Problem reductions: reducing problem A to problem B

Denoted as A $\rightarrow$ B

Also denoted A $\leq$ B

Let *a* be an instance of problem A

Let R be a function that maps instances of A to instances of B

That is, R(*a*) = *b*, such that *b* is an instance of B

$$
\begin{array}{ccc}
 & R & \\
a & \rightarrow & b \\
\downarrow f & & \downarrow g \\
 & \leftrightarrow & \\
\end{array}
$$

f(*a*) = g(*b*)

If f is an NP decision algorithm for A and g is an NP decision algorithm for B, then <u>R is a valid reduction</u> iff both give the same yes/no answer, i.e., f(*a*) = g(*b*).

That is, if there exists a solution to instance *a* of A then there must be a solution to instance *b* of B, and if there is no solution to instance *a* of A then there must be no solution to instance *b* of B. Otherwise, R is not a valid reduction.

R is a valid reduction of problem A to B when R runs in polynomial time and R(*a*)= *b* iff f(*a*)=g(*b*) where f and g are NP decision algorithms for A and B respectively.

Since R is easy (runs in polynomial time), if g(b) is easy, then f(a) must be easy since f(a)=g(R(a)) and both g and R are easy.

Reduce MST $\rightarrow$ SS
Assume MST returns the edges in the solution: T $\leftarrow$ MST(G)

MST: Does graph G have a spanning tree with total weight at most MAX?

SS: Given a triple (S, *min*, *max*), where S is a set of positive integers and *max* and *min* are positive integers. Is there a subset T of S such that the sum of the integers in T is at most *max* and at least *min*?

(G, MAX) -> (S, min, max)

Algorithm verifyMST(G, MAX, T)

```
    T ← MST(G)  // O(m log n)
    sum <- 0
    for each e in T.elements() do
        sum ← sum + weight(e)
    if sum>MAX then
        return no
    return yes
```

```
Algorithm reduceMST2SS(G, MAX)
    R ← new Sequence
    R.insertLast(2)
    if verifyMST(G,MAX) = no then
        return (R, 1, 1)
    return (R, 2, 2)
```