

Problem type 1:

Let's say we have the following two problems:

(See variants below)

As you can see, they are both the same problem but one is a decision version of the problem and the other is a optimization version of the same problem. We have a block-box algorithm that solves the decision version in polynomial time. Using this black box program, describe an algorithm that solves the optimization version of this problem.

Does this algorithm demonstrate $\text{ProblemOPT} \implies \text{ProblemDEC}$, or $\text{ProblemDEC} \implies \text{ProblemOPT?}$ (select one and draw a box around it on your test sheet).

a. BYB/BYE

Independent Set Decision: ($\text{IndSetDec}(G, k)$)

- INPUT: A undirected graph G and integer k
- OUTPUT: True if G has a independent set of size $\geq k$, False otherwise

Independent Set Optimization: ($\text{IndSetOpt}(G, k)$)

- INPUT: A undirected graph G
- OUTPUT: The size of the largest independent set in G

Solution: Simply iterate on k from n down to 1.

```
IndSetOpt(G)
  for k = |V| to 1
    if IndSetDec(G, k)
      return k
```

This reduction shows $\text{INDSETOPT} \implies \text{INDSETDEC}$



b. BYA/BYH

Clique Decision: ($\text{CliqueDec}(G, k)$)

- INPUT: A undirected graph G and integer k
- OUTPUT: True if G has a clique of size $\geq k$, False otherwise

Clique Optimization: ($\text{CliqueOpt}(G, k)$)

- INPUT: A undirected graph G
- OUTPUT: The size of the largest clique in G

Solution: Simply iterate on k from n down to 1.

```
CliqueOpt(G)
  for k = |V| to 1
    if CliqueDec(G, k)
      return k
```

This reduction shows $\text{CLIQUEOPT} \implies \text{CLIQUEDEC}$



c. BYC/BYF

Traveling Salesman Decision: ($\text{TSDec}(G, k)$)

- INPUT: A undirected weighted, all-positive graph G and integer k
- OUTPUT: True if there exists a path that visits every vertex exactly once, ends at the vertex it started at and costs $\leq k$. False otherwise

Traveling Salesman Optimization: ($\text{TSOpt}(G, k)$)

- INPUT: A undirected weighted, all-positive graph G
- OUTPUT: The *weight* of the minimum cycle in G that visits every vertex exactly once. (-1 if one doesn't exist)

Solution: Simply iterate on all possible weights of the TSP tour. The max possible tour weight would be approximately the maximum edge weight multiplied by $n - 1$ (since the tour must have n edges).

```
TSOpt(G)
  for  $k = 0$  to  $|V| - 1 \times \ell(E)$ 
    if  $\text{TSDec}(G, k)$ 
      return  $k$ 
  return -1
```

This reduction shows $\text{TSOPT} \implies \text{TSDEC}$. ■

d. BYD/BYG

kColor Decision: ($\text{kColorDec}(G, k)$)

- INPUT: A undirected graph G and integer k
- OUTPUT: True if the vertices in G can be colored with k colors such that no two adjacent vertices share the same color, False otherwise

kColor Optimization: ($\text{kColorOpt}(G, k)$)

- INPUT: A undirected graph G
- OUTPUT: The *minimum* number of colors needed to color the vertices in G such that no two adjacent vertices share the same color.

Solution: We can assume that the maximum number of colors we'd need is n . So, simply iterate on k from n down to 1.

```
kColorOpt(G)
  for  $k = 1$  to  $|V|$ 
    if  $\text{kColorDec}(G, k)$ 
      return  $k$ 
```

This reduction shows $\text{KCOLOROPT} \implies \text{KCOLORDEC}$. ■