Day 15

PROBLEM REDUCTION

LINEAR PROGRAMMING

Problem Reduction for Algorithm Design

Transform-and-Conquer idea: solve problem A by transforming it into different problem B for which an algorithm is already available.

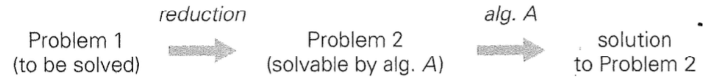


FIGURE 6.15 Problem reduction strategy.

The runtime is determined by:

- the time of the reduction A → B
- the time of solving B

Problem Reduction Examples

computing LCM → computing GCD

lcm(*m* , *n*)

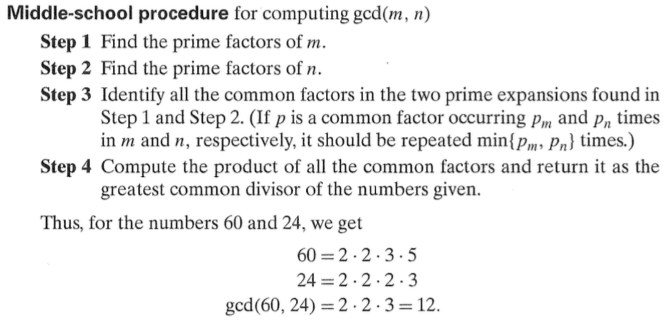
Example: lcm(24, 60) = 120.

Hard way: Given prime factorization of *m* and *n*, compute the product of all common prime factors of *m* and *n*, all of the prime factors of *m* that are not in *n* and all the prime factors of *n* that are not in *m*:

- 24 = 2 * 2 * 2 * 3
- 60 = 2 * 2 * 3 * 5
- lcm(24, 60) = (2 * 2 * 3) * (2) * (5)

Problem Reduction Examples

lcm(*m* , *n*) requires a list of consecutive primes. Like:



Middle-school procedure for computing gcd(m, n)

Step 1 Find the prime factors of m.

Step 2 Find the prime factors of n.

Step 3 Identify all the common factors in the two prime expansions found in Step 1 and Step 2. (If p is a common factor occurring p_m and p_n times in m and n, respectively, it should be repeated $\min\{p_m, p_n\}$ times.)

Step 4 Compute the product of all the common factors and return it as the greatest common divisor of the numbers given.

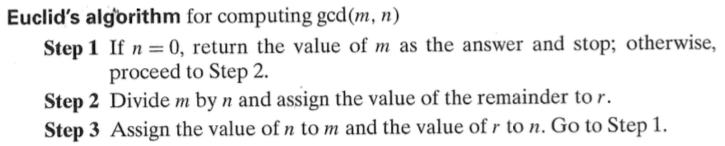
Thus, for the numbers 60 and 24, we get

$$60 = 2 \cdot 2 \cdot 3 \cdot 5$$
$$24 = 2 \cdot 2 \cdot 2 \cdot 3$$
$$\gcd(60, 24) = 2 \cdot 2 \cdot 3 = 12.$$

Problem Reduction Examples

Instead, calculate lcm(m,n) as follows: (m*n)/gcd(m, n)

Use



Euclid's algorithm for computing gcd(m, n)

```
Step 1  If $n = 0$, return the value of $m$ as the answer and stop; otherwise, proceed to Step 2.
Step 2  Divide $m$ by $n$ and assign the value of the remainder to $r$.
Step 3  Assign the value of $n$ to $m$ and the value of $r$ to $n$. Go to Step 1.
```

Preview: Reductions for Complexity Theory

$$A \xrightarrow{\text{efficient mapping}} B$$

Suppose we know of an algorithm for solving problem A that runs in O(N^3) time.

Suppose we think of an algorithm that maps problem A into problem B, say in O(N) time.

Suppose we know of an algorithm that solves B in O(N^2) time.

By combining the mapping algorithm with that for solving B, we now have an algorithm for A that runs in O(N^2) time.

So far so good.

What does this mean for the relative complexities of problems A and B?

Preview: Reductions for Complexity Theory

$$A \xrightarrow{\text{efficient mapping}} B$$

When we have an efficient mapping of A→B, then solving A cannot be harder than solving B.

In other words, we should use the $O(N^2)$ algorithm that includes the reduction rather than the original $O(N^3)$ algorithm.

We can state that: Solving *A* cannot be harder than solving *B*, or

$$A \leq_m B$$

Linear Programming (LP)

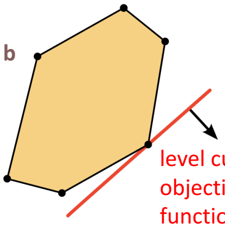
Linear programming (**LP**) is a method to achieve the best outcome (such as maximum profit or lowest cost) in a mathematical model whose requirements are represented by linear relationships. [Wikipedia]

Linear Programming (LP)

Maximize a linear function subject to linear constraints and nonnegative constraints

Canonical form:

$$\begin{array}{ll} \text{maximize} & \mathbf{c}^{\mathrm{T}}\mathbf{x} \\ \text{subject to} & A\mathbf{x} \leq \mathbf{b} \\ \text{and} & \mathbf{x} \geq \mathbf{0} \end{array}$$



Feasible region: defined by $A\mathbf{x} \leq \mathbf{b}$

level curve of objective function $\mathbf{c}^{\mathrm{T}}\mathbf{x}$

Solution algorithms

- Simplex method (Ch. 10)
  - Worst-case exponential time, but fastest in practice
- Interior point methods (Ellipsoid, Karmarkar's algorithm)
  - Worst-case polynomial time, theoretical importance

Linear Programming Example

A university endowment has to invest $100 million.

Sum is to be split into three types of investments: stocks, bonds and cash.

The expected returns are 10%, 7% and 3% for each of the above.

Since stocks are more risky than bonds, the endowment rules require that the amount invested in stocks be no more than one-third on the money invested in bonds.

Additionally, at least 25% of the total amount invested in stock and bonds must be invested in cash.

How should the money be allocated to maximize return?

Linear Programming Example

A university endowment has to invest $100 million.

Sum is to be split into three types of investments: stocks, bonds and cash.

The expected returns are 10%, 7% and 3% for each of the above.

Since stocks are more risky than bonds, the endowment rules require that the amount invested in stocks be no more than one-third on the money invested in bonds.

Additionally, at least 25% of the total amount invested in stock and bonds must be invested in cash.

How should the money be allocated to maximize return?

$$\begin{array}{ll} \text{maximize} & 0.10x + 0.07y + 0.03z \\ \text{subject to} & x + y + z = 100 \\ & x \leq \frac{1}{3}y \\ & z \geq 0.25(x + y) \\ & x \geq 0, \quad y \geq 0, \quad z \geq 0. \end{array}$$

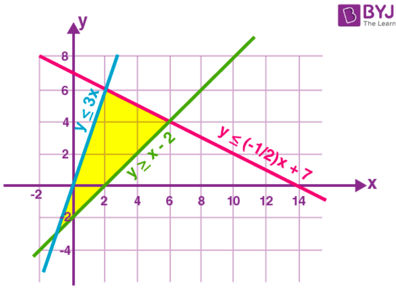
LP Example

Calculate the maximum value of $z = 5x + 3y$, given the following constraints.

$x + 2y \leq 14$
$3x - y \geq 0$
$x - y \leq 2$

The marked area of the plane is the feasible region.

You have to find the (x,y) corner points that give the largest values of z.



Example source: https://byjus.com/maths/linear-programming/

LP Example

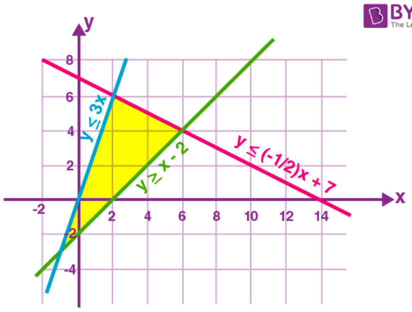
To begin with, first solve each inequality.

x + 2y ≤ 14 ⇒ y ≤ -(1/2)x + 7

3x – y ≥ 0 ⇒ y ≤ 3x

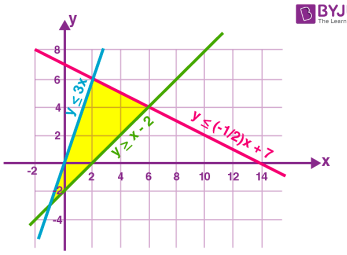
x – y ≤ 2 ⇒ y ≥ x – 2

BYJU'S
The Learning App



Example source: https://byjus.com/maths/linear-programming/

BYJU'S
The Learning App



LP Example

Now pair the lines to form a system of linear equations to find the corner points.

y = -(½) x + 7

y = 3x

Solving the above equations, we get the corner point of (2, 6)

y = -1/2 x + 7

y = x – 2

Solving the above equations, we get the corner point of (6, 4)

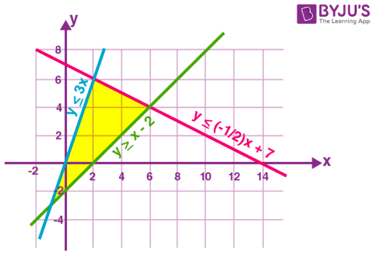
y = 3x

y = x – 2

Solving the above equations, we get the corner point of (-1, -3)

Example source: https://byjus.com/maths/linear-programming/

LP Example



For linear systems, the maximum values of the optimization equation lie on the corners of the feasibility region.

To find the optimum solution, you only need to plug these three points in z = 5x + 3y

(2, 6) : z = 5*2 + 3*6 = 10 + 18 = 28

(6, 4): z = 5*6 + 3*4 = 30 + 12 = 42

(–1, –3): z = 5*(-1) + 3*(-3) = -5 -9 = -14

The maximum of z = 42 lies at (6, 4)

Example source: https://byjus.com/maths/linear-programming/

Reduction: Fractional Knapsack → LP

Consider the variant of Knapsack that allows fractional amounts of each object

- *n* items; weights w_i and values v_i
- Knapsack weight capacity *C*
- x_i represents amount of item *i*
- LP problem:

maximize $\sum v_i x_i$

subject to $\sum w_i x_i \le C$

$\mathbf{x} \le \mathbf{1}$ and $\mathbf{x} \ge \mathbf{0}$