrightarrow[\text{poli.}]{\text{preprocess}} \mathcal{L}$$

$$\downarrow \text{ call alg for } Q \text{ (a poli nb. of times)}$$

$$P(p) \xleftarrow[\text{poli}]{\text{postprocess}} \mathcal{L}'$$

## Example: MAX $\propto$ SORT

FieLet MAX be the following problem:

*Input* A set $S$ totally ordered.
*Output* The bigest element in $S$.

The following algorithm solves MAX:

1. represent $S$ with an array $s$ (preprocessing);
2. call a sorting algorithm for $s$;
3. return the last element in $s$ (postprocessing);

So, we have MAX $\propto$ SORT!?
$\propto$ does not necessarily mean "the reduction from a complex problem to an easier one"!!!
$\propto$ is rather a "transformation" ...

$$p = \{x_1, \dots, x_m\} \xrightarrow{\text{preprocess}} \mathcal{L} = [x_1, \dots, x_m]$$

$$\downarrow \text{ SORT}$$

$$p' = x_1' \xleftarrow{\text{postprocess}} \mathcal{L}' = \text{sorted array} = [x_1', \dots, x_m']$$

3. Karp Reduction

## Karp reduction

Let $P$ and $Q$ be decision problems.
The problem $P$ polynomially reduces to (the solvable) $Q$, write $P \propto Q$, if we can design an algorithm that solves $P$ as follows:

1. let $p$ be an instance of $P$;
2. preprocess the input $p$ in polynomial time to get an instance $q$ of $Q$;
3. call (once) an algorithm solving $Q$;
4. the answer for $Q$ for $q$ is the same with the answer of $P$ for $p$ (no post processing).

If the preprocessing time is $O(g(n))$, then we write $P \propto_{g(n)} Q$.

The Karp reduction is a particular case of the Turing/Cook reduction.

O(f(n)) = margine superioara a complexitatii unui algoritm

In algoritmul gcd prin scaderi repetate se executa max(a,b) pasi, deci complexitatea algoritmului este O(max(a,b)), unde a si b sunt numerele pentru care se calculeaza gcd-ul. Acest caz il intalnim de exemplu pentru a=n si b=1.

Omega(f(n)) = margine inferioara a complexitatii unui algoritm

In algoritmul de sortare prin inserare se executa minimum n pasi (n = dimensiunea array-ului de sortat). Acest caz il intalnim de exemplu daca arrayul este deja sortat.

If the (pre+post)processing time requires $O(g(n))$ time, then we write $P \propto_{g(n)} Q$.

## Reduction: properties

**Theorem**

a) If $P$ has the time complexity $\Omega(f(n))$ and $P \propto_{g(n)} Q$ (Karp version) the $Q$ has the time complexity $\Omega(f(n) - g(n))$.

b) If $Q$ has the time complexity $O(f(n))$ and $P \propto_{g(n)} Q$ (Karp version) then $P$ has the time complexity $O(f(n) + g(n))$.

$$p \xrightarrow[O(g(n))]{\text{preprocess}} \mathcal{L}$$

$$\downarrow \text{ call alg for } Q$$

$$P(p) \xleftarrow[O(g(n))]{\text{postprocess}} \mathcal{L}'$$