# Operations Research II: Algorithms

# Branch & Bound and Heuristic Algorithms

Ling-Chieh Kung

Department of Information Management
National Taiwan University

# Introduction

- In some cases, variables must only take **integer values**.
- The subject of formulating and solving models with integer variables is **Integer Programming** (IP).
  - An IP is typically a linear IP (LIP).
  - If the objective function or any functional constraint is nonlinear, it is a nonlinear IP (NLIP).
  - We will focus on linear IP in this course.

# Introduction

- The **branch-and-bound algorithm** finds an **optimal** solution for any IP.
  - It "decomposes" an IP to multiple LPs, solve all the LPs, and compares those outcomes to reach a conclusion.
  - Each LP is solved separately (with the simplex method or other ways).
  - In general, the process may take a very long time.
- As finding an optimal solution for an IP may be too time-consuming, we often look for a **near-optimal** feasible solution instead in practice.
  - An algorithm that generates a **feasible** solution in a **short time** is called a **heuristic algorithm**.
  - **Hopefully** it is **near-optimal**.
  - A good heuristic algorithm does not always work, but it works for most of the time.

# Road map

- **Linear relaxation**.
- The branch-and-bound algorithm.
- Solving the knapsack problem.
- Heuristic algorithms.

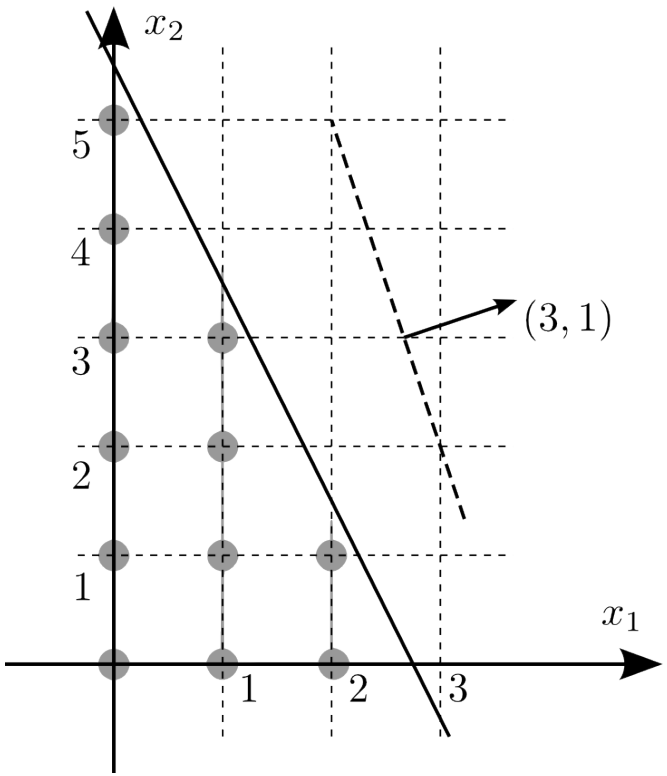
# Solving an IP

- Suppose we are given an IP, how may we solve it?
- The simplex method does not work!
  - The feasible region is not "a region".
  - It is **discrete**.
  - There is no way to "move along edges".
- But all we know is how to solve LPs. How about solving a **linear relaxation** first?

## Definition 1 (Linear relaxation)

*For a given IP, its linear relaxation is the resulting LP after removing all the integer constraints.*

$$\begin{array}{lrcrcl} \max & 3x_1 & + & x_2 & & \\ \text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\ & x_i \in \mathbb{Z}_+ & & \forall i = 1, 2. & & \end{array}$$



# Linear relaxation

- What is the linear relaxation of

$$\begin{array}{lrcrcl} \max & x_1 & + & x_2 & & \\ \text{s.t.} & x_1 & + & 3x_2 & \leq & 10 \\ & 2x_1 & - & x_2 & \geq & 5 \\ & x_i \in \mathbb{Z}_+ & & \forall i = 1, 2? & & \end{array}$$

  - $\mathbb{Z}$ is the set of all integers. $\mathbb{Z}_+$ is the set of all nonnegative integers.
- The linear relaxation is

$$\begin{array}{lrcrcl} \max & x_1 & + & x_2 & & \\ \text{s.t.} & x_1 & + & 3x_2 & \leq & 10 \\ & 2x_1 & - & x_2 & \geq & 5 \\ & x_i \geq 0 & & \forall i = 1, 2. & & \end{array}$$

# Linear relaxation

- For the knapsack problem

$$\begin{array}{lrcrcrcrcl} \max & 16x_1 & + & 22x_2 & + & 12x_3 & + & 8x_4 & & \\ \text{s.t.} & 5x_1 & + & 7x_2 & + & 4x_3 & + & 3x_4 & \leq & 10 \\ & & & x_i \in \{0,1\} & & \forall i = 1,...,4, & & & & \end{array}$$

  the linear relaxation is

$$\begin{array}{lrcrcrcrcl} \max & 16x_1 & + & 22x_2 & + & 12x_3 & + & 8x_4 & & \\ \text{s.t.} & 5x_1 & + & 7x_2 & + & 4x_3 & + & 3x_4 & \leq & 10 \\ & & & x_i \in [0,1] & & \forall i = 1,...,4, & & & & \end{array}$$

  - $x_i \in [0,1]$ is equivalent to $x_i \geq 0$ and $x_i \leq 1$.

# Linear relaxation provides a bound

- For a **minimization** IP, its linear relaxation provides a **lower bound**.

  **Proposition 1**

  *Let $z^*$ and $z'$ be the objective values associated to optimal solutions of a minimization IP and its linear relaxation, respectively, then $z' \leq z^*$.*

  *Proof.* They have the same objective function. However, the linear relaxation's feasible region is (weakly) larger than that of the IP. □

- For a **maximization** IP, linear relaxation provides an **upper bound**.

# Linear relaxation may solve the IP

- If we are lucky, the linear relaxation may be infeasible or unbounded.
  - The IP is then infeasible or unbounded.
- If we are lucky, an optimal solution to the linear relaxation may be **feasible** to the original IP. When this happens, the IP is solved:

## Proposition 2

*Let $x'$ be an optimal solutions to the linear relaxation of an IP. If $x'$ is feasible to the IP, it is optimal to the IP.*

*Proof.* Suppose $x'$ is not optimal to the IP, there must be another feasible solution $x''$ that is better. However, as $x''$ is feasible to the IP, it is also feasible to the linear relaxation, which implies that $x'$ cannot be optimal to the linear relaxation. □

- What if we are **unlucky**?

# Rounding a fractional solution

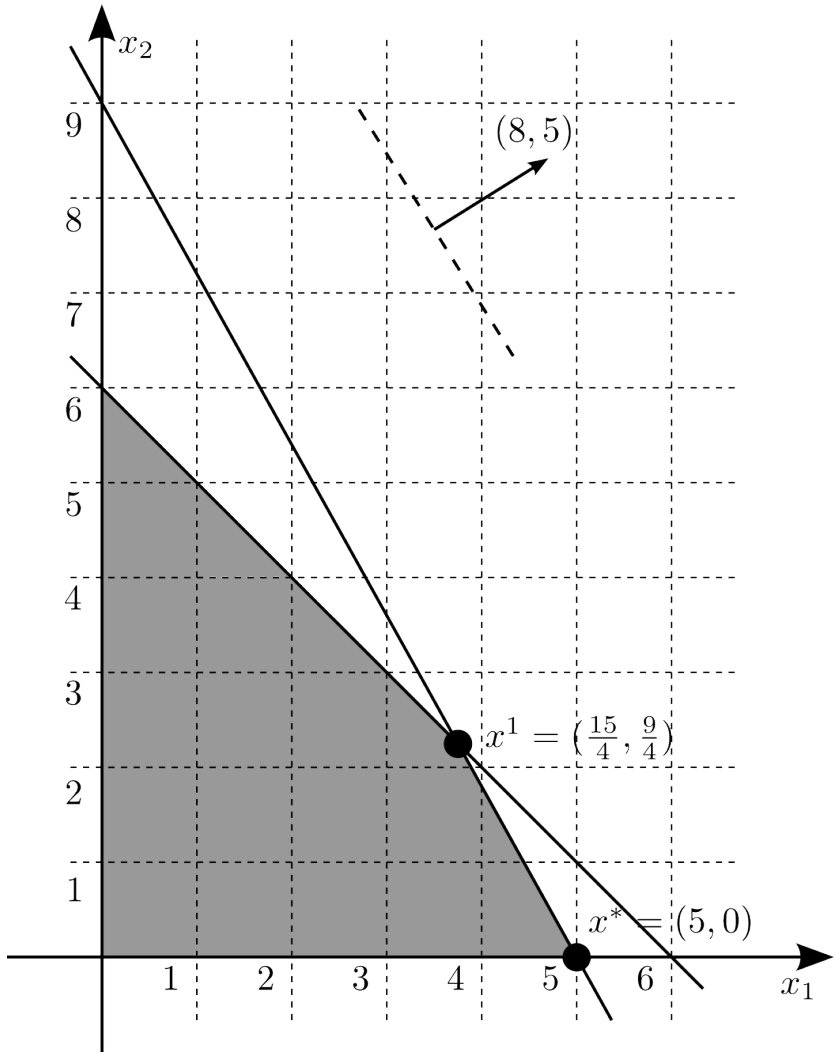
- Suppose we solve a linear relaxation with an LR-optimal solution $x'$.
  - "LR-optimal" means $x'$ is optimal to the linear relaxation.
- $x'$, however, has at least one variable violating the integer constraint in the original IP.
- We may choose to **round** the variable.
  - Round up or down?
  - Is the resulting solution always feasible?
  - Will the resulting solution be close to an IP-optimal solution $x^*$?

# Rounding a fractional solution

- Consider the following IP

$$\begin{array}{lrcrcl} \max & 8x_1 & + & 5x_2 & & \\ \text{s.t.} & x_1 & + & x_2 & \leq & 6 \\ & 9x_1 & + & 5x_2 & \leq & 45 \\ & x_i \in \mathbb{Z}_+ & & \forall i = 1, 2. & & \end{array}$$

- $x^* = (5, 0)$ is IP-optimal.
- But $x^1 = (\frac{15}{4}, \frac{9}{4})$ is LR-optimal!
  - Rounding up any variable results in infeasible solutions.
  - None of the four grid points around $x^1$ is optimal.
- We need a way that guarantees to find an optimal solution.

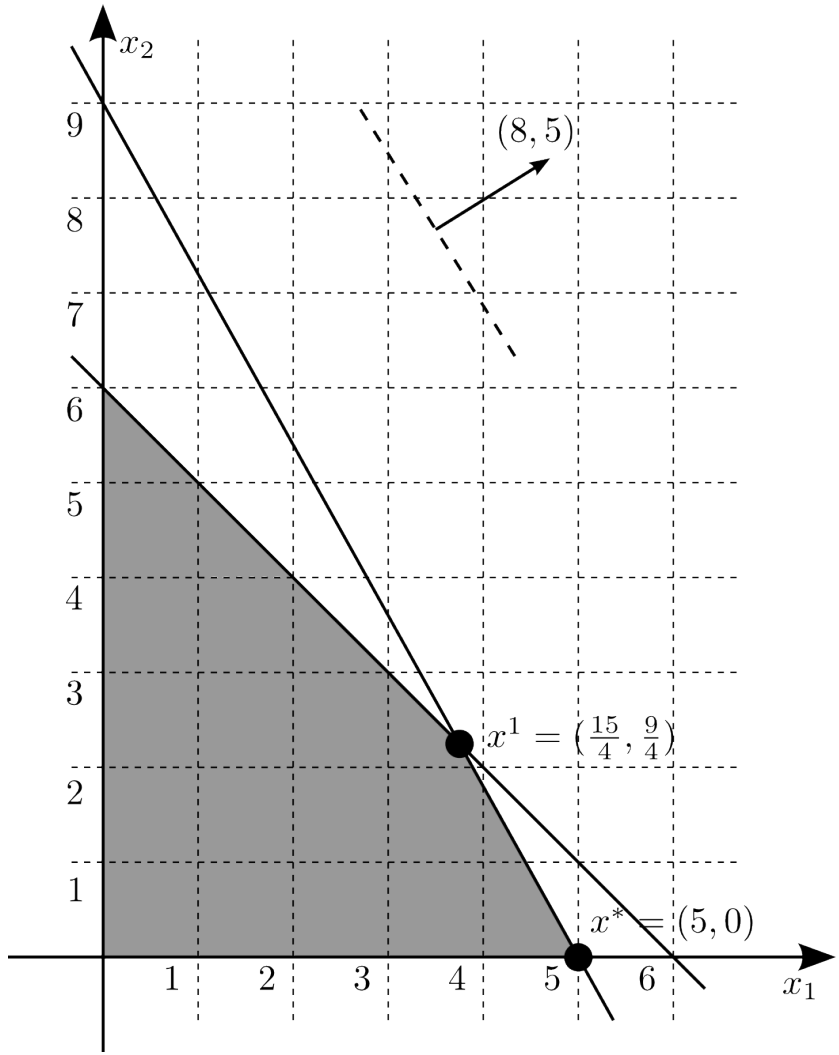


# Road map

- Linear relaxation.
- **The branch-and-bound algorithm**.
- Solving the knapsack problem.
- Heuristic algorithms.

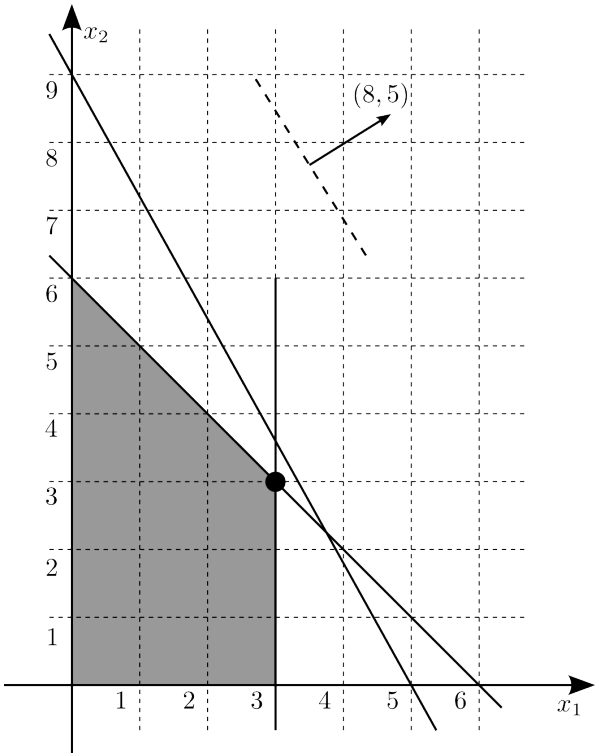
# Rounding a fractional solution

- $x^1 = (\frac{15}{4}, \frac{9}{4})$ is LR-optimal.
  - Rounding up or down $x_1$ (i.e., adding $x_1 = 4$ or $x_1 = 3$ into the program) both **fail** to find the optimal solution.
  - Because we eliminate too many feasible points!
  - Instead of adding equalities, we should add **inequalities**.
- What will happen if we add $x_1 \geq 4$ or $x_1 \leq 3$ into the program?
- We will **branch** this problem into two problems, one with an additional constraint.

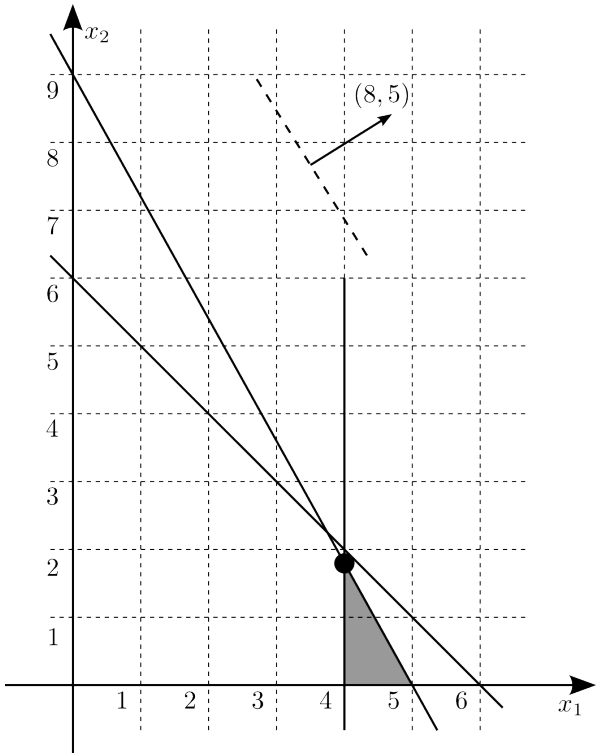


# Rounding a fractional solution

If we add $x_1 \leq 3$:




If we add $x_1 \geq 4$:




- The optimal solution to the IP must be contained in one of the above two feasible regions. Why?

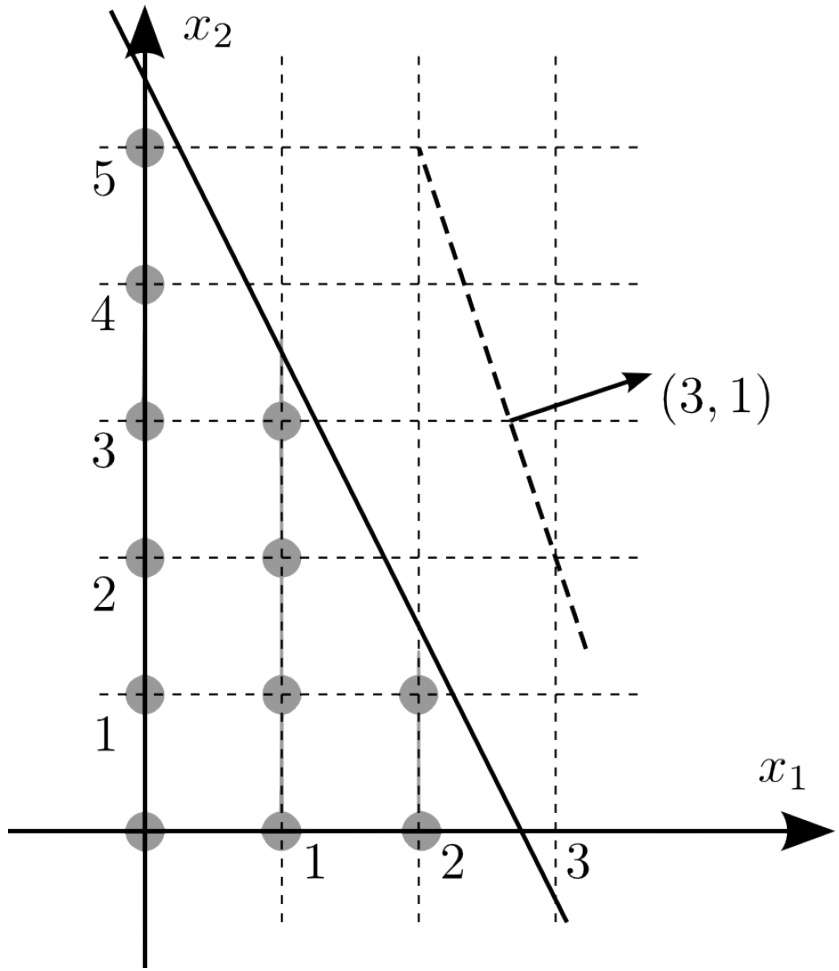
# Rounding a fractional solution

- So when we solve the linear relaxation and find any variable violating an integer constraint, we will **branch** this problem into two problems, one with an additional constraint.
- The two new programs are still linear programs.
- Once we solved them:
  - If their LR-optimal solutions are both IP-feasible, compare them and choose the better one.
  - If any of them results in a variable violating the integer constraint, **branch** on that variable **recursively**.
  - Eventually compare all the IP-feasible solutions we obtain.

# Example

- Let's illustrate the branch-and-bound algorithm with the following example:

$$
(P_0)\quad
\begin{array}{lrcrcl}
\max & 3x_1 & + & x_2 & & \\
\text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\
& x_i & \in & \mathbb{Z}_+ & \forall i = 1, 2. &
\end{array}
$$

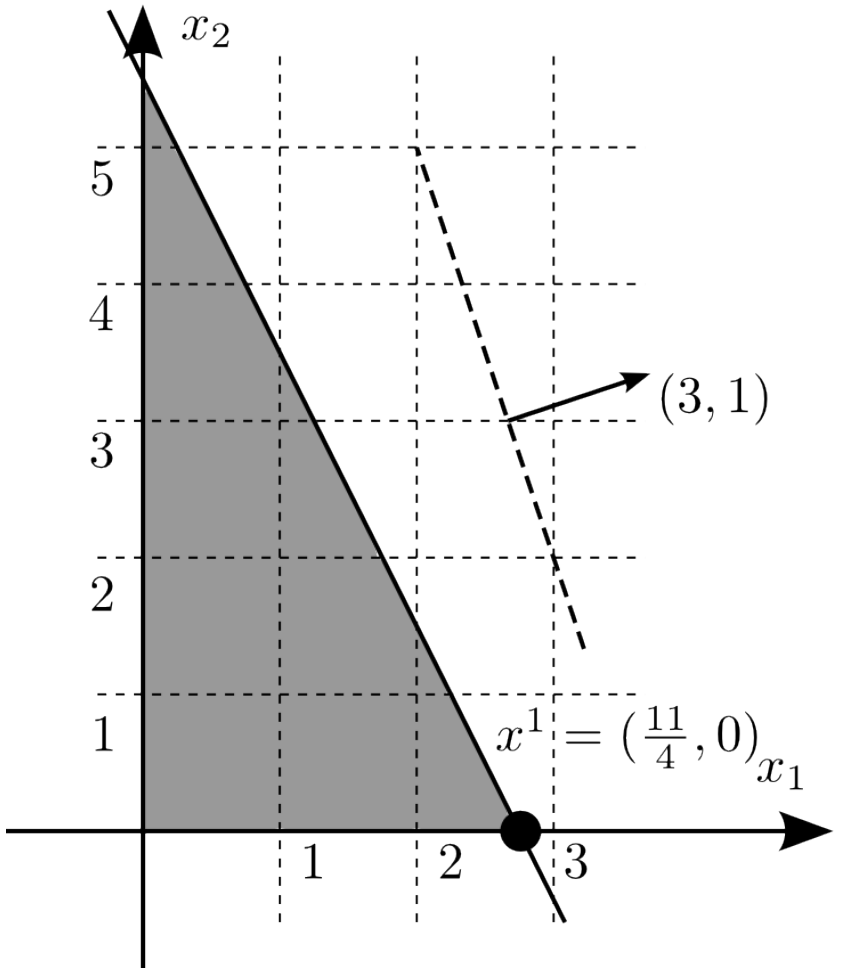


# Subproblem 1

- First we solve the linear relaxation:

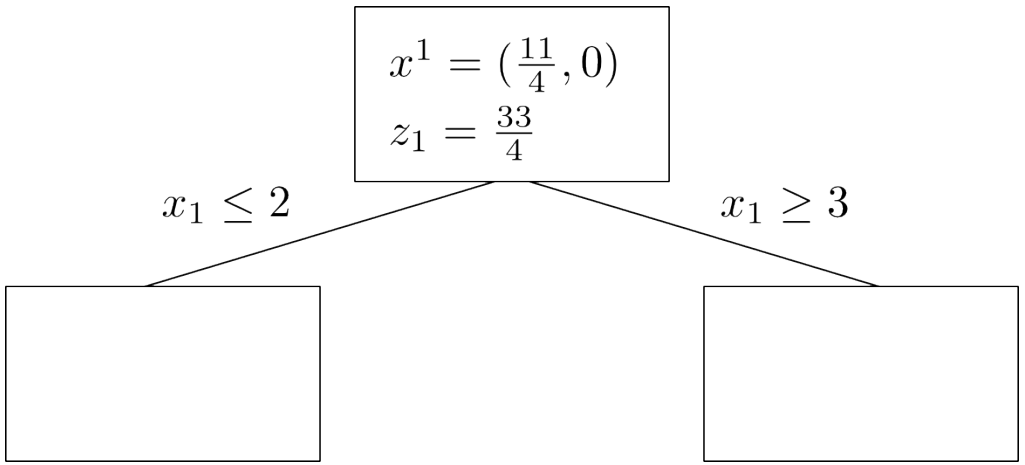
$$(P_1)\quad \begin{array}{lrcrcl} & \max & 3x_1 & + & x_2 & \\ & \text{s.t.} & 4x_1 & + & 2x_2 & \leq \quad 11 \\ & & x_i \geq 0 & & \forall i = 1, 2. & \end{array}$$

- The optimal solution is $x^1 = (\frac{11}{4}, 0)$.
- So we need to branch on $x_1$.



# Branching tree

- The branch and bound algorithm produces a **branching tree**.
  - Each node represents a subproblem (which is an LP).
  - Each time we branch on a variable, we create two child nodes.

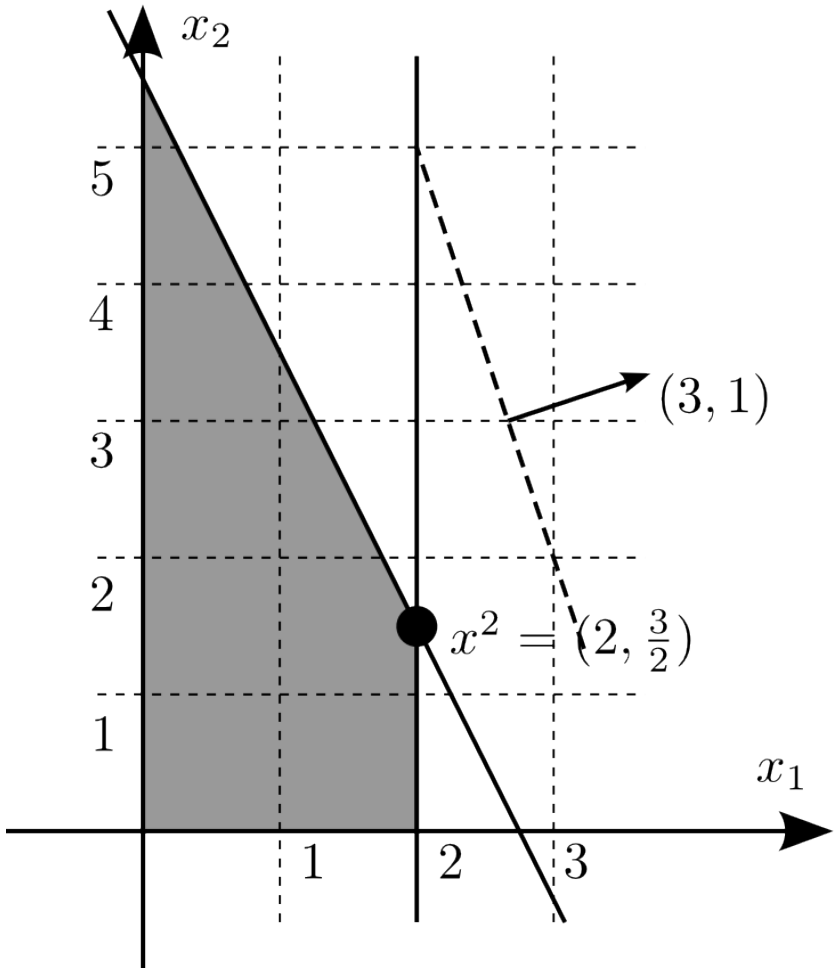


# Subproblem 2

- When we add $x_1 \leq 2$:

$$(P_2) \quad \begin{array}{lrcrcl} \max & 3x_1 & + & x_2 & & \\ \text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\ & x_1 & & & \leq & 2 \\ & x_i \geq 0 & \forall i = 1, 2. & & & \end{array}$$

- An $(P_2)$-optimal solution is $x^2 = (2, \frac{3}{2})$.
  - So later we need to branch on $x_2$.
- Before that, let's solve $(P_3)$.

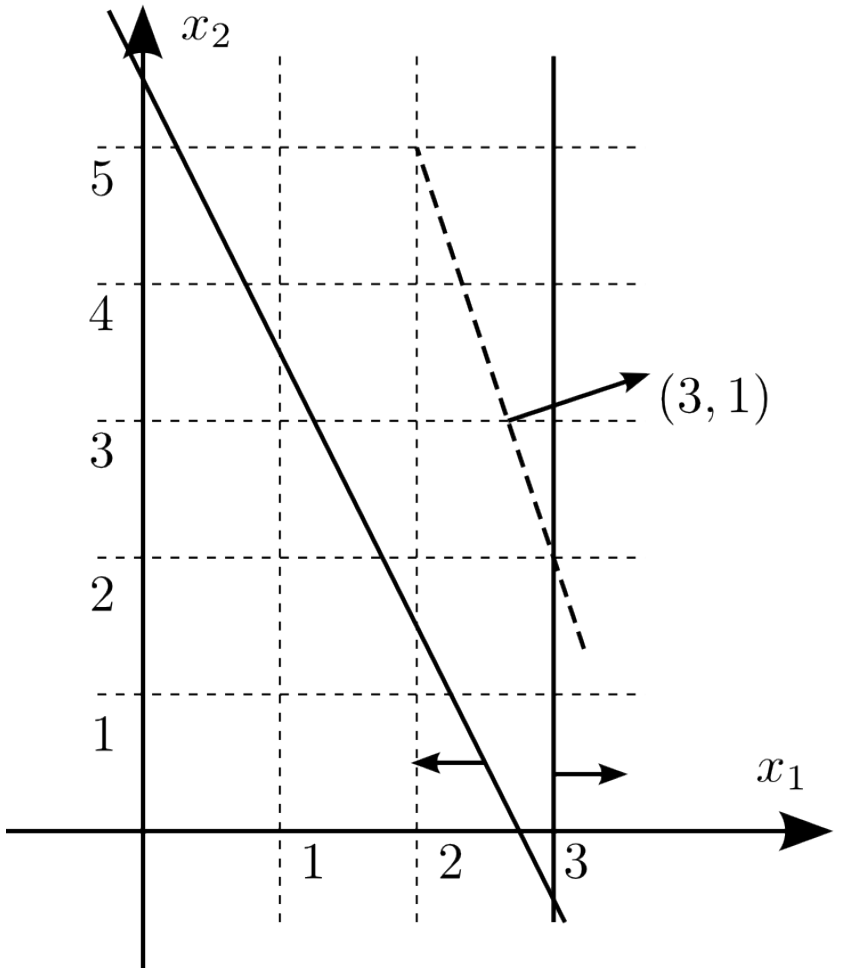


# Subproblem 3

- When we add $x_1 \geq 3$:

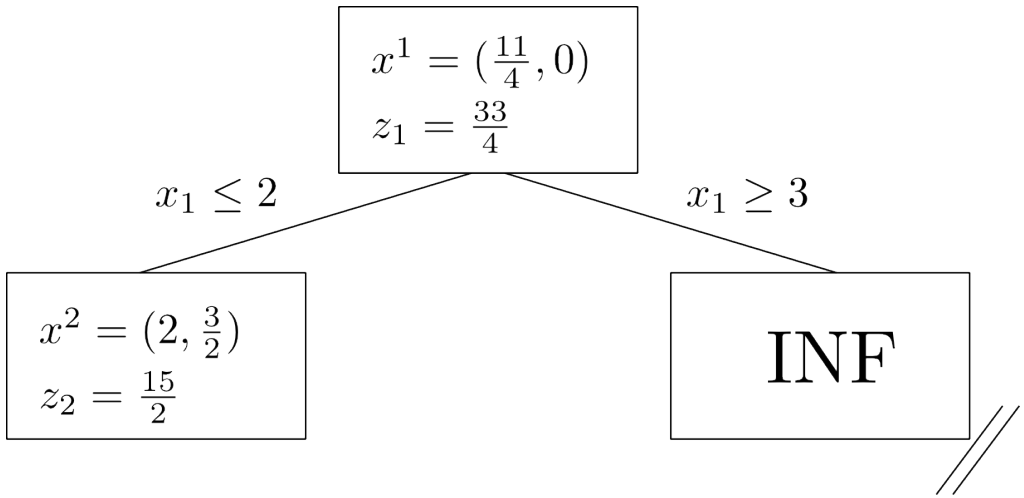
$$(P_3) \quad \begin{array}{lrcrcl} \max & 3x_1 & + & x_2 & & \\ \text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\ & x_1 & & & \geq & 3 \\ & x_i \geq 0 & \forall i = 1, 2. & & & \end{array}$$

- The problem is infeasible!
- This node is "dead" and does not produce any candidate solution.



# Branching tree

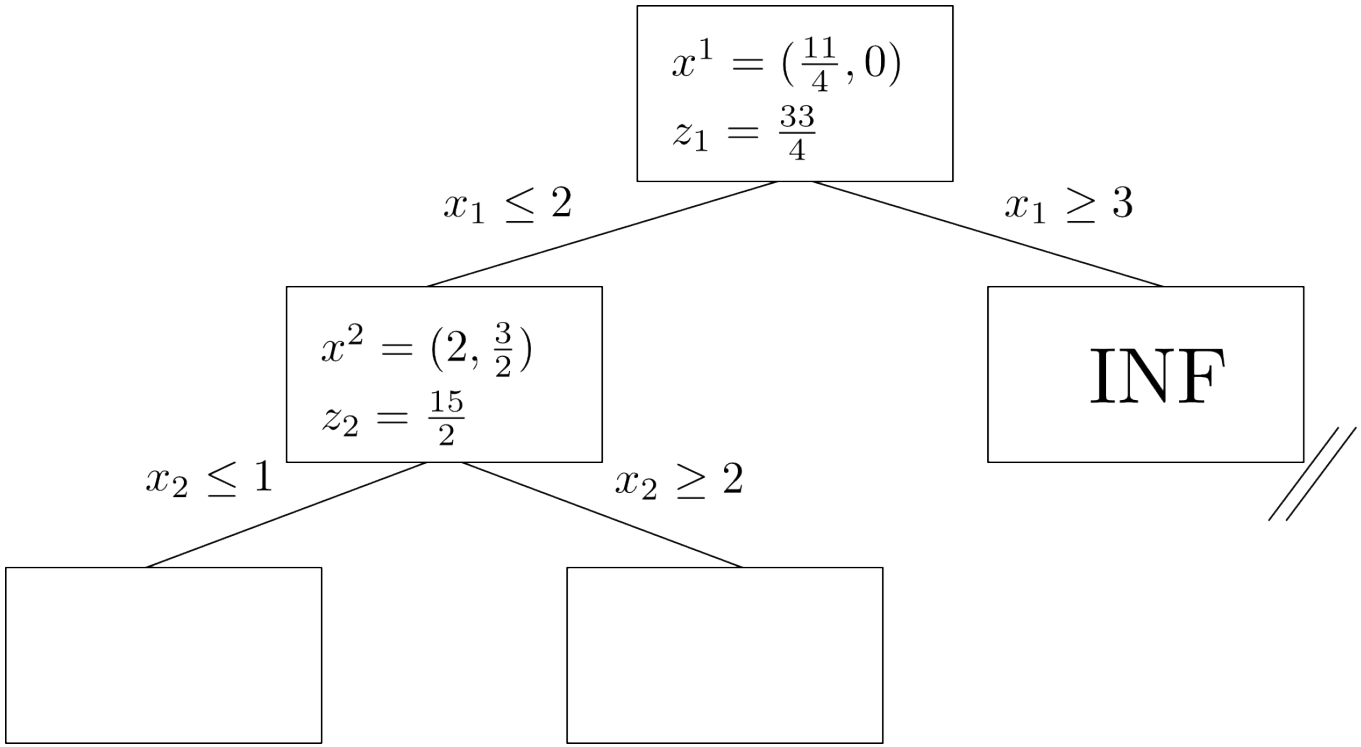
- The current progress can be summarized in the branching tree.




- Note that $z_2 = 7.5 < 8.25 = z_1$.
- In general, when we branch to the next level, the objective value associated with a subproblem-optimal solution will **always** be weakly **lower** (for a maximization problem). Why?

# Branching tree

- As $x_2 = \frac{3}{2}$ in $x^2$, we will branch subproblem 2 on $x_2$.

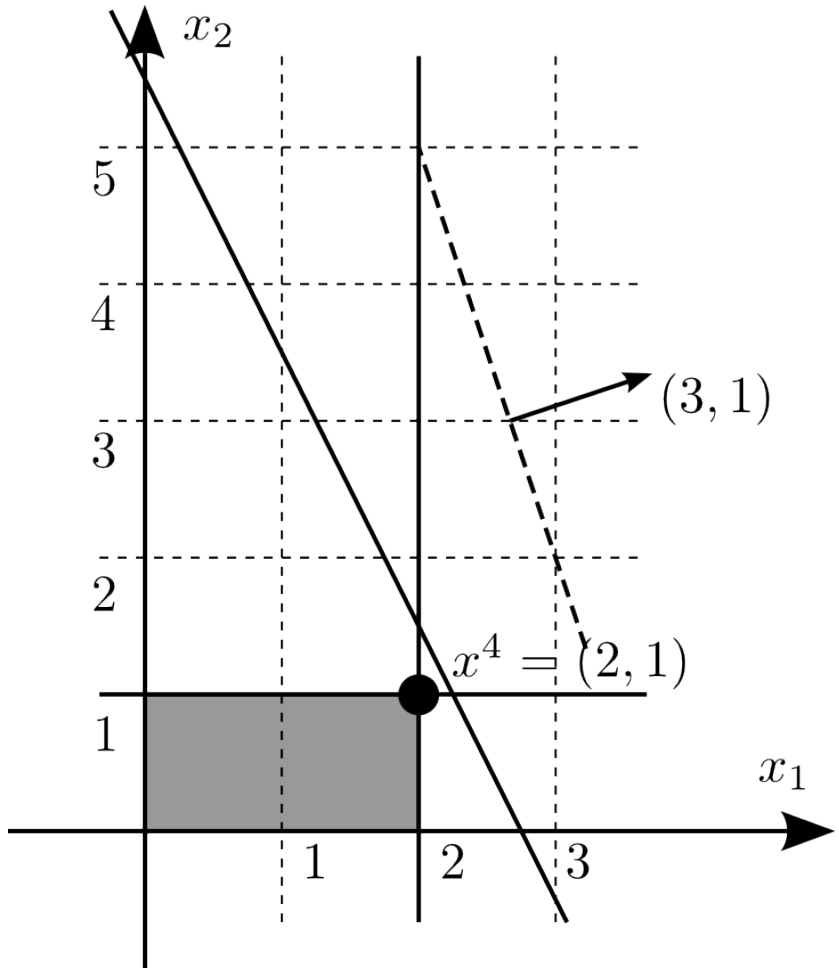


# Subproblem 4

- When we add $x_2 \leq 1$:

$$(P_4) \quad \begin{array}{lrcrcl} \max & 3x_1 & + & x_2 & & \\ \text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\ & x_1 & & & \leq & 2 \\ & & & x_2 & \leq & 1 \\ & x_i \geq 0 & & \forall i = 1, 2. & & \end{array}$$

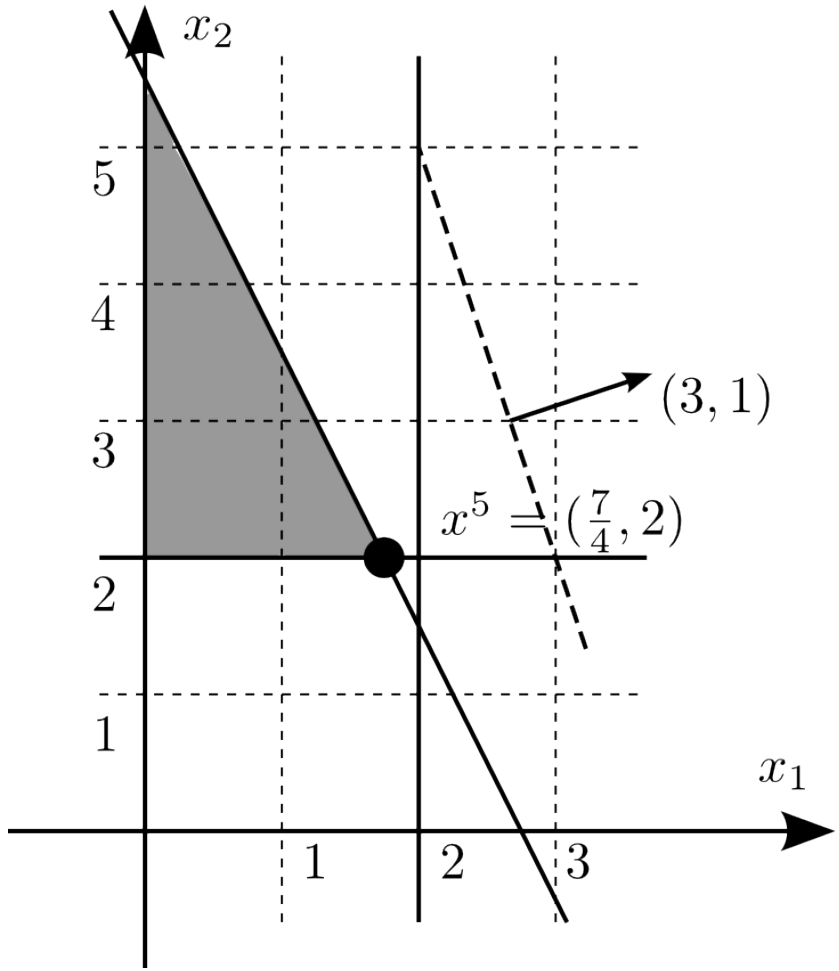
- Note that we add $x_2 \leq 1$ into subproblem 2, so $x_1 \leq 2$ is still there.



# Subproblem 5

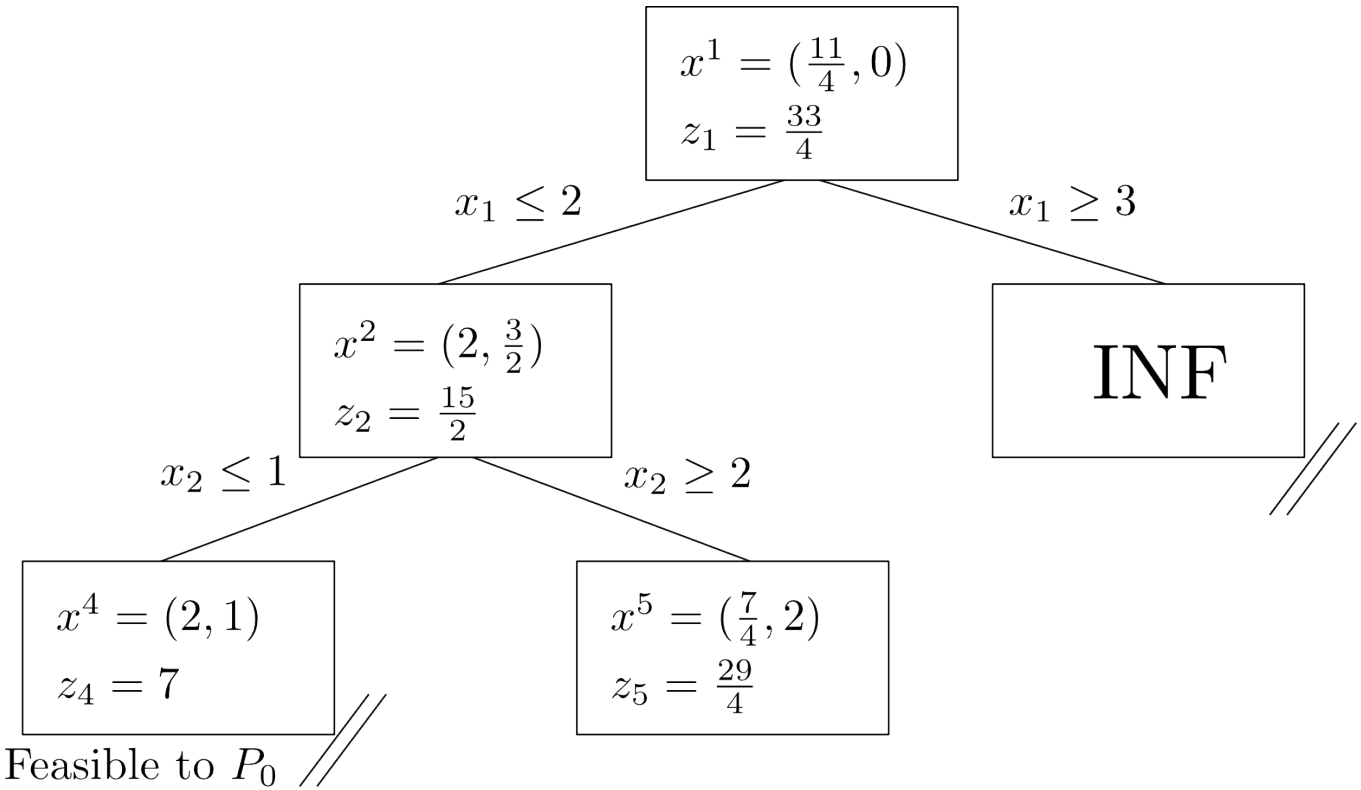
- When we add $x_2 \geq 2$:

$$
(P_5)\quad
\begin{array}{rrcrcl}
\max & 3x_1 & + & x_2 & & \\
\text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\
 & x_1 & & & \leq & 2 \\
 & & & x_2 & \geq & 2 \\
 & x_i \geq 0 & & \forall i = 1, 2. & &
\end{array}
$$



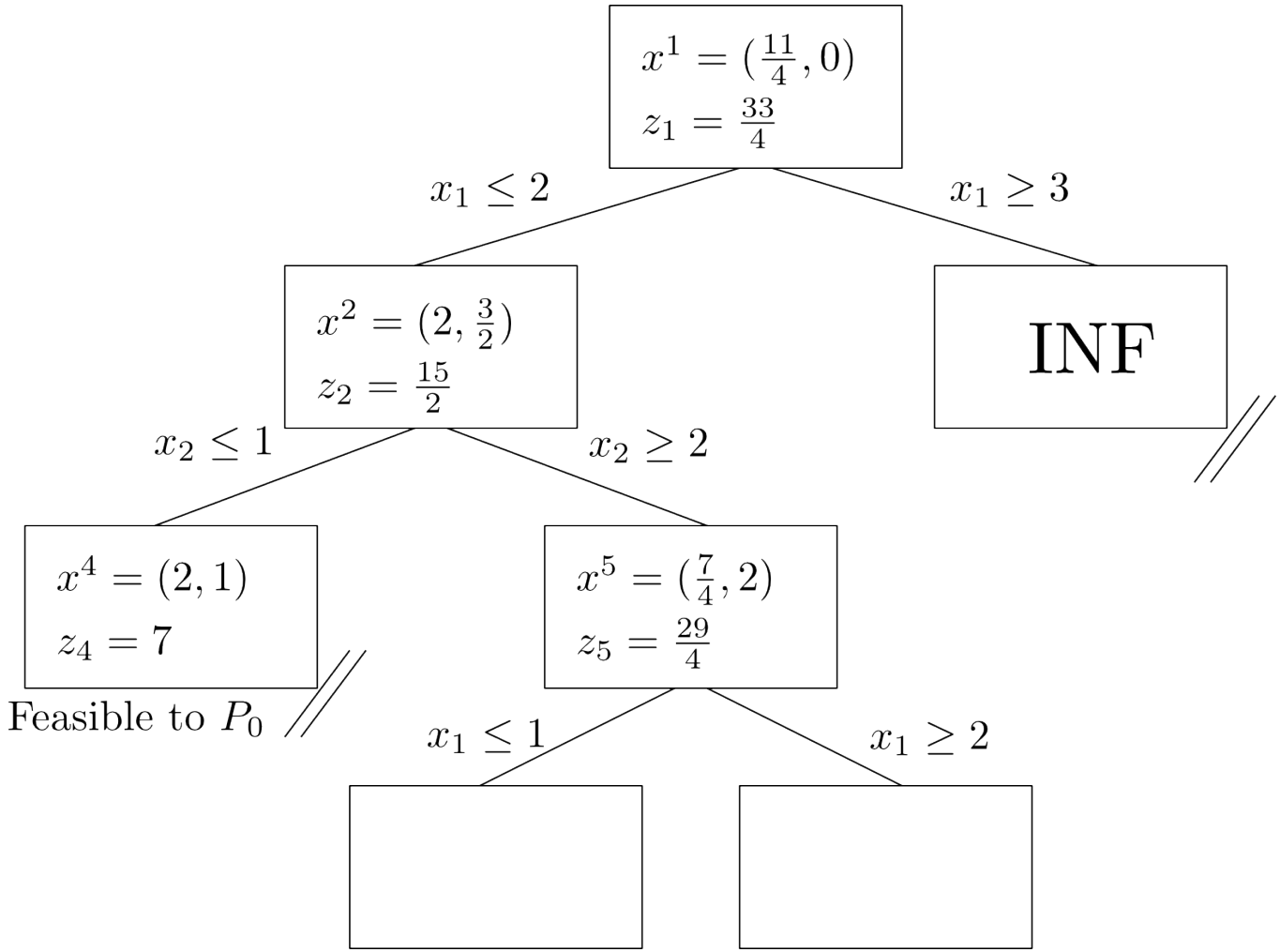
# Branching tree

- $x^4$ satisfies all the integer constraints.
- It is IP-feasible and thus a **candidate solution** to the original IP.
- But branching subproblem 5 may result in a better solution.



# Branching tree

- Let's branch subproblem 5 on $x_1$.

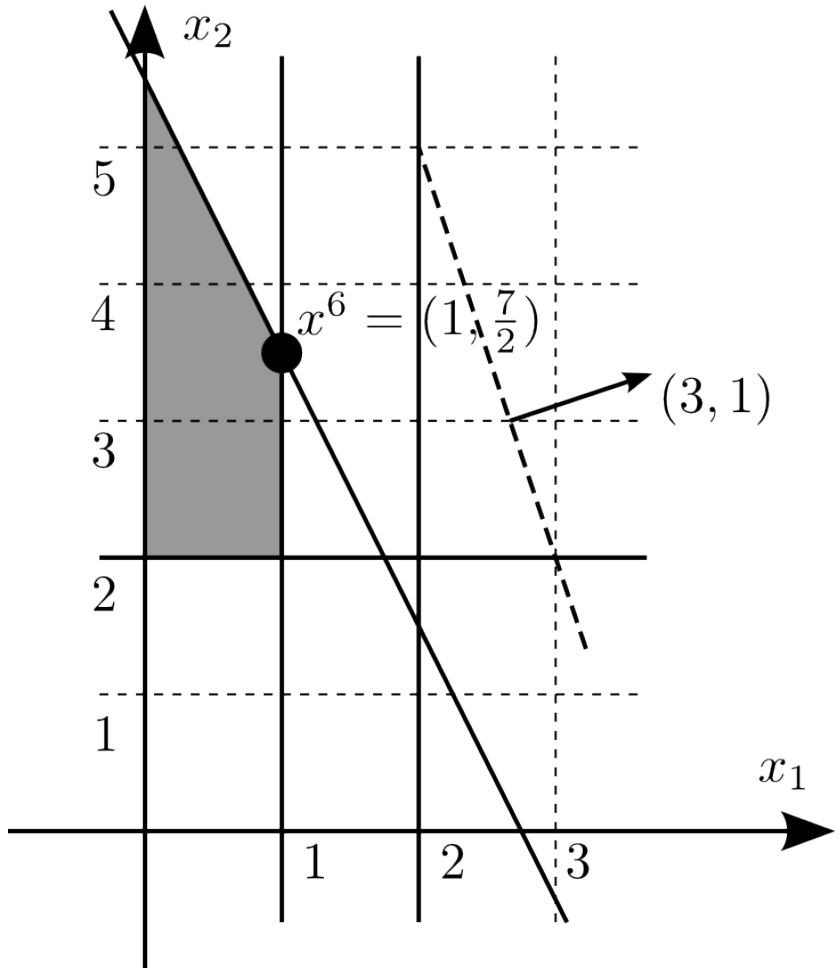


# Subproblem 6

- When we add $x_1 \leq 1$:

$$
(P_6)\quad
\begin{array}{rrcrcl}
\max & 3x_1 & + & x_2 & & \\
\text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\
 & x_1 & & & \leq & 2 \\
 & & & x_2 & \geq & 2 \\
 & x_1 & & & \leq & 1 \\
 & x_i \geq 0 & & \forall i = 1, 2. & &
\end{array}
$$

- $x^6 = (1, \frac{7}{2})$. We may need to branch on $x_2$ again. However, let's solve subproblem 7 first.

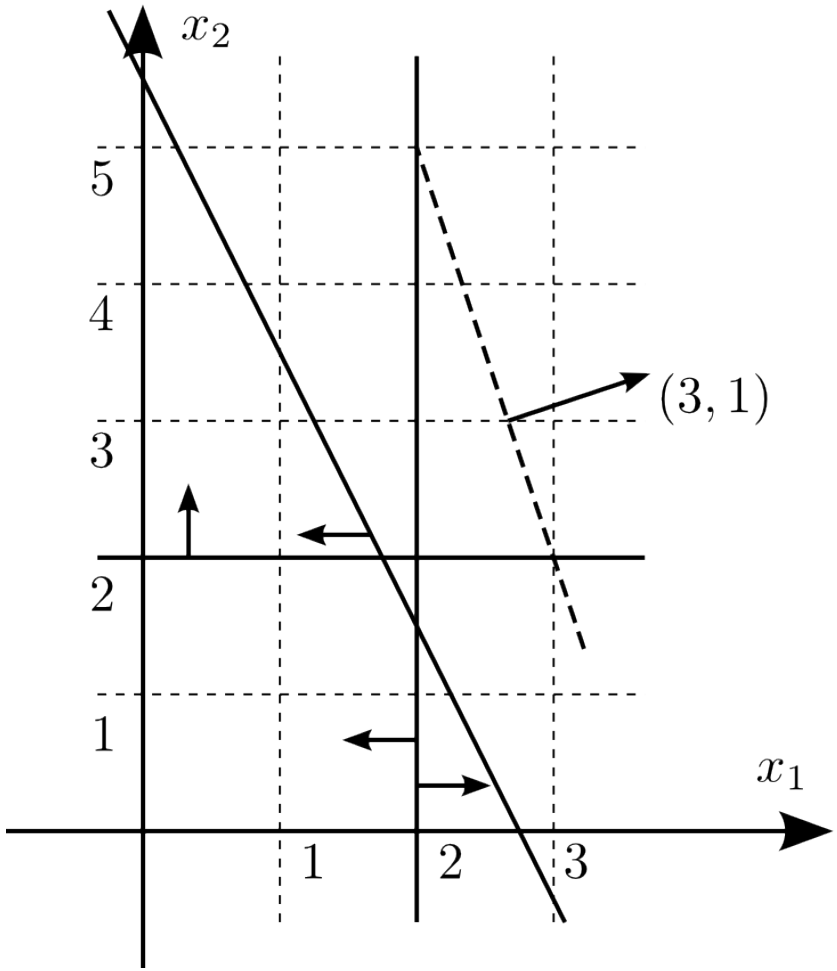


# Subproblem 7

- When we add $x_1 \geq 2$:

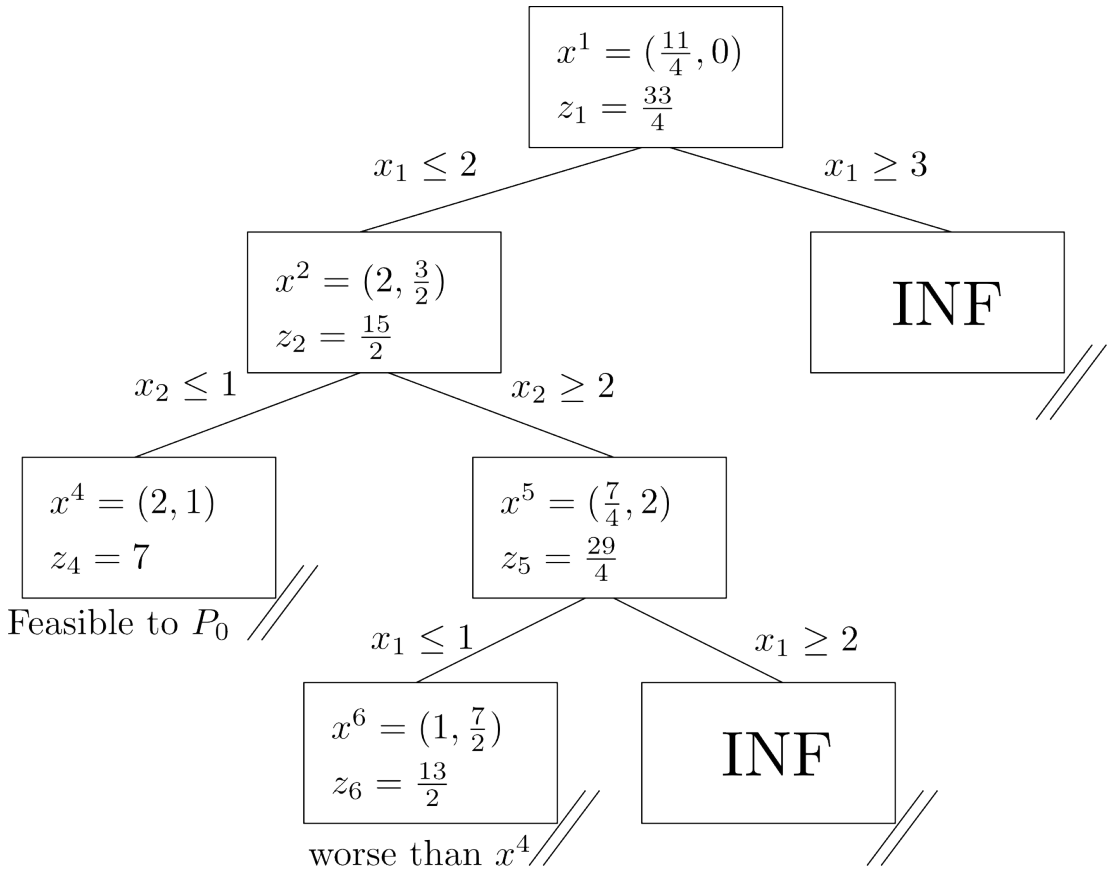
$$
(P_7)\quad
\begin{array}{rrcrcl}
\max & 3x_1 & + & x_2 & & \\
\text{s.t.} & 4x_1 & + & 2x_2 & \leq & 11 \\
& x_1 & & & \leq & 2 \\
& & & x_2 & \geq & 2 \\
& x_1 & & & \geq & 2 \\
& x_i \geq 0 & \forall i = 1, 2. & & &
\end{array}
$$

- The problem is infeasible.
- The node is "dead".



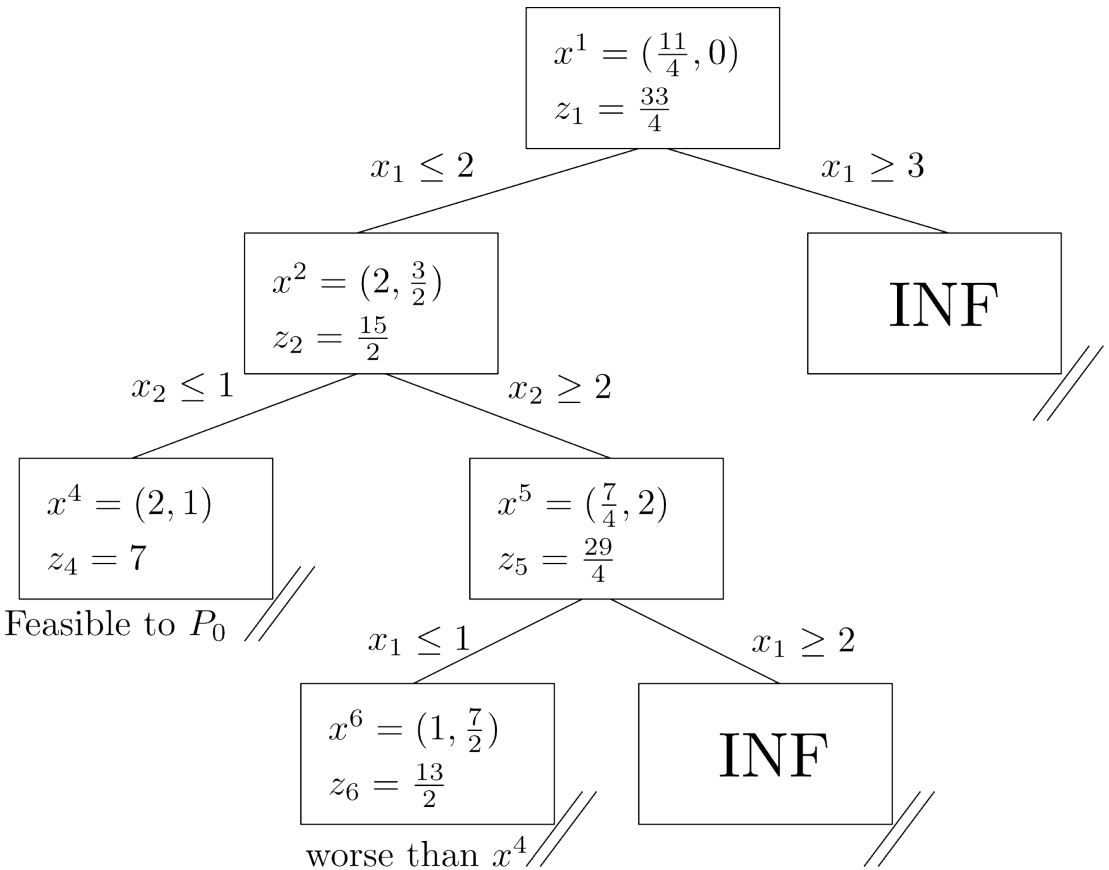
# Branching tree

- The only "alive" node is subproblem 6, with $x_2$ fractional.
- Before we branch subproblem 6, consider the following:



# Bounding

- $z_6 = \frac{13}{2}$. If we branch ($P_6$), all the candidate solutions (if any) under it will be (weakly) **worse** than $\frac{13}{2}$.
- However, $\frac{13}{2} < 7 = z_4$, and $x_4$ is already a candidate!
- So there is no need to branch ($P_6$). This is the "**bounding**" situation in the branch-and-bound algorithm.
  - This allows us to solve fewer subproblems.



# Summary

- In running the branch-and-bound algorithm, we maintain a tree.
- If a subproblem-optimal solution is IP-feasible, set it to the candidate solution if it is currently the best among all IP-feasible solutions. Stop branching this node.
- If a subproblem is infeasible, stop branching this node.
- If a subproblem-optimal solution is not IP-feasible:
  - If it is better than the current candidate solution, branch.
  - Otherwise, stop branching.

# Another example

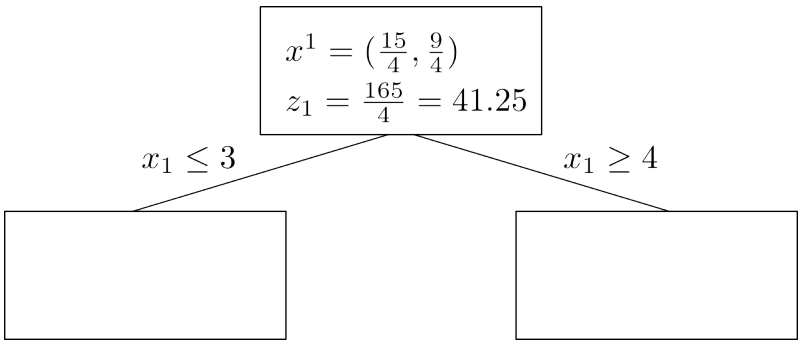
- Now let's go back to our motivating example:

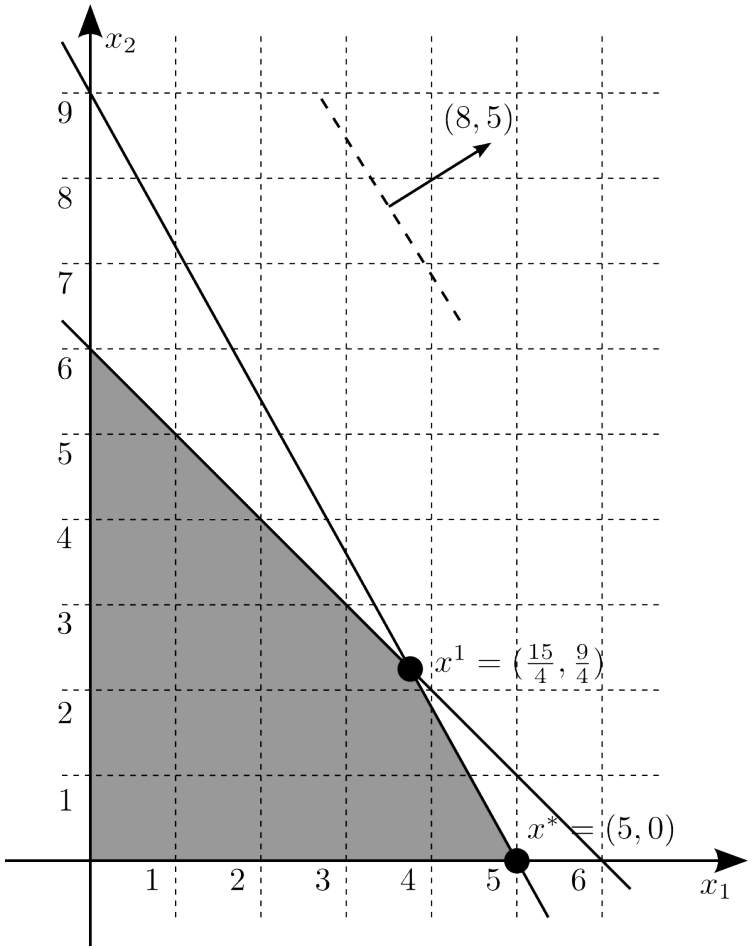
$$(Q_0) \qquad \begin{array}{rrcrcl} \max & 8x_1 & + & 5x_2 & & \\ \text{s.t.} & x_1 & + & x_2 & \leq & 6 \\ & 9x_1 & + & 5x_2 & \leq & 45 \\ & x_i \in \mathbb{Z}_+ & & \forall i = 1, 2. & & \end{array}$$

- Let's solve it with the branch-and-bound algorithm.

# Subproblem 1

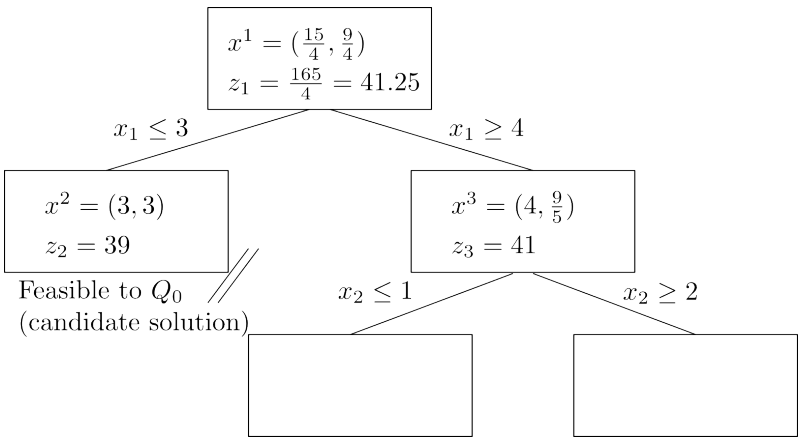
- $x^1 = (\frac{15}{4}, \frac{9}{4})$.
- We may branch on either variable. Let's branch on $x_1$.

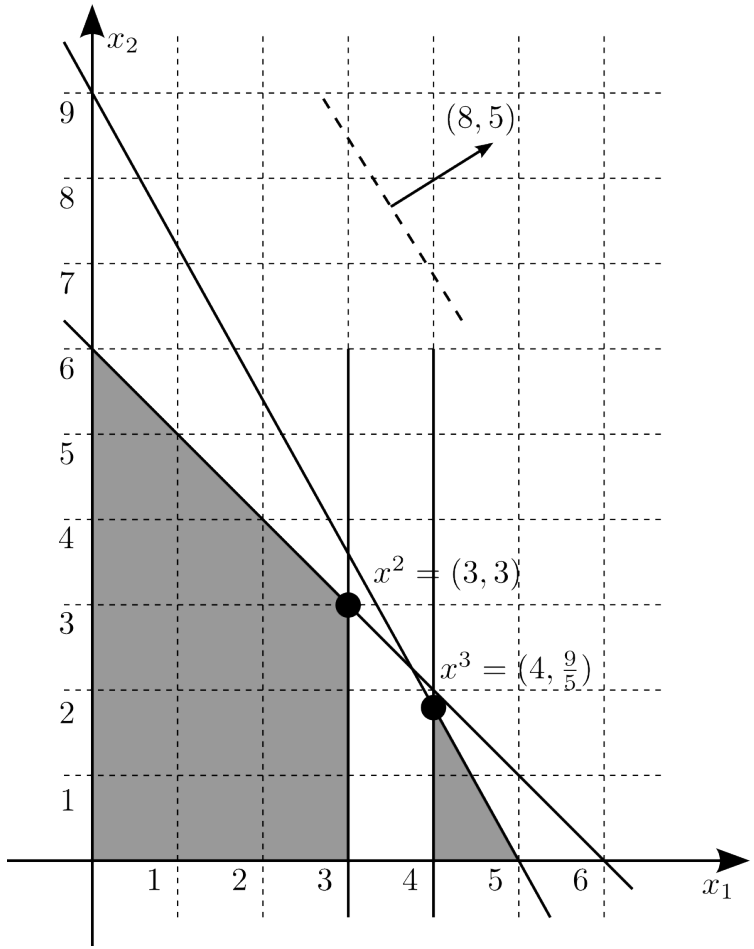


# Subproblems 2 and 3

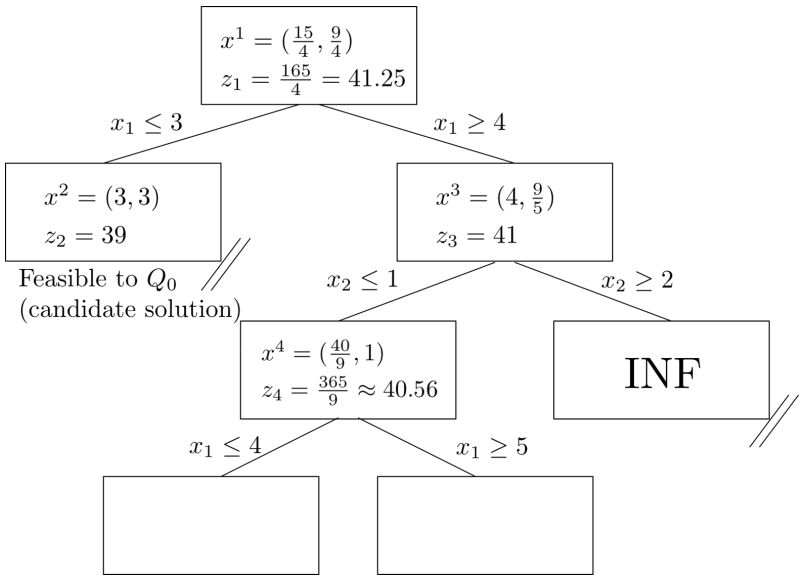
- Subproblem 2 generates a candidate solution.
- $x^3 = (4, \frac{9}{5})$. As $z_3 = 41 > z_2 = 39$, we should branch subproblem 3.

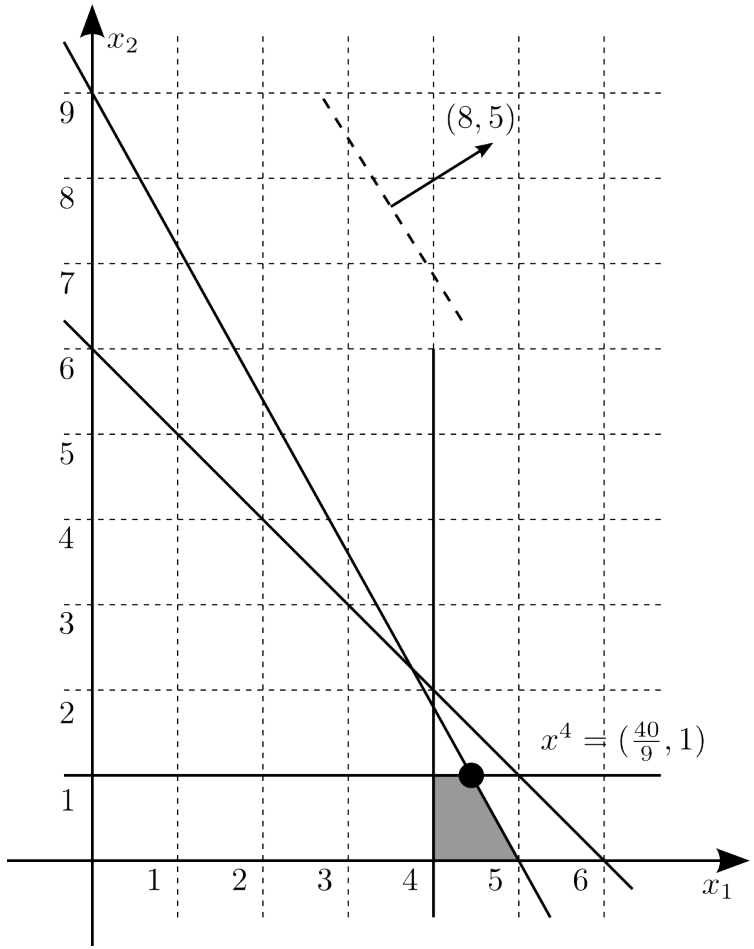


# Subproblems 4 and 5

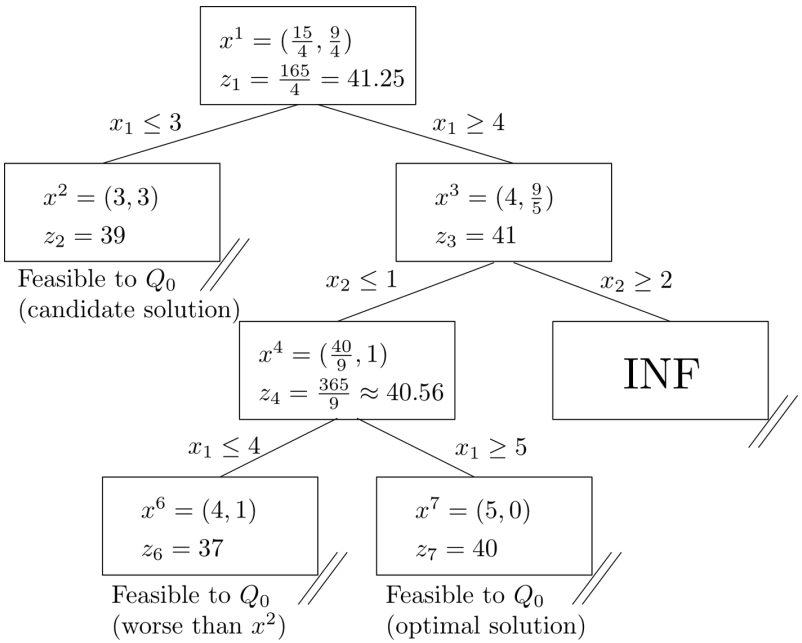
- $x^4 = (\frac{40}{9}, 1)$. As $z_4 = 40.56 > z_2 = 39$, we should branch subproblem 4.
- Subproblem 5 is infeasible.

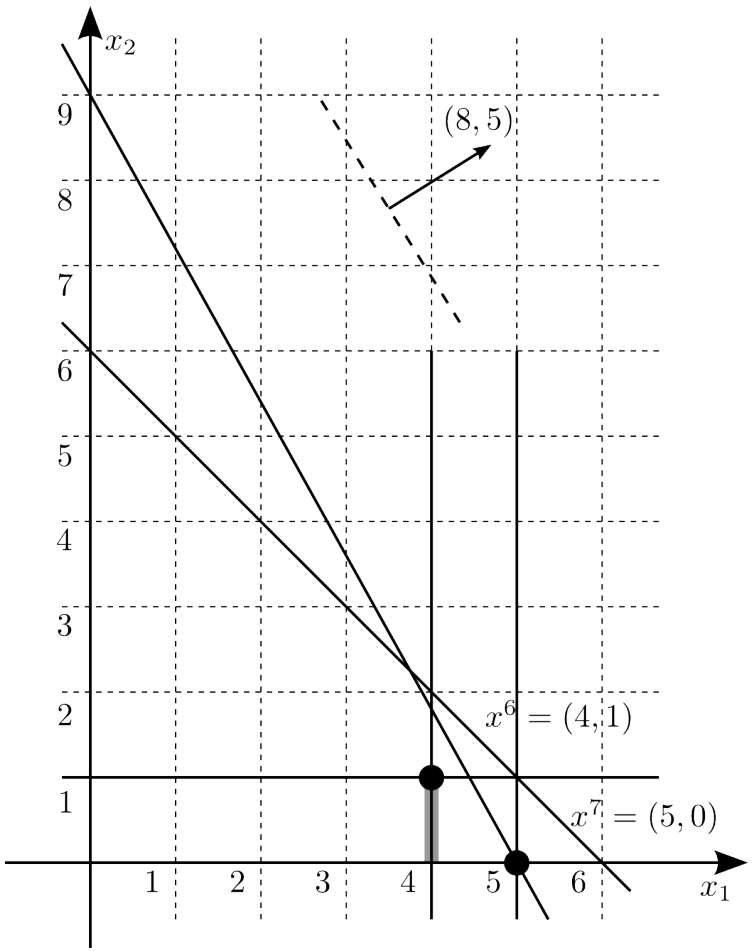


# Subproblems 6 and 7

- $x^6 = (4, 1)$ but $z_6 = 37 < 39 = z_2$.
- $x^7 = (5, 0)$ and $z_7 = 40 > 39 = z_2$. As it is also the last node, $x^7$ is an optimal solution.



# Remarks

- To select a node to branch:
  - Among all alive nodes, there are many different ways of selecting a node to branch.
  - One common approach is to branch the node with the highest objective value (for a maximization problem). Why?
  - Another popular approach is "once a node is branched, all its descendants are branched before any nondescendant. Why? 因为我们希望至少找到一个可行解来bound后续的branch
- Choosing a variable to branch on is also a challenging task.
- The branch-and-bound algorithm guarantees to find an optimal solution, if one exists.
- However, it is an **exponential-time** algorithm.
  - Roughly speaking, with $n$ integer variables, the number of subproblems solved is approximately proportional to $2^n$.

# Road map

- Linear relaxation.
- The branch-and-bound algorithm.
- **Solving the knapsack problem**.
- Heuristic algorithms.

# The knapsack problem

- We start our illustration with the classic **knapsack** problem.
- There are five items to select:

| Item | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Value ($) | 2 | 3 | 4 | 1 | 3 |
| Weight (kg) | 4 | 5 | 3 | 1 | 4 |

- The knapsack capacity is 11 kg.
- We maximize the total value without exceeding the knapsack capacity.

# The knapsack problem

- The complete formulation:

$$\begin{array}{rrcrcrcrcrcl} \max & 2x_1 & + & 3x_2 & + & 4x_3 & + & x_4 & + & 3x_5 & \\ \text{s.t.} & 4x_1 & + & 5x_2 & + & 3x_3 & + & x_4 & + & 4x_5 & \leq & 11 \end{array}$$
$$x_i \in \{0,1\} \quad \forall i = 1, ..., 5.$$

- Let's solve the knapsack problem with the branch-and-bound algorithm.

# Solving the linear relaxation

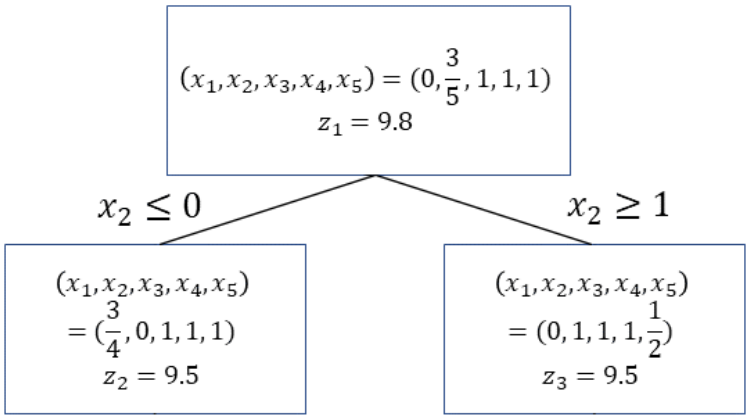
- The first step is always to solve the linear relaxation:

$$\begin{array}{rrcrcrcrcrcl} \max & 2x_1 & + & 3x_2 & + & 4x_3 & + & x_4 & + & 3x_5 & \\ \text{s.t.} & 4x_1 & + & 5x_2 & + & 3x_3 & + & x_4 & + & 4x_5 & \leq & 11 \\ & x_i \in [0,1] & \forall i = 1,...,5. & & & & & & & & & \end{array}$$

- A very intuitive way works:
  - Sort all items in the **descending order of** $\frac{v_i}{w_i}$, where $v_i$ and $w_i$ are the value and weight of item $i$, respectively.
  - Select items one by one according to the order until the knapsack is full. Note that the last item may be **partially** selected.
- The five ratios are $\frac{2}{4}$, $\frac{3}{5}$, $\frac{4}{3}$, $\frac{1}{1}$, and $\frac{3}{4}$. The order is item 3, item 4, item 5, item 2, and lastly item 1.
  - An optimal solution to the linear relaxation is $(0, \frac{3}{5}, 1, 1, 1)$.

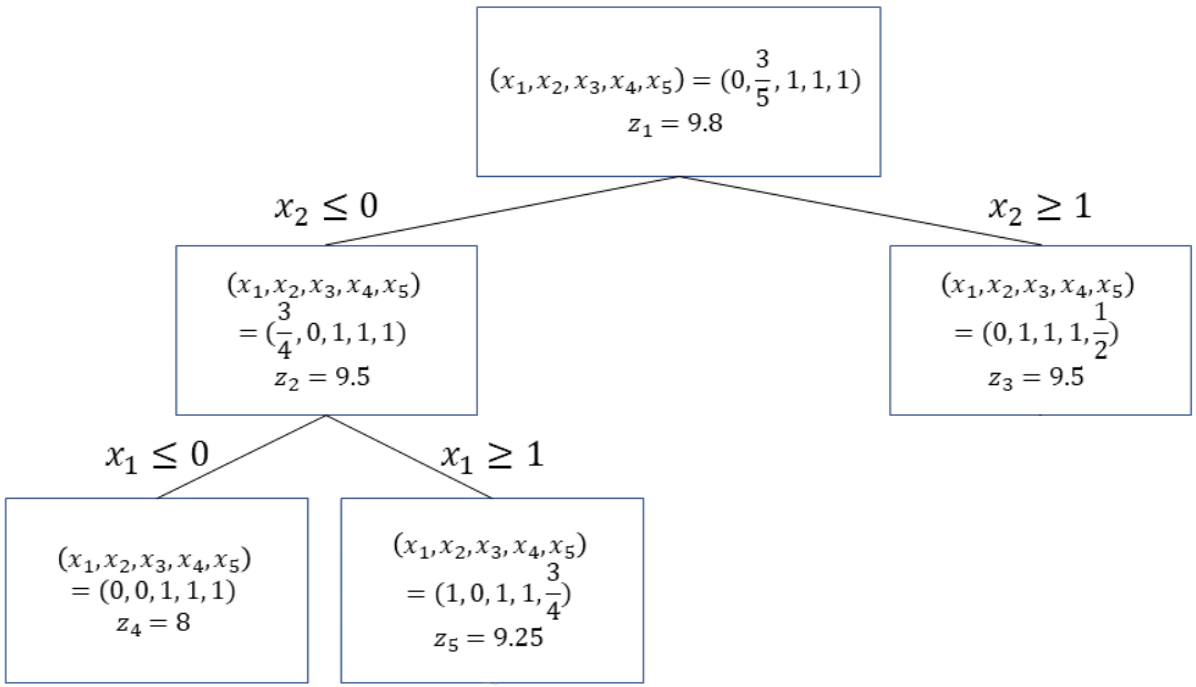
# Subproblems 1, 2, and 3

- We solve the linear relaxation and get $x^1 = (0, 0.6, 1, 1, 1)$ with objective value 9.8.
- We branch on $x_2$.
- With $x_2 \leq 0$: $x^2 = (0.75, 0, 1, 1, 1)$ and $z_2 = 9.5$.
  - We set $x_2 = 0$ and then apply the sorting idea on the remaining.
- With $x_2 \geq 1$: $x^3 = (0, 1, 1, 1, 0.5)$ and $z_3 = 9.5$.
  - We set $x_2 = 1$ and then apply the sorting idea on the remaining four variables and **residual capacity** 6.



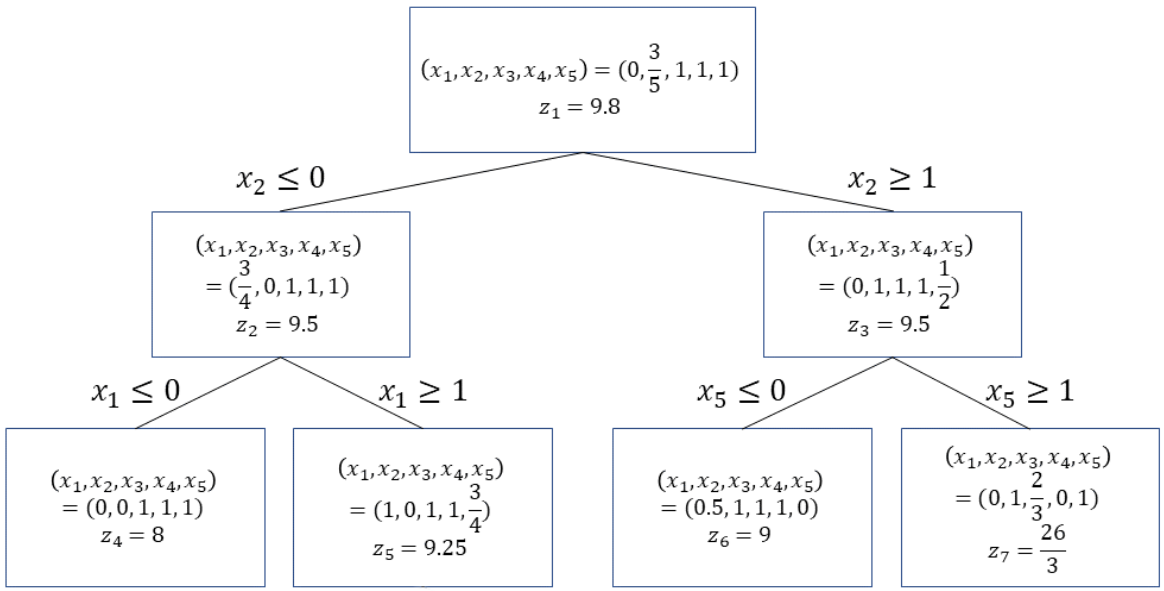
# Subproblems 4 and 5

- We branch the node for $x^2$.
- With $x_2 \leq 0$ and $x_1 \leq 0$: $x^4 = (0, 0, 1, 1, 1)$ and $z_4 = 8$. $x^4$ is the first candidate solution.
- With $x_2 \leq 0$ and $x_1 \geq 1$: $x^5 = (1, 0, 1, 1, \frac{3}{4})$ with $z_5 = 9.25$.



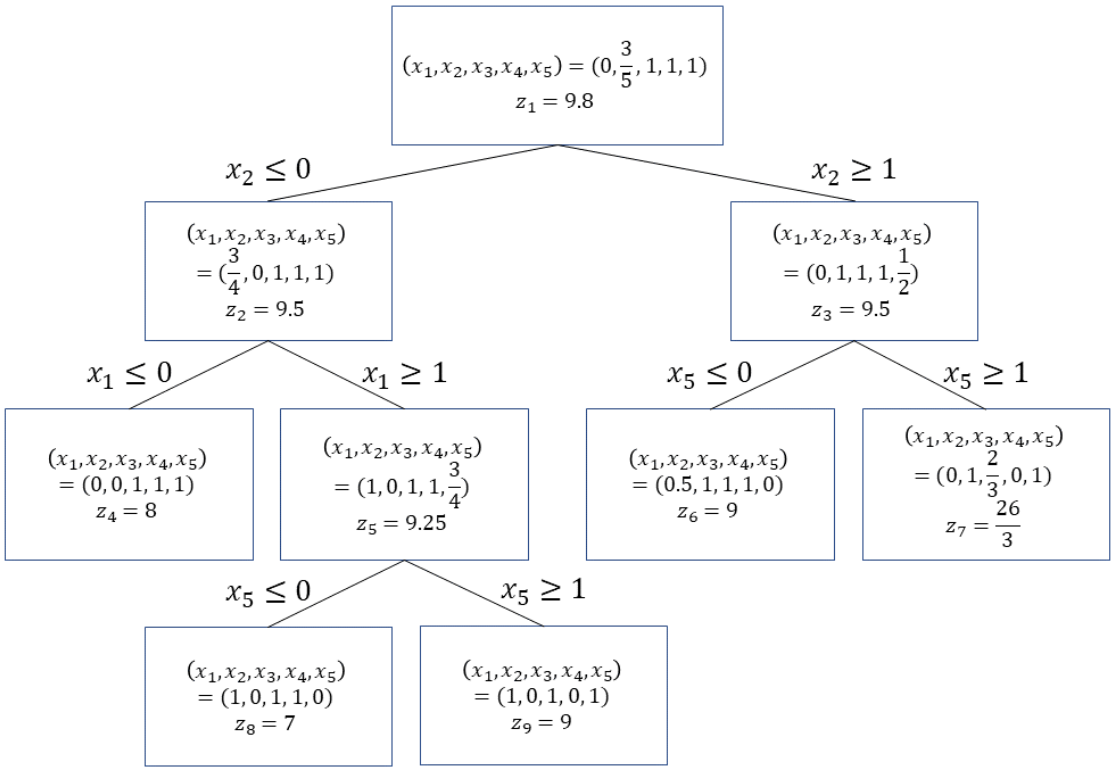
# Subproblems 6 and 7

- We branch the node for $x^3$.
- With $x_2 \geq 1$ and $x_5 \leq 0$: $x^6 = (0.5, 1, 1, 1, 0)$ and $z_6 = 9$.
- With $x_2 \geq 1$ and $x_5 \geq 1$: $x^7 = (0, 1, \frac{2}{3}, 0, 1)$ and $z_7 = \frac{26}{3}$.



# Subproblems 8 and 9

- With $x_2 \leq 0$, $x_1 \geq 1$, $x_5 \leq 0$: $x^8 = (1, 0, 1, 1, 0)$ and $z_8 = 7$.
- With $x_2 \leq 0$, $x_1 \geq 1$, $x_5 \geq 1$: $x^9 = (1, 0, 1, 0, 1)$ and $z_9 = 9$. $x^9$ becomes the currently best candidate.
- No need to branch the nodes for $x^6$ and $x^7$ (why?).
- $x^9$ is optimal.



# Remarks

- The major process is still branch and bound.
- The only unique thing is that we have a **special way** to solve the linear relaxation of the knapsack problem.
- In other words, we may **customize** the branch-and-bound algorithm for each specific type of IP.

# Road map

- Linear relaxation.
- The branch-and-bound algorithm.
- Solving the knapsack problem.
- **Heuristic algorithms**.

# The knapsack problem

- Recall our knapsack example with five items to select and a knapsack with capacity 11 kg:

| Item | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Value ($) | 2 | 3 | 4 | 1 | 3 |
| Weight (kg) | 4 | 5 | 3 | 1 | 4 |

- The formulation is

$$\begin{array}{rrcrcrcrcrcl} \max & 2x_1 & + & 3x_2 & + & 4x_3 & + & x_4 & + & 3x_5 & & \\ \text{s.t.} & 4x_1 & + & 5x_2 & + & 3x_3 & + & x_4 & + & 4x_5 & \leq & 11 \end{array}$$
$$x_i \in \{0,1\} \quad \forall i = 1, ..., 5.$$

- An optimal solution $x^* = (1, 0, 1, 0, 1)$ may be found by branch and bound. The associated objective value is $z^* = 9$.

# Exact and heuristic algorithms

- Unfortunately, the knapsack problem is **NP-hard**.
  - Roughly speaking, most researchers in the world believe that there is no efficient (i.e., polynomial-time) **exact algorithm** that finds an optimal solution.
  - Branch and bound may need too much time to solve an instance of a knapsack problem with thousands of variables.
- In this case, people look for good **heuristic algorithms**.
  - The algorithm reports a solution in a short (i.e., **polynomial**) time.
  - The reported solution is **near-optimal**.

# A greedy algorithm

- Our intuitive **greedy** algorithm may serve as a good one.
  - First, we sort all items by their value-to-weight ratios from large to small.
  - Along the order, we then try to put each item into the knapsack. If possible, do it; otherwise, throw it away.
  - We stop when all items are tried once.
- Recall our example:

| Item | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Value ($) | 2 | 3 | 4 | 1 | 3 |
| Weight (kg) | 4 | 5 | 3 | 1 | 4 |

  - The order: items 3, 4, 5, 2, and 1.
  - Items 3, 4, and 5 are okay. Items 2 and 1 are not.
  - The reported solution is $x^{\text{ALG}} = (0, 0, 1, 1, 1)$. The associated objective value is $z^{\text{ALG}} = 8$.

# Optimality gap

- The reported solution is $x^{\text{ALG}} = (0, 0, 1, 1, 1)$ with $z^{\text{ALG}} = 8$.
- An optimal solution is $x^* = (1, 0, 1, 0, 1)$ with $z^* = 9$.
- A natural measurement of the quality/performance of a reported solution is the **optimality gap**:
  - The **absolute error** is
  $$z^* - z^{\text{ALG}} = 1.$$
  - The **percentage error** is
  $$\frac{z^* - z^{\text{ALG}}}{z^*} = 11.1\%.$$

# Performance evaluation

- To test the **average-case performance** of a heuristic algorithm:
  - **Small-scale random instances** are generated.
  - Each instance is solved by the algorithm and an exact one.
  - All the **percentage errors** are averaged.
  - The smaller the average percentage error, the better the algorithm.
- Moreover:
  - **Large-scale** random instances are generated.
  - Each instance is solved by the algorithm.
  - An **upper bound** of the objective value of an optimal solution (for a maximization problem) is obtained (e.g., by linear relaxation).
  - The smaller the average percentage error (which is an **upper bound** for that calculated with an optimal solution), the better the algorithm.
- In the above example:
  - Linear relaxation gives $x^{\mathrm{LR}} = (0, 0.6, 1, 1, 1)$ with $z^{\mathrm{LR}} = 9.8$.
  - The errors are $z^{\mathrm{LR}} - z^{\mathrm{ALG}} = 1.8$ and $\frac{z^{\mathrm{LR}} - z^{\mathrm{ALG}}}{z^{\mathrm{LR}}} = 18.4\%$.

# Remarks

- Some people refer to the gap between a reported solution and an upper bound (rather than an optimal solution) as the optimality gap.
- Some people analyze the problem and heuristic algorithm to prove a **worst-case performance guarantee**.
  - Fascinating but difficult.
- Designing (or proposing) a heuristic algorithm is simple. **Evaluation** and **analysis** are hard.
- Thanks to integer programming, branch and bound, and linear relaxation:
  - We may generate an **optimal solution**.
  - We may use an optimal solution or its **upper/lower bound** as a **benchmark** to evaluate a feasible solution.
  - We may thus evaluate a heuristic algorithm.