15.053/8 April 2, 2013

IP Techniques 1. Branch and Bound

Quotes of the Day

“The time to relax is when you don't have time for it.”
-- Attributed to Jim Goodwin and Sydney J. Harris

“There is more to life than increasing its speed.”
-- Mohandas K. Gandhi

Overview

- **Enumerating all solutions is too slow for most problems.**
- **Branch and bound (B & B) starts the same as enumerating, but it cuts out a lot of the enumeration whenever possible.**
- **B & B is the starting point for all solution techniques for integer programming.**

Overview of this lecture

- Complete Enumeration
- How to compute a bound
- The branch and bound algorithm

Trading for Profit Game

Prize	iPad 1	server 2	Brass Rat 3	Au Bon Pain 4	6.041 tutoring 5	15.053 dinner 6
Points	5	7	4	3	4	6
Utility	16	22	12	8	11	19

Budget: 14 IHTFP points.

IP(1)

$$\text{maximize} \quad 16x_1 + 22x_2 + 12x_3 + 8x_4 + 11x_5 + 19x_6$$

$$\text{subject to} \quad 5x_1 + 7x_2 + 4x_3 + 3x_4 + 4x_5 + 6x_6 \leq 14$$

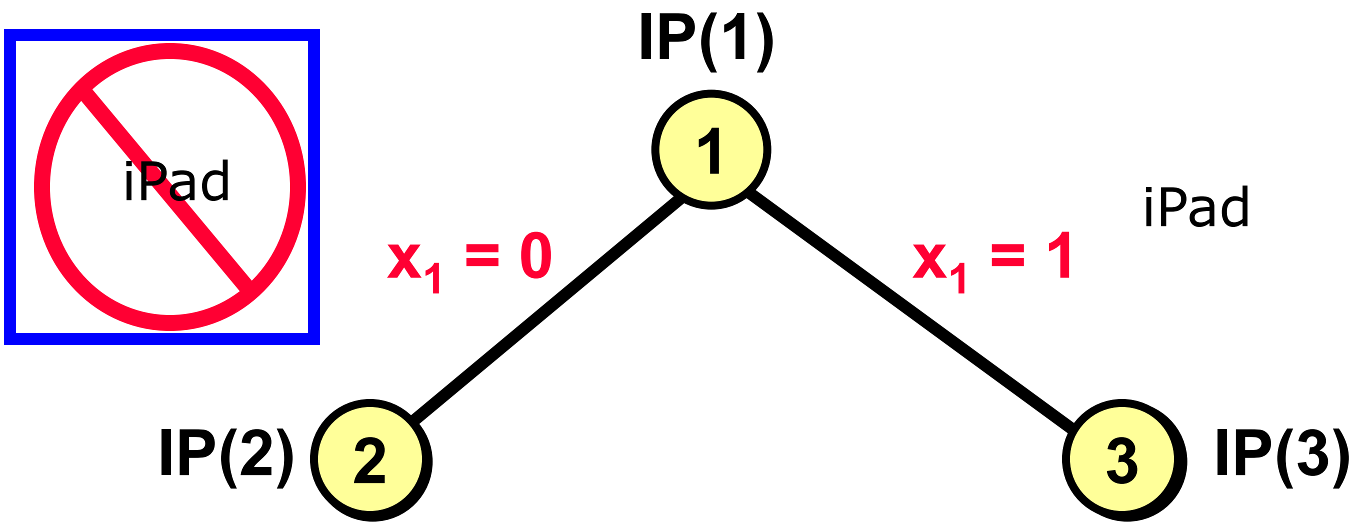
$$x_j \text{ binary for } j = 1 \text{ to } 6$$

Complete Enumeration

- Systematically considers all possible values of the decision variables.
  - If there are n binary variables, there are 2^n different ways.
- Usual idea: iteratively break the problem in two. At the first iteration, we consider separately the case that $x_1 = 0$ and $x_1 = 1$.
- Each node of the tree represents the original problem plus additional constraints.

An Enumeration Tree

An Enumeration Tree

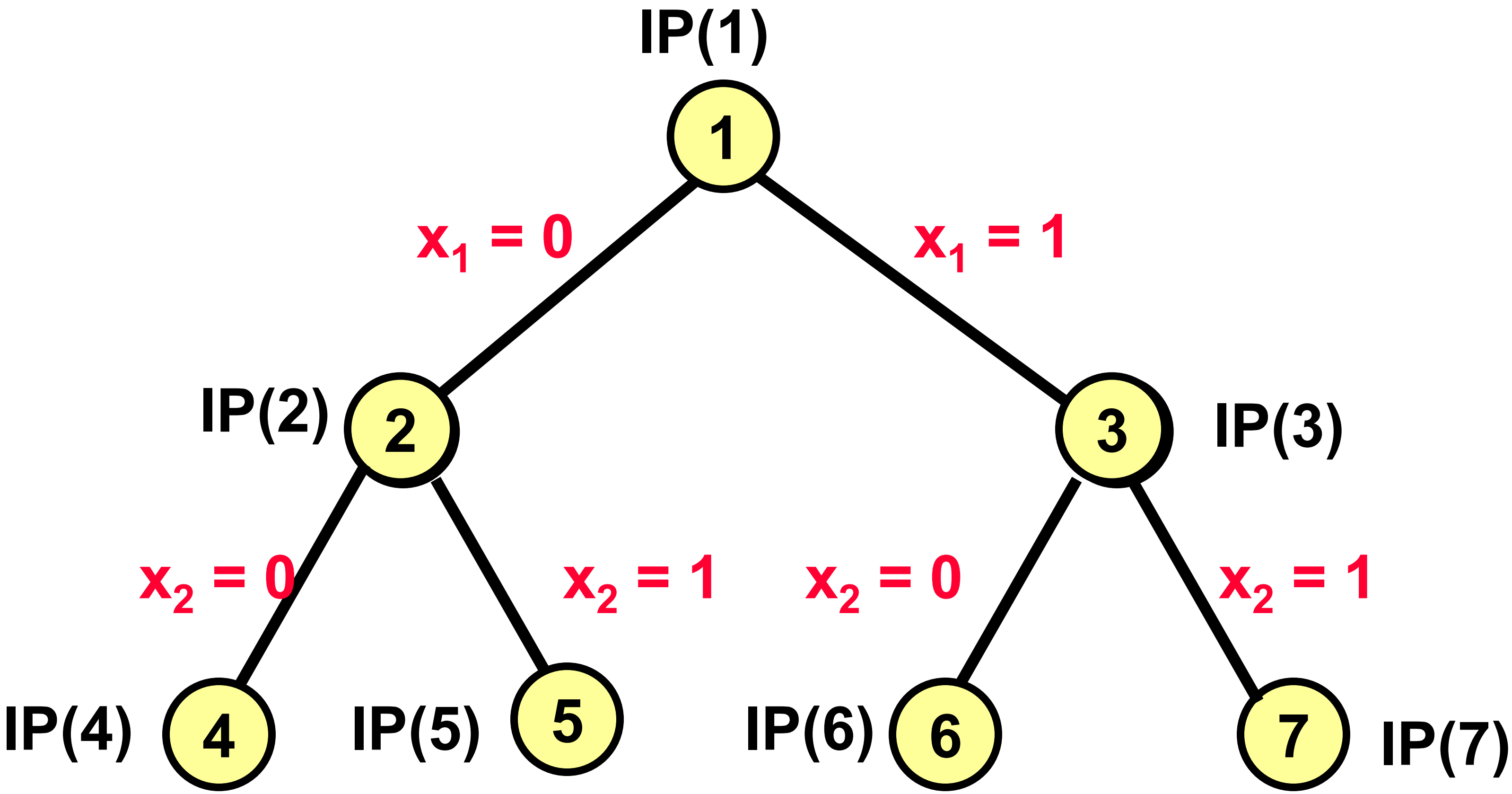


We refer to nodes 2 and 3 as the children of node 1 in the enumeration tree. We refer to node 1 as the parent of nodes 2 and 3.

Branch and bound is family friendly -- so long as you don't mind "pruning" children.

Which of the following is false?

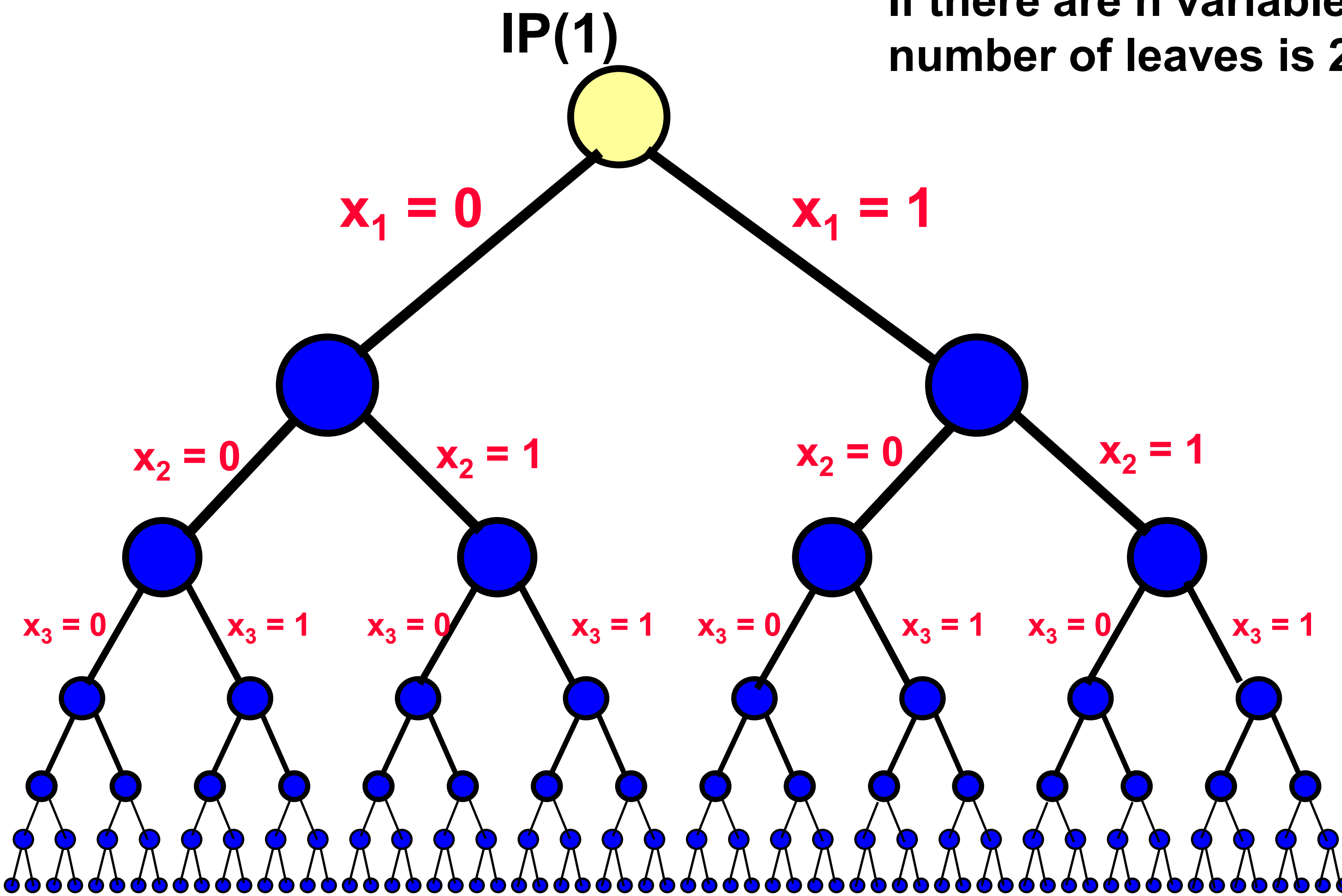
1. IP(1) is the original integer program.
2. IP(3) is obtained from IP(1) by adding the constraint "$x_1 = 1$".
3. An optimal solution for IP(1) can be obtained by taking the best solution from IP(2) and IP(3).
4. It is possible that there is some solution that is feasible for both IP(2) and IP(3).


IP(1)
1
$x_1 = 0$
$x_1 = 1$
IP(2)
2
3
IP(3)
$x_2 = 0$
$x_2 = 1$
$x_2 = 0$
$x_2 = 1$
IP(4)
4
IP(5)
5
IP(6)
6
7
IP(7)

An Enumeration Tree

Number of leaves of the tree: 64.

If there are n variables, the number of leaves is 2^n.



On complete enumeration

- Suppose that we could evaluate 1 billion solutions per second.
- Let n = number of binary variables
- Solutions times
  - n = 30, 1 second
  - n = 40, 17 minutes
  - n = 50 11.6 days
  - n = 60 31 years
  - n = 70 31,000 years

On complete enumeration

- Suppose that we could evaluate 1 trillion solutions per second, and instantaneously eliminate 99.9999999% of all solutions as not worth considering
- Let n = number of binary variables
- Solutions times
  - n = 70, 1 second
  - n = 80, 17 minutes
  - n = 90 11.6 days
  - n = 100 31 years
  - n = 110 31,000 years

Suppose that the number of binary variables is 150.

Suppose that we could evaluate 1 trillion solutions of an integer program per second.

Which of the following is false?

1. **Complete enumeration would take more than 1000 years.**
2. **We couldn't even solve it in 1000 years if we only had to enumerate 0.000000001 of the solutions.**
3. **No matter what algorithm we use for this problem, it cannot be solved in less than 1000 years.**

How to solve large integer programs faster

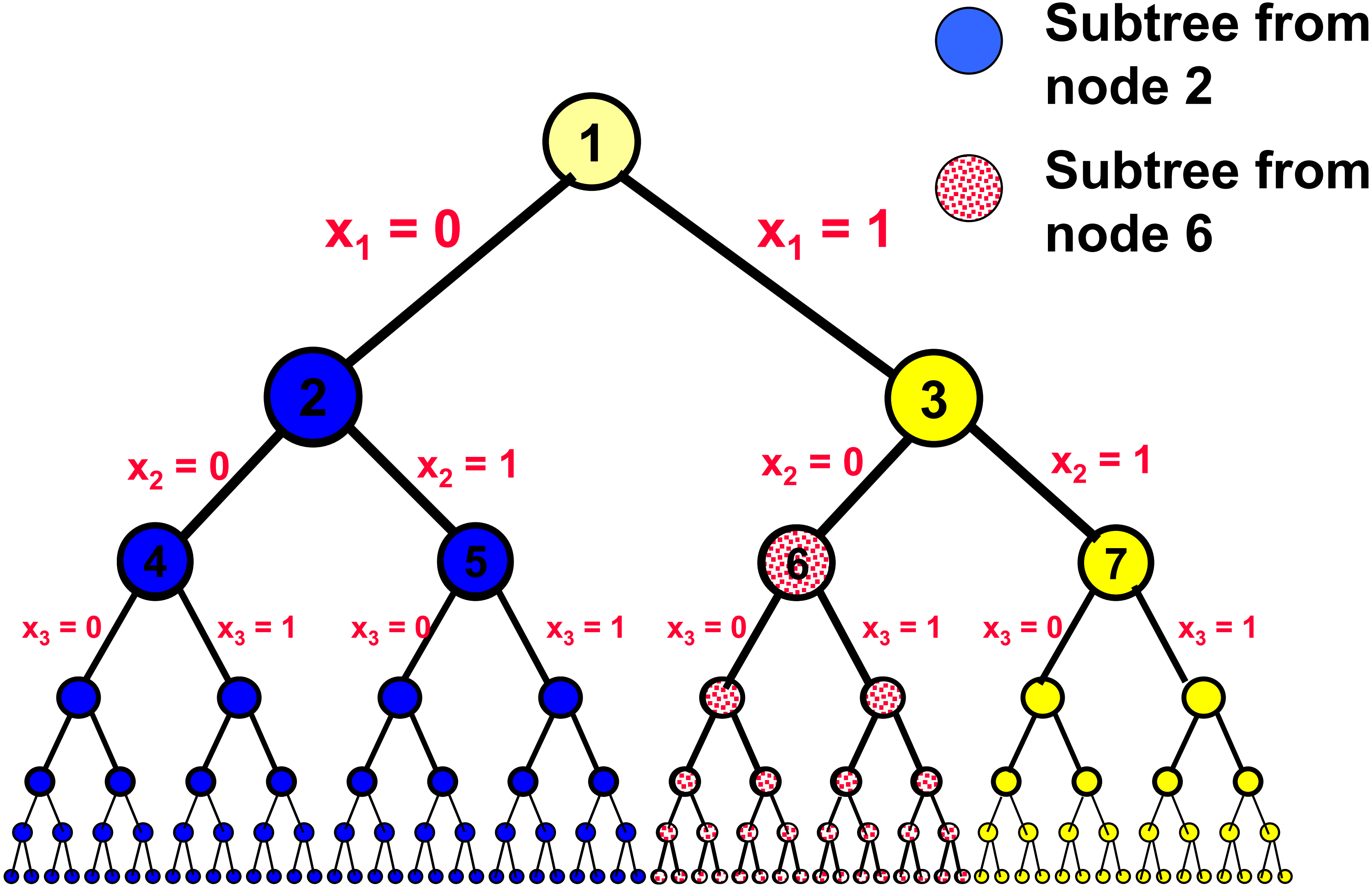
Eliminate much more than

99.99999999999999999999%

of the solutions without

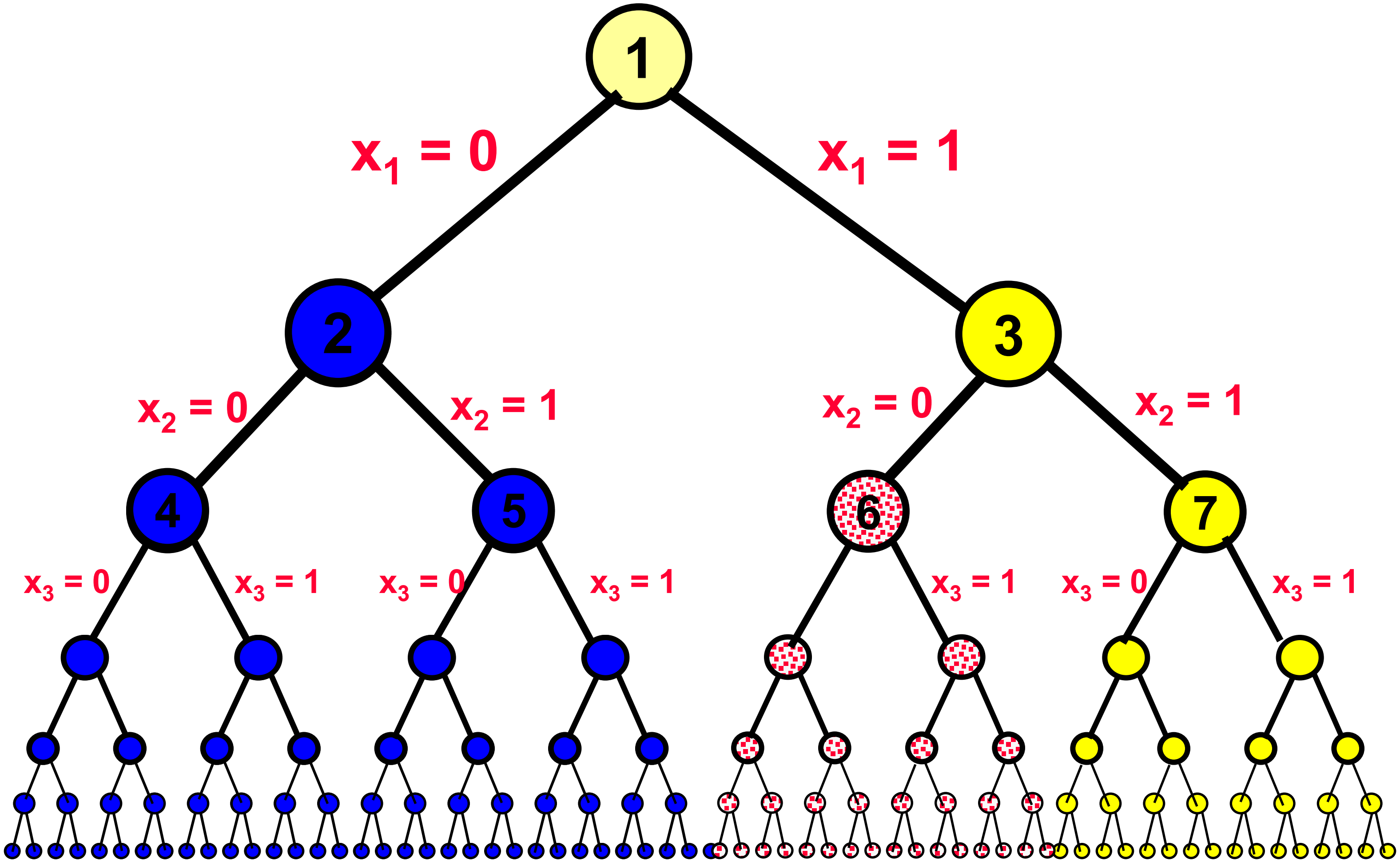
having to evaluate them.

Subtrees of an Enumeration Tree



The bottom nodes are *leaves* of the tree.

If we can eliminate an entire subtree in one step, we can eliminate a fraction of all complete solutions at in a single step.



A simpler problem to work with

Maximize $24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

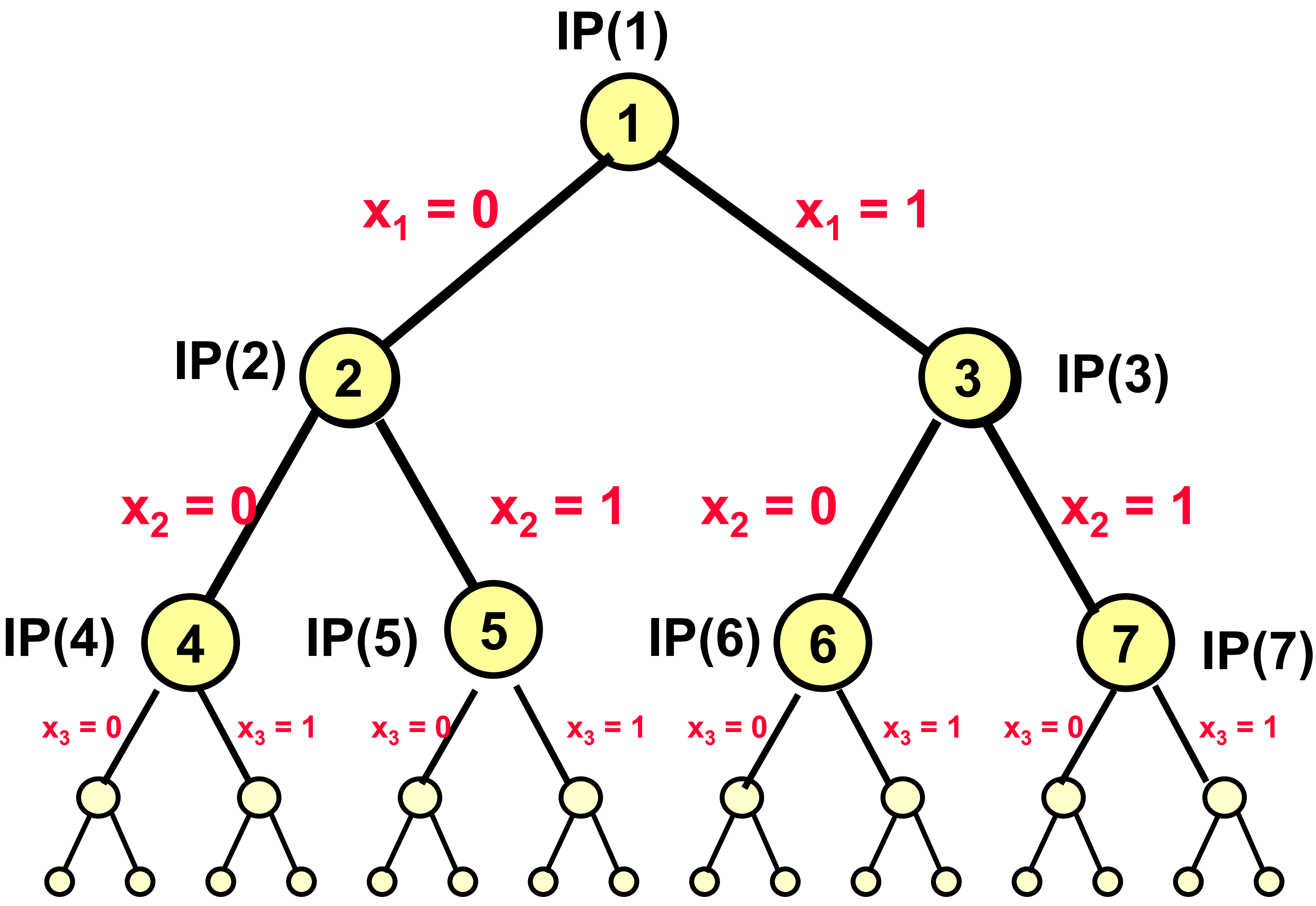
subject to $8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \leq 9$

$x_i \in \{0,1\}$ for $i = 1$ to 4.

IP(1)

This will be much easier to work with. I hope it's OK that we will be using IP(1) now to mean this 4-variable problem.

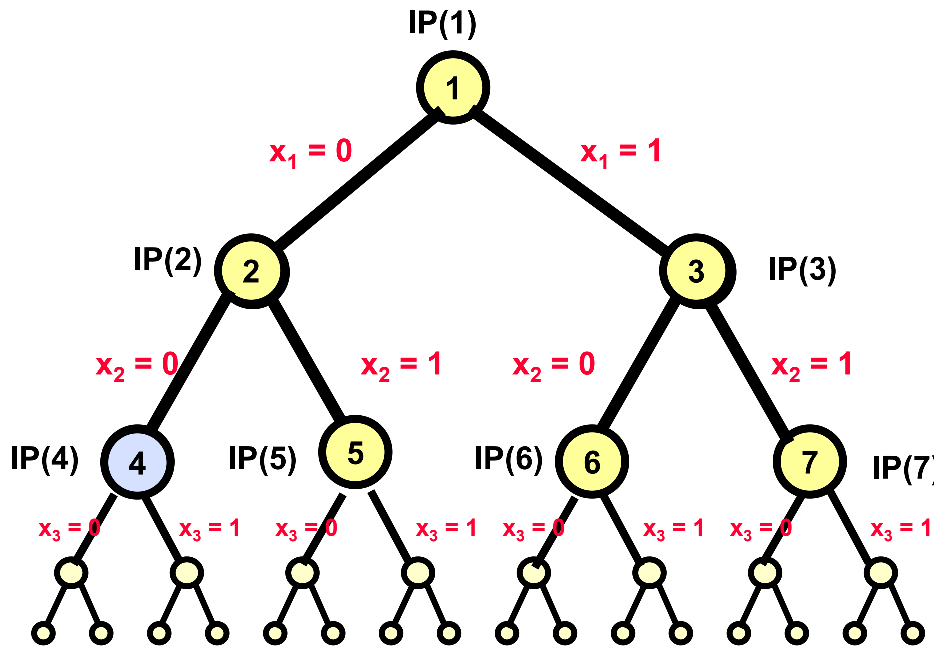
The entire enumeration tree (16 leaves)



The entire enumeration tree (16 leaves)

In a branch and bound tree, the nodes represent integer programs.

Each integer program is obtained from its ***parent node*** by adding an additional constraint.



For example, IP(4) is obtained from its parent node IP(2) by adding the constraint $x_2 = 0$.

What is the optimal objective value for IP(4)?

Maximize $24\ x_1 + 2\ x_2 + 20\ x_3 + 4\ x_4$

Original IP

subject to $8\ x_1 + 1\ x_2 + 5\ x_3 + 4\ x_4 \leq 9$

$x_i \in \{0,1\}$ for $i = 1$ to 4.

A. 24

B. 26

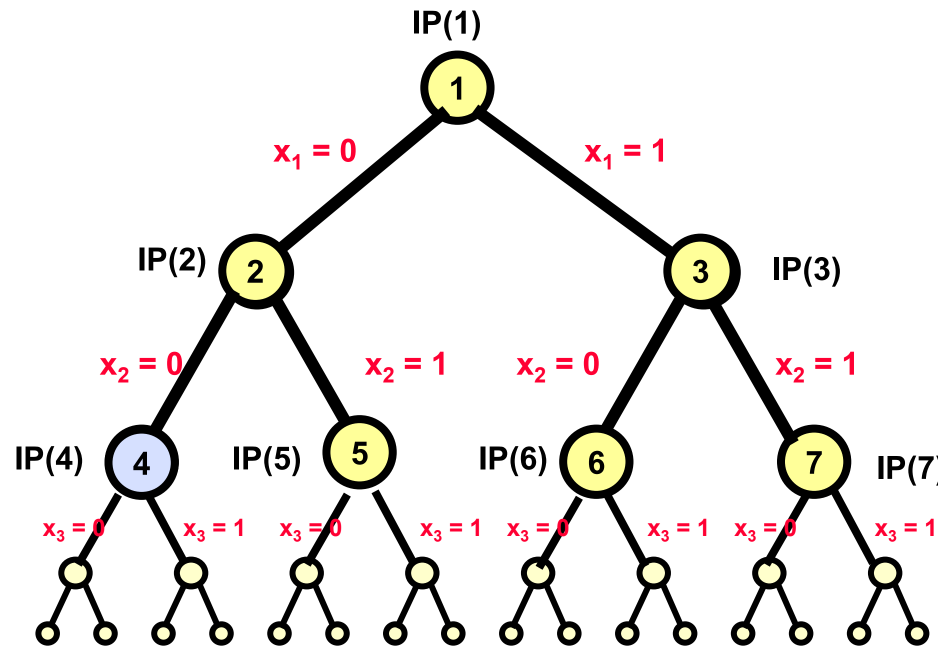
C. 9

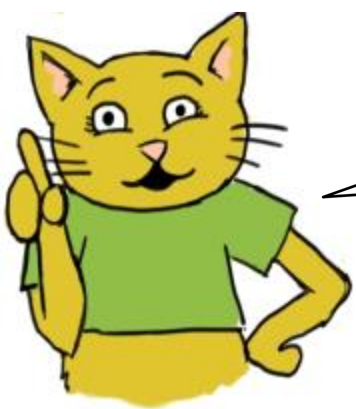
D. You didn't give me enough time to figure it out.

Eliminating subtrees

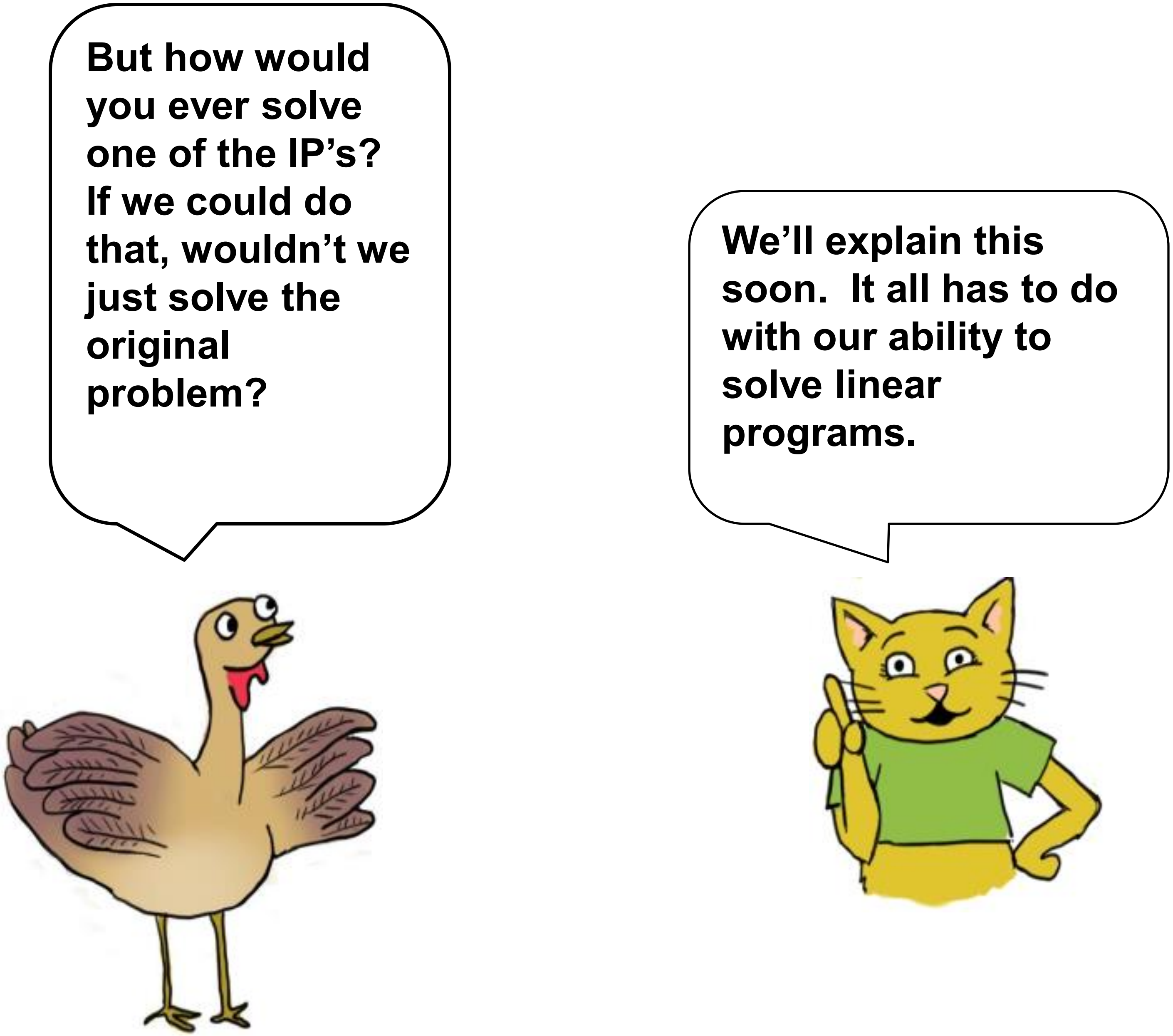
We eliminate a subtree if

1. We have **solved the IP** for the root of the subtree or
2. We have proved that the IP solution at the root of the subtree **cannot** be optimal.





After you solved IP(4), you don't need to look at its children.


But how would you ever solve one of the IP's? If we could do that, wouldn't we just solve the original problem?
We'll explain this soon. It all has to do with our ability to solve linear programs.

The LP Relaxation of the IP

$$\text{Maximize} \quad 24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$$

$$\text{subject to} \quad 8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \le 9 \qquad \text{LP(1)}$$

$$0 \le x_i \le 1 \quad \text{for } i = 1 \text{ to } 4.$$

If we drop the requirements that variables be integer, we call it the LP relaxation of the IP.

The LP relaxation of the knapsack problem can be solved using a "greedy algorithm."

Think of the objective in terms of dollars, and consider the constraint as a bound on the weight.

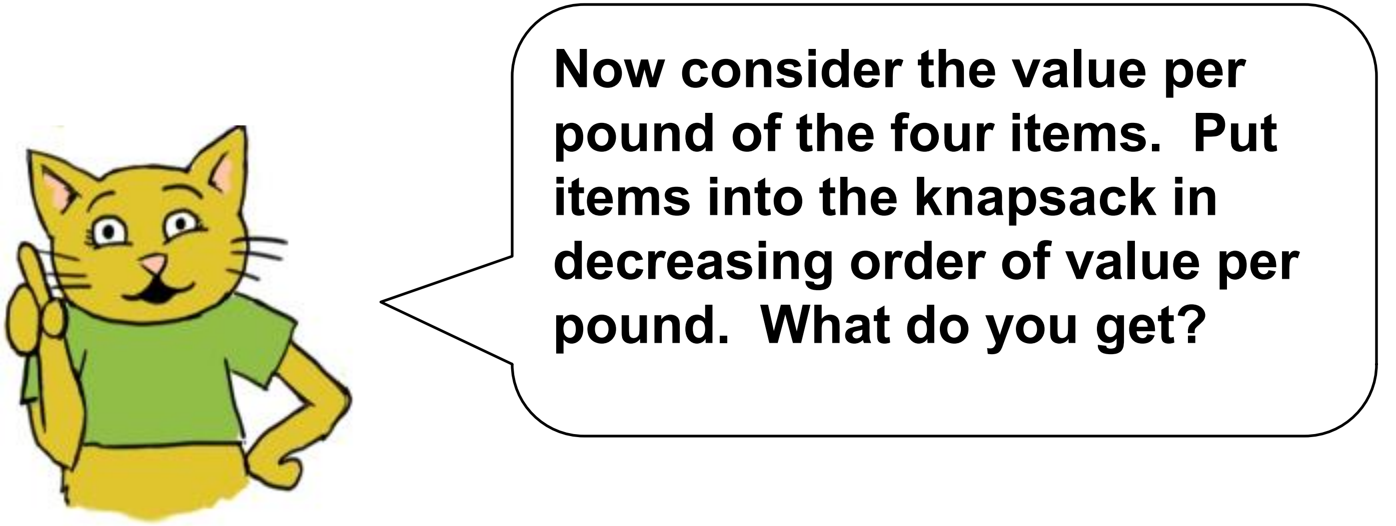
Solving this LP relaxation

Maximize $24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

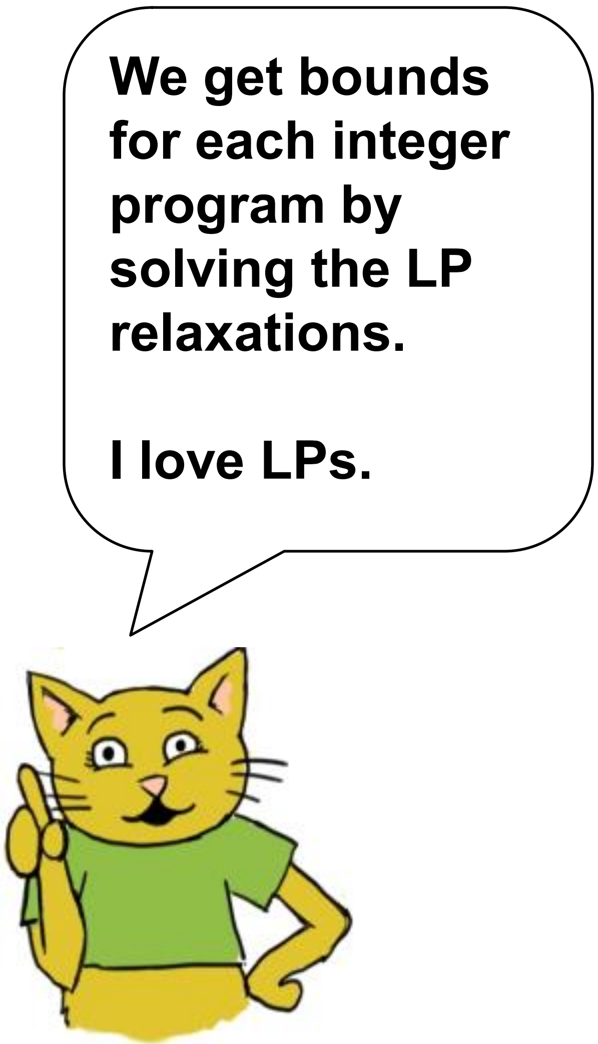
subject to $8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \leq 9$ **LP(1)**

$0 \leq x_i \leq 1$ for $i = 1$ to 4.

item	1	2	3	4
value/lb.	\$3	\$2	\$4	\$1



The LP relaxation of an IP



Maximize $24\ x_1 + 2\ x_2 + 20\ x_3 + 4\ x_4$

subject to $8\ x_1 + 1\ x_2 + 5\ x_3 + 4\ x_4 \leq 9$

IP(1) $x_i \in \{0,1\}$ for i = 1 to 4.

LP(j) = the integer programming relaxation of IP(j).

Maximize $24\ x_1 + 2\ x_2 + 20\ x_3 + 4\ x_4$

subject to $8\ x_1 + 1\ x_2 + 5\ x_3 + 4\ x_4 \leq 9$

LP(1) $0 \leq x_i \leq 1$ for i = 1 to 4.

This LP relaxation solves the IP.

Usually, when we solve the LP, we get fractional solutions. But occasionally, we get a solution that satisfies all of the integer constraints.

Maximize $24\ x_1 + 2\ x_2 + 20\ x_3 + 4\ x_4$

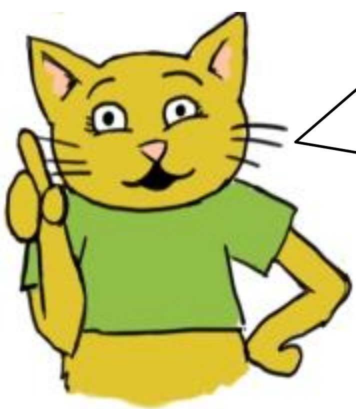
subject to $8\ x_1 + 1\ x_2 + 5\ x_3 + 4\ x_4 \leq 9$

LP(4) $x_1 = 0,\ x_2 = 0$ for i = 1 to 4.

$0 \leq x_3 \leq 1 \qquad 0 \leq x_4 \leq 1$

Opt solution for LP(4):

$x_1 = 0,\ x_2 = 0,\ x_3 = 1,\ x_4 = 1,\ z = 24.$



If the optimal solution for LP(k) is feasible for IP(k), then it is also optimal for IP(k).

In this example, the solution to LP(4) has z = 24 and the solution is feasible for the IP. There can't possibly be an IP solution for IP(4) with value better than 24.

This LP relaxation also solves the IP.

Maximize $24\ x_1 + 2\ x_2 + 20\ x_3 + 4\ x_4$

subject to $8\ x_1 + 1\ x_2 + 5\ x_3 + 4\ x_4 \leq 9$

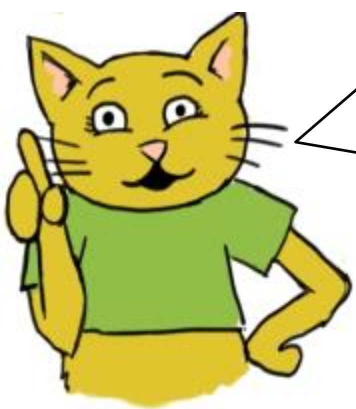
LP(15) $x_1 = 1,\ x_2 = 1,\ x_3 = 1$

$0 \leq x_4 \leq 1$

And occasionally, the LP relaxation is infeasible.

In this case, the IP is also infeasible.

There is no feasible solution for LP(15):



If LP(k) is infeasible, then IP(k) is infeasible.

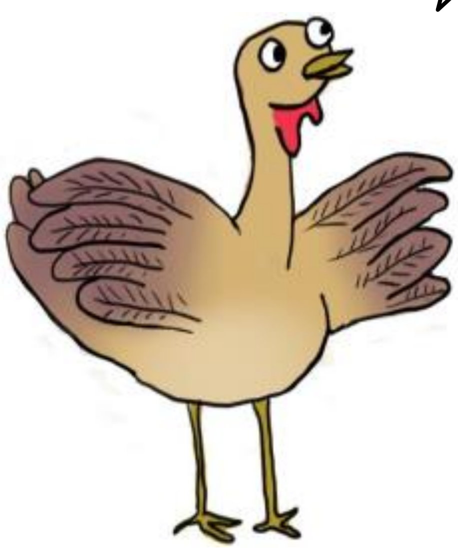
In this example, the LHS of the constraint is at least 13. There is no way that the constraint can be satisfied by fractional values or integer values of x_3 and x_4.

We eliminate a subtree if

1. We have solved the IP for the root of the subtree or
2. We have proved that the IP solution at the root of the subtree cannot be optimal.

I see that sometimes the IP gets solved, almost by accident.

Five slides ago you said that we could eliminate a node if we can prove that the optimal solution for the IP is not optimal for the original problem. How is that possible?



We'll explain this after the mental break.

Mental Break

The Incumbent Solution

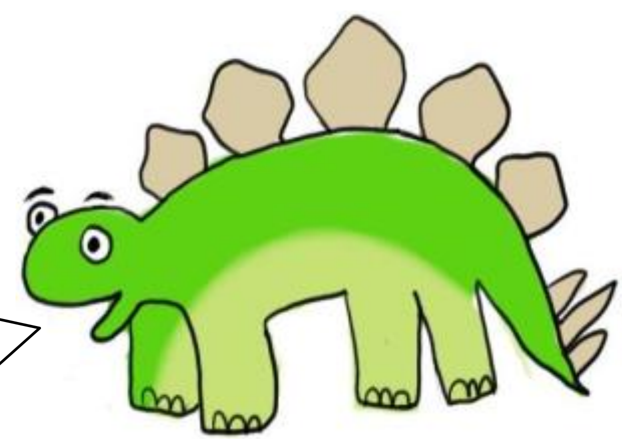
The incumbent is a feasible solution for the IP. It is the best solution so far in the B&B search.

Occasionally, the algorithm will find a feasible integer solution. We will keep track of the feasible integer solution with the best objective value so far. It is called the incumbent.

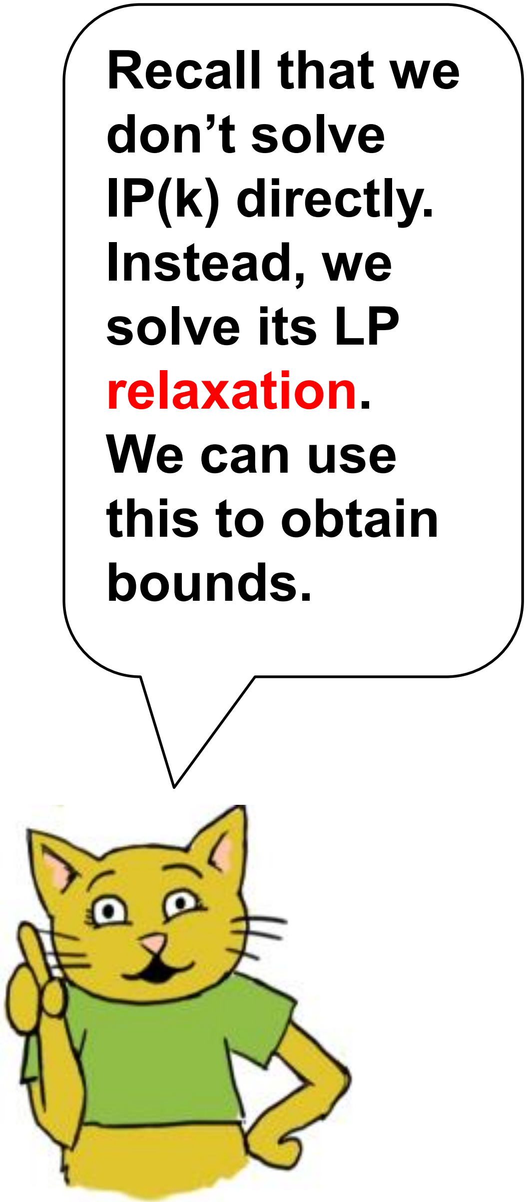
In the "vanilla" version of Branch and Bound, there is no initial incumbent. We need to wait until an LP relaxation gives a feasible integer solution.

In real versions of Branch and Bound, there are special subroutines that seek out feasible integer solutions with a large objective. The best of these is the initial incumbent.

Does Branch and Bound come in any other flavors? I prefer leafy flavors.



Bounds



Maximize $24\, x_1 + 2\, x_2 + 20\, x_3 + 4\, x_4$

subject to $8\, x_1 + 1\, x_2 + 5\, x_3 + 4\, x_4 \leq 9$

LP(1) $0 \leq x_i \leq 1$ for $i = 1$ to 4.

Opt solution for LP(1):

$x_1 = 1/2,\ x_2 = 0,\ x_3 = 1,\ x_4 = 0,\ z = 32$

$z_{IP}(j)$ = optimal value for IP(j).

$z_{LP}(j)$ = optimal value for LP(j).

$z_{LP}(1) = 32$

Note: $z_{IP}(1) \leq 32$.

On computing bounds

$z_{IP}(j)$ = optimal value for IP(j).

$z_{LP}(j)$ = optimal value for LP(j).

$x(j)$ = optimal solution for LP(j)

IP(1)

Maximize $24x_1 + 2x_2 + 20x_3 + 4x_4$

subject to $8x_1 + 1x_2 + 5x_3 + 4x_4 \le 9$

$x_i \in \{0,1\}$ for $i = 1$ to 4.

LP(1)

Maximize $24x_1 + 2x_2 + 20x_3 + 4x_4$

subject to $8x_1 + 1x_2 + 5x_3 + 4x_4 \le 9$

$0 \le x_i \le 1$ for $i = 1$ to 4.

We want to find $z_{IP}(1)$. But that's really hard. What's much easier is to determine $z_{LP}(j)$ for any j. We then rely an an important observation.

IMPORTANT OBSERVATION.

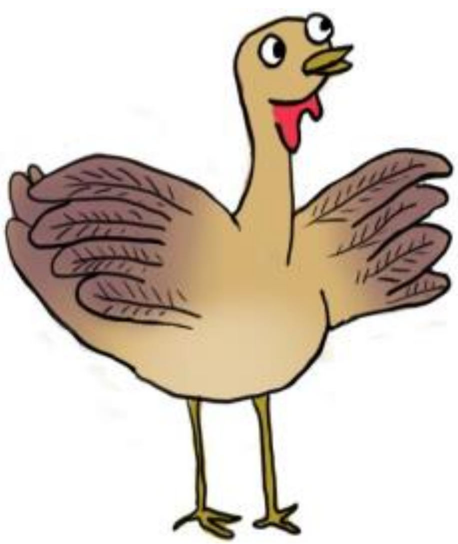
$z_{IP}(j) \le z_{LP}(j)$ for all j.

I'm sorry. But I think I zoned out for a minute. Have you answered my question from before the break? It was about eliminating subtrees from IP(k).

I'm just about to. We can **prune** the **active** node k IP(k) if

$$z_{LP}(k) \leq z_I,$$

where z_I is the objective value of the incumbent.



A node is **active** if it has not been pruned and if LP(k) has not been solved yet.

Pruning (fathoming) a node using bounding

Maximize $24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

subject to $8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \leq 9$

$x_1 = 0$

LP(2) $0 \leq x_j \leq 1$ for $j = 2, 3, 4$

Suppose that the incumbent is

$x_1 = 1, \quad x_2 = 1$

$x_3 = 0, \quad x_4 = 0$

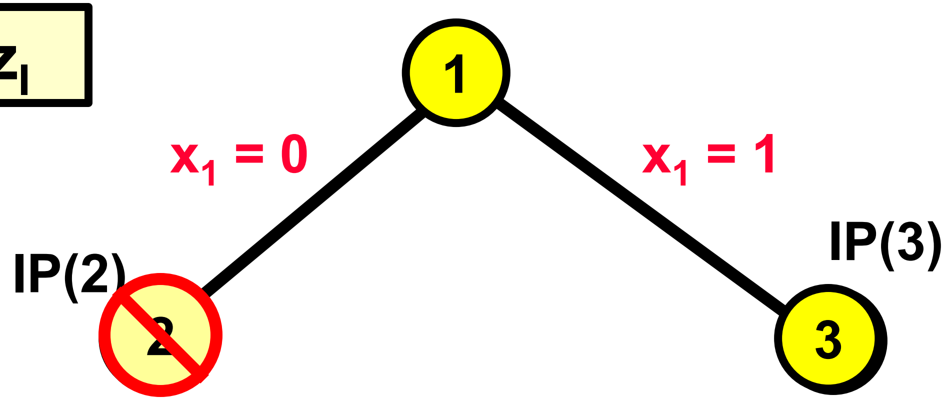
$z_I = 26$

Opt solution for LP(2) is:

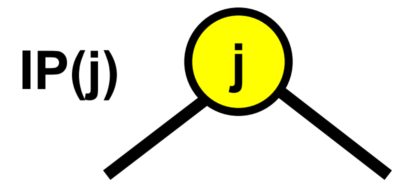
$x_1 = 0, \quad x_2 = 1, \quad x_3 = 1, \quad x_4 = 3/4$

$z_{LP}(2) = 25.$

Then $z_{IP}(2) \leq z_{LP}(2) = 25 < z_I$



Under which condition can we not prune active node j from the B&B Tree for a maximization problem?



1. LP(j) has no feasible solution.
2. LP(j) has a feasible solution, and $z_{LP}(j) > z_I$
3. LP(j) has a feasible solution, and $z_{LP}(j) < z_I$
4. The solution for LP(j) is feasible for the original integer program.

The branch and bound algorithm in one slide.

while there is some active nodes **do**

select an active node *j*

mark j as inactive

Solve LP(j): denote solution as x(j);

Case 1 -- **if** $z_{LP}(j) \le z_I$ **then** ***prune*** node j;

Case 2 -- **if** $z_{LP}(j) > z_I$ and
if x(j) is feasible for IP(j)

then Incumbent := x(j), and $z_I := z_{LP}(j)$;

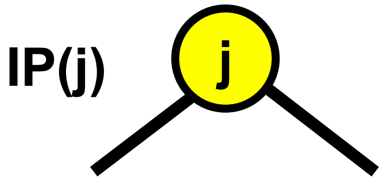
then ***prune*** node j;

Case 3 -- If **if** $z_{LP}(j) > z_I$ and
if x(j) is not feasible for IP(j) **then**

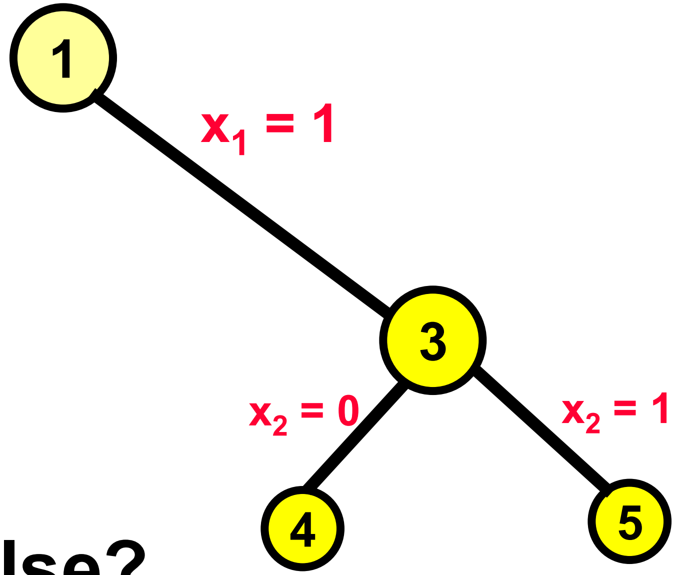
mark the children of node *j* as active

endwhile

z_I: incumbent obj. value



The following question is about **ANY** branch and bound tree in which node 3 was not pruned.



Which of the following is false?

1. $Z_{IP}(3) \geq Z_{IP}(4)$.
2. Every feasible solution for IP(5) is also a feasible solution for IP(3).
3. Every feasible solution for IP(3) is feasible for IP(4) and for IP(5)
4. Every feasible solution for IP(3) is feasible or for IP(4) or IP(5) but not both.

Branch and Bound: Node 1

Maximize $24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

subject to $8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \le 9$

$0 \le x_j \le 1$ for $j = 1, 2, 3, 4$

No incumbent

$z_I = -\infty$

IP(1)

Opt solution for LP(1):

$x_1 = 1/2$; $x_2 = 0$,
$x_3 = 1$; $x_4 = 0$

$z_{LP}(1) = 32$.

Branch and Bound: Node 2

Maximize $\quad 24\, x_1 + 2\, x_2 + 20\, x_3 + 4\, x_4$

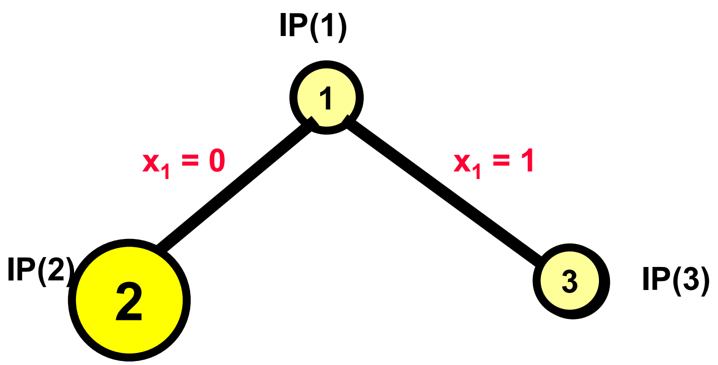
subject to $\quad 8\, x_1 + 1\, x_2 + 5\, x_3 + 4\, x_4 \le 9$

$x_1 = 0$

$0 \le x_j \le 1 \quad \text{for } j = 2, 3, 4$

No incumbent

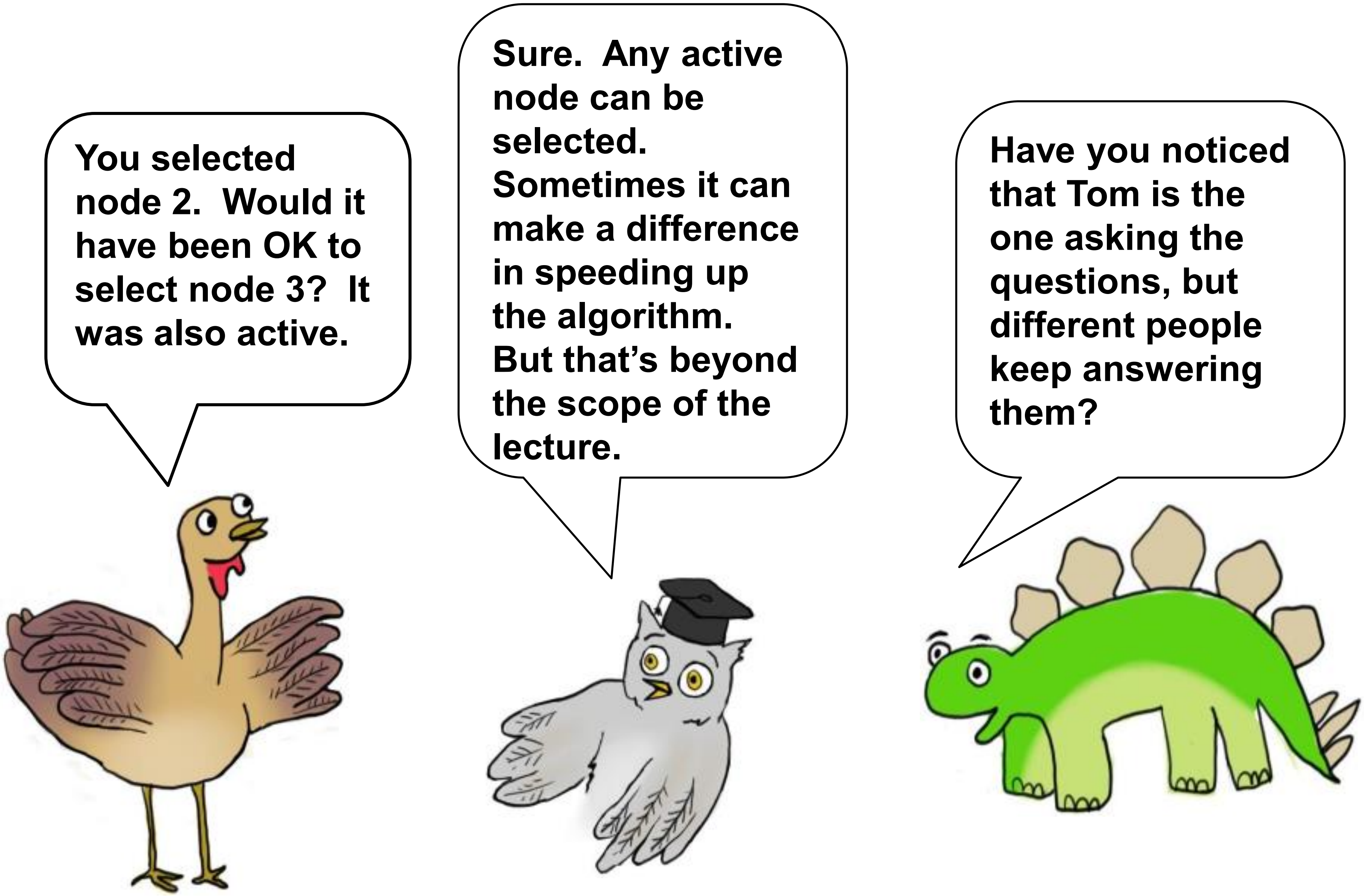
$z_I = -\infty$



Opt solution for LP(2):

$x_1 = 0; \quad x_2 = 1,$
$x_3 = 1; \quad x_4 = 3/4$

$z_{LP}(2) = 25.$


You selected node 2. Would it have been OK to select node 3? It was also active.
Sure. Any active node can be selected. Sometimes it can make a difference in speeding up the algorithm. But that’s beyond the scope of the lecture.
Have you noticed that Tom is the one asking the questions, but different people keep answering them?

Branch and Bound: Node 3

Maximize $24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

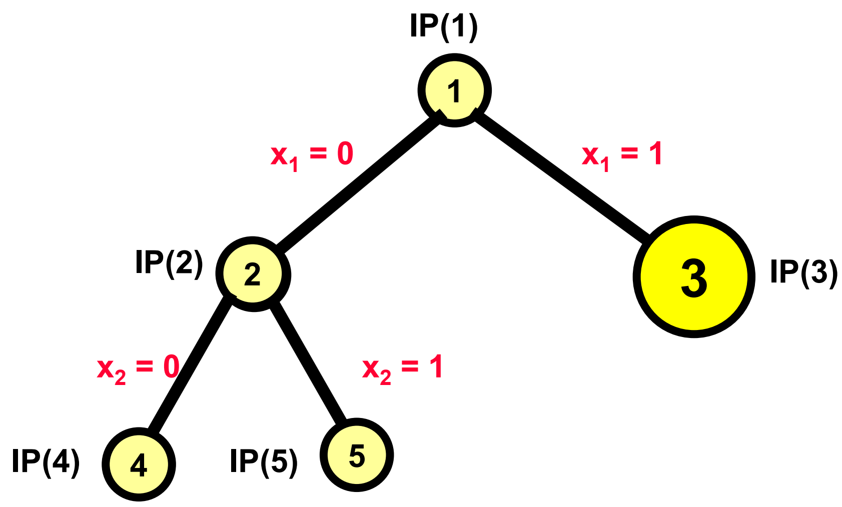
subject to $8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \le 9$

$x_1 = 1$

$0 \le x_j \le 1$ for $j = 2, 3, 4$

No incumbent

$z_I = -\infty$

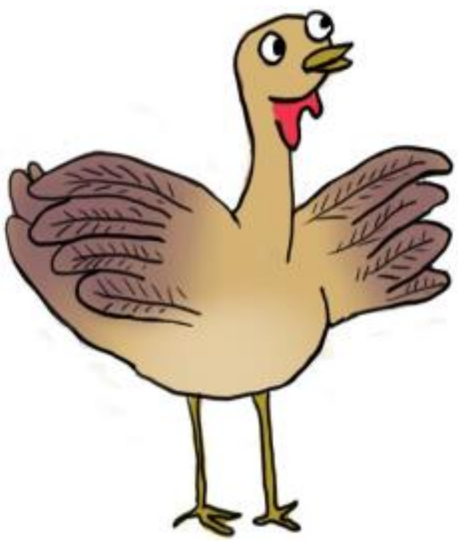


Opt solution for LP(3):

$x_1 = 1$; $x_2 = 0$,
$x_3 = 1/4$; $x_4 = 0$

$z_{LP}(3) = 28$.

I notice that when you create nodes 4 and 5, you “branch” on variable x_2. On one branch, we require $x_2 = 0$. On the other side, we require that $x_2 = 1$. Is that always the way that B&B works?



No. We could branch on any variable. If we branched on x_4, then node 4 would correspond to the original IP with the additional constraints:

$$x_1 = 0, x_4 = 0.$$

Branching makes a big difference in B&B. The best B&B algorithms use heuristics to choose the branching variable. A good choice can lead to a much faster solution.

Branch and Bound: Node 4

Maximize $24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

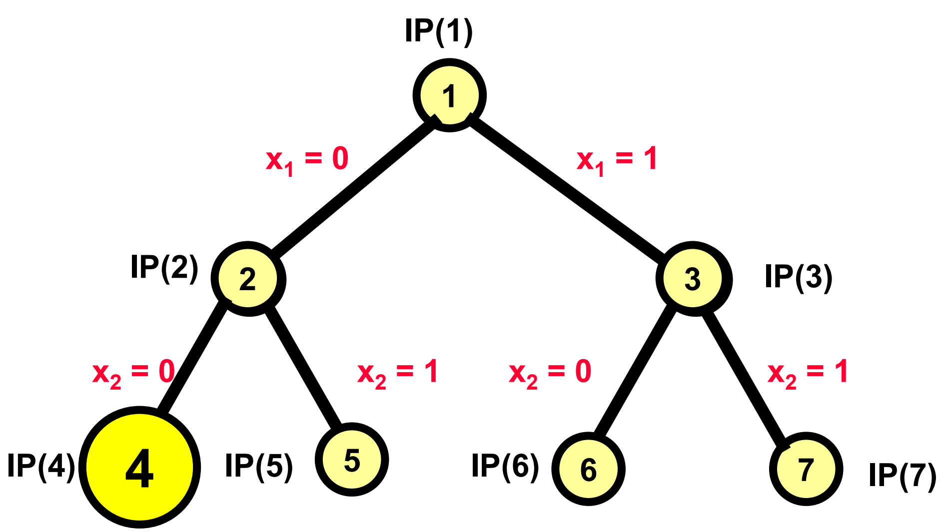
subject to $8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \le 9$

$x_1 = 0\ ,\ x_2 = 0$

$0 \le x_j \le 1$ for $j = 3, 4$

No incumbent

$z_I = -\infty$



Opt solution for LP(4):

$x_1 = 0$; $x_2 = 0$,
$x_3 = 1$; $x_4 = 1$

$z_{LP}(4) = 24.$

Branch and Bound: Node 5

Maximize $24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

subject to $8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \le 9$

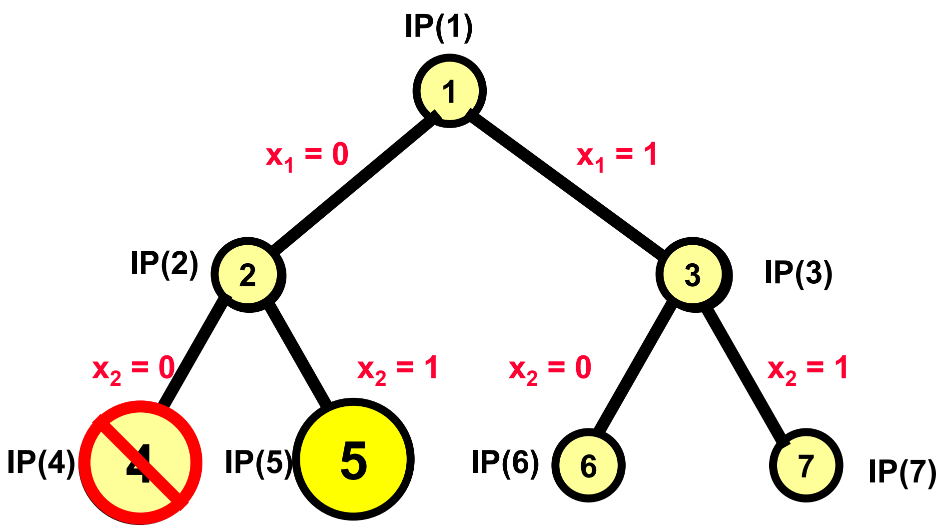
$x_1 = 0\ ,\ x_2 = 1$

$0 \le x_j \le 1 \quad$ for $j = 3, 4$

$x_1 = 0, \quad x_2 = 0,$
$x_3 = 1, \quad x_4 = 1$

$z_I = 24$

Incumbent



Opt solution for LP(5):

$x_1 = 0; \qquad x_2 = 1,$
$x_3 = 1; \qquad x_4 = 3/4$

$z_{LP}(5) = 25.$

Branch and Bound: Node 6

Maximize $\quad 24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

subject to $\quad 8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \le 9$

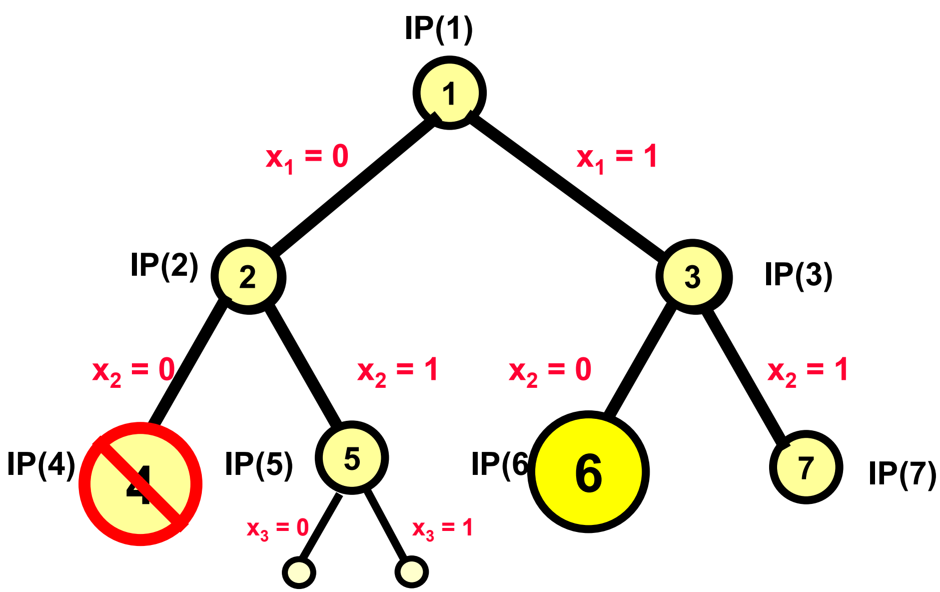
$$x_1 = 1\,,\ x_2 = 0$$

$$0 \le x_j \le 1 \quad \text{for } j = 3, 4$$

$x_1 = 0, \quad x_2 = 0,$
$x_3 = 1, \quad x_4 = 1$

$z_I = 24$

Incumbent



Opt solution for LP(6):

$x_1 = 1; \qquad x_2 = 0,$
$x_3 = 1/5; \qquad x_4 = 0$

$z_{LP}(6) = 28.$

Branch and Bound: Node 7

Maximize $24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

subject to $8\,x_1 + 1\,x_2 + \ 5\,x_3 + 4\,x_4 \le 9$

$x_1 = 1\ ,\ x_2 = 1$

$0 \le x_j \le 1 \quad \text{for } j = 3, 4$

$x_1 = 0, \quad x_2 = 0,$
$x_3 = 1, \quad x_4 = 1$

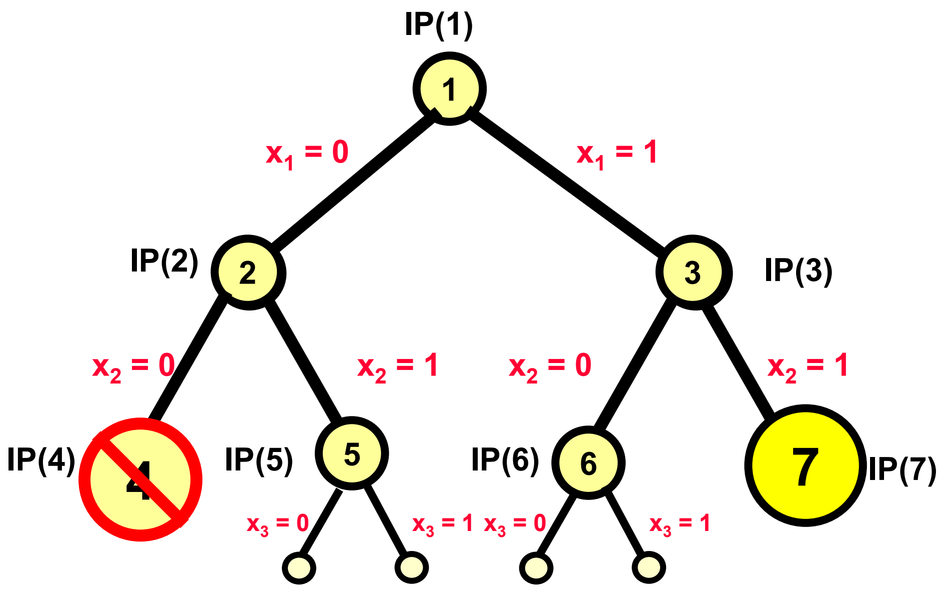
$z_I = 24$

Incumbent

Opt solution for LP(7):

$x_1 = 1; \qquad x_2 = 1,$
$x_3 = 0; \qquad x_4 = 0$

$z_{LP}(7) = 26$



Branch and Bound: Node 8

Maximize $24 x_1 + 2 x_2 + 20 x_3 + 4 x_4$

subject to $8 x_1 + 1 x_2 + 5 x_3 + 4 x_4 \leq 9$

$x_1 = 0 , x_2 = 1, x_3 = 0$

$0 \leq x_4 \leq 1$

$x_1 = 1, \quad x_2 = 1,$
$x_3 = 0, \quad x_4 = 0$

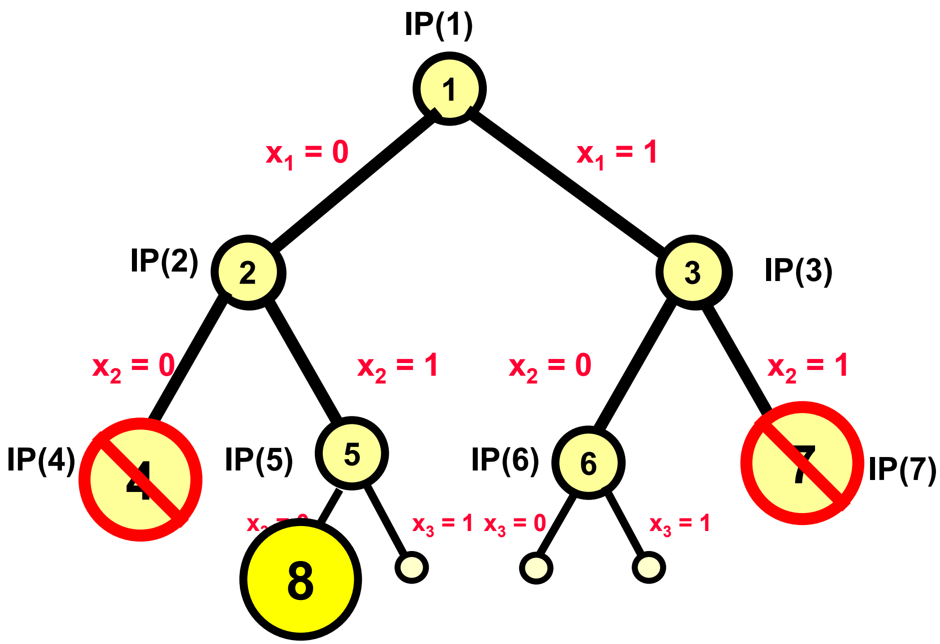
$z_I = 26$

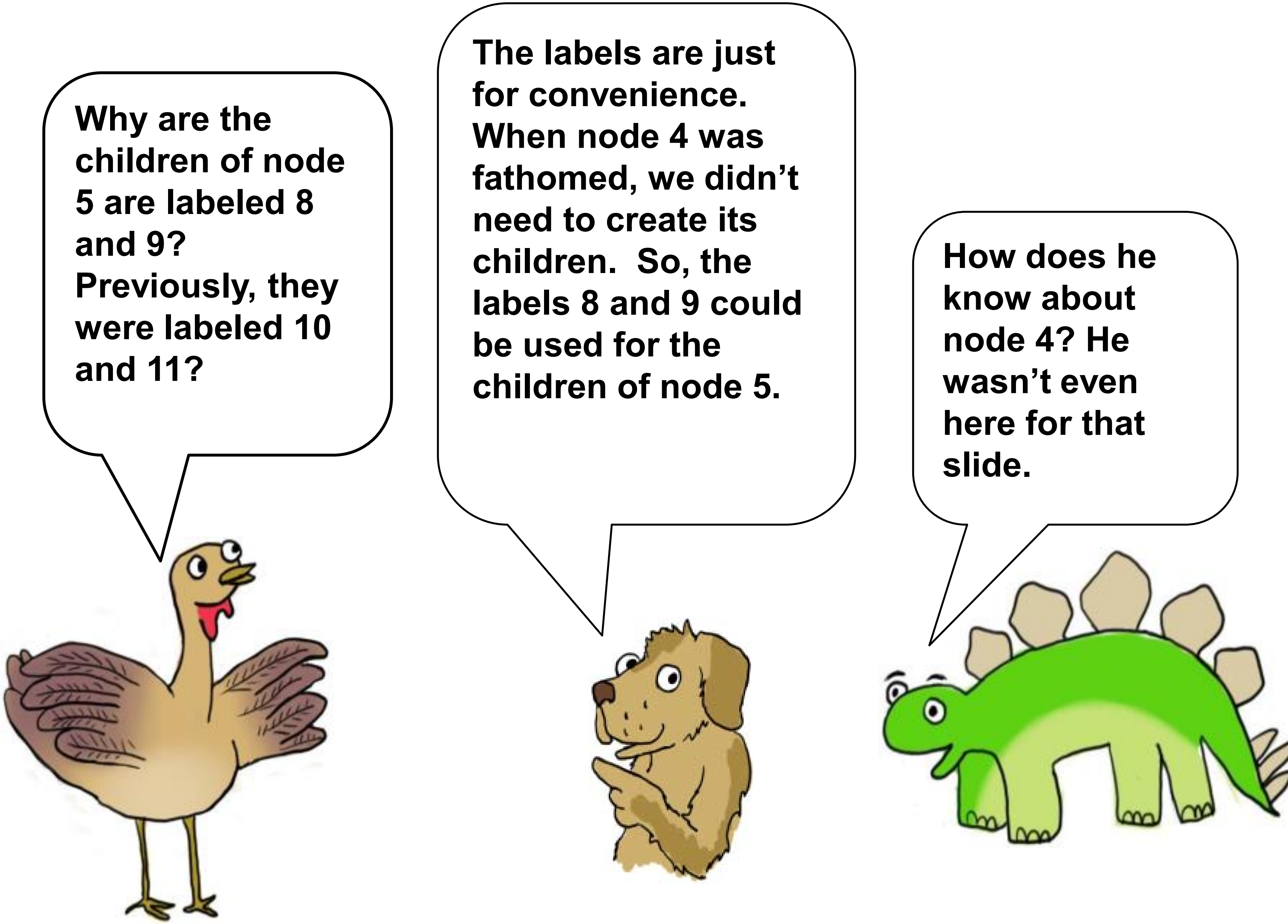
Incumbent

Opt solution for LP(8):

$x_1 = 0; \qquad x_2 = 1,$
$x_3 = 0; \qquad x_4 = 1$

$z_{LP}(8) = 6$




Why are the children of node 5 are labeled 8 and 9? Previously, they were labeled 10 and 11?
The labels are just for convenience. When node 4 was fathomed, we didn't need to create its children. So, the labels 8 and 9 could be used for the children of node 5.
How does he know about node 4? He wasn't even here for that slide.

Branch and Bound: Node 9

Maximize $\quad 24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

subject to $\quad 8\,x_1 + 1\,x_2 + \;\; 5\,x_3 + 4\,x_4 \le 9$

$$x_1 = 0\;,\; x_2 = 1,\; x_3 = 1$$

$$0 \le x_4 \le 1$$

$x_1 = 1, \quad x_2 = 1,$
$x_3 = 0, \quad x_4 = 0$

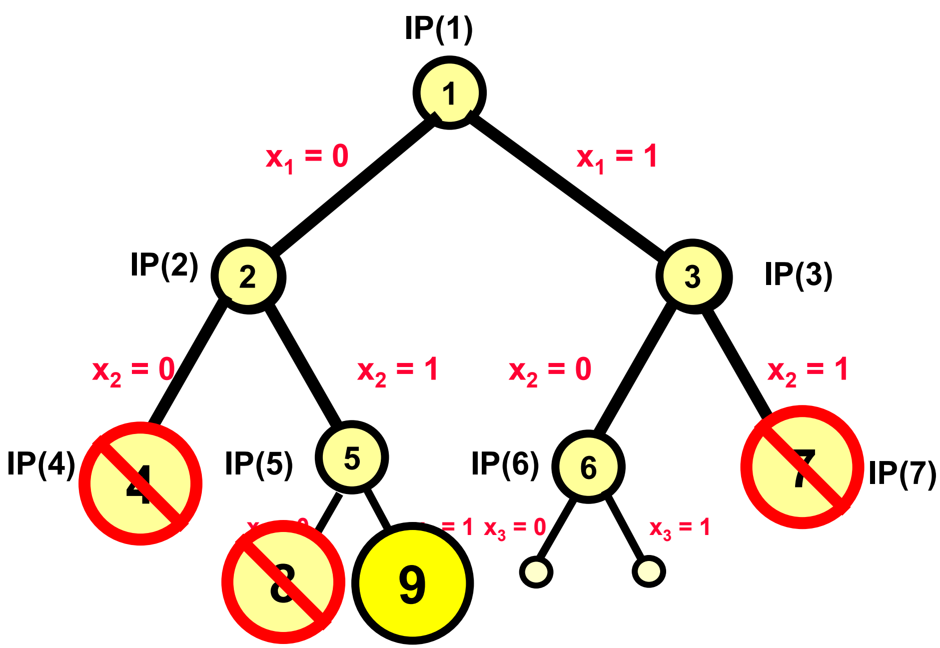
$z_I = 26$

Incumbent

Opt solution for LP(9):

$x_1 = \;0; \qquad x_2 = 1,$
$x_3 = \;1; \qquad x_4 = 3/4$

$z_{LP}(9) = 25$



Branch and Bound: Node 10

Maximize $\quad 24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

subject to $\quad 8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \le 9$

$x_1 = 1\,,\ x_2 = 0,\ x_3 = 0$

$0 \le x_4 \le 1$

$x_1 = 1, \quad x_2 = 1,$
$x_3 = 0, \quad x_4 = 0$

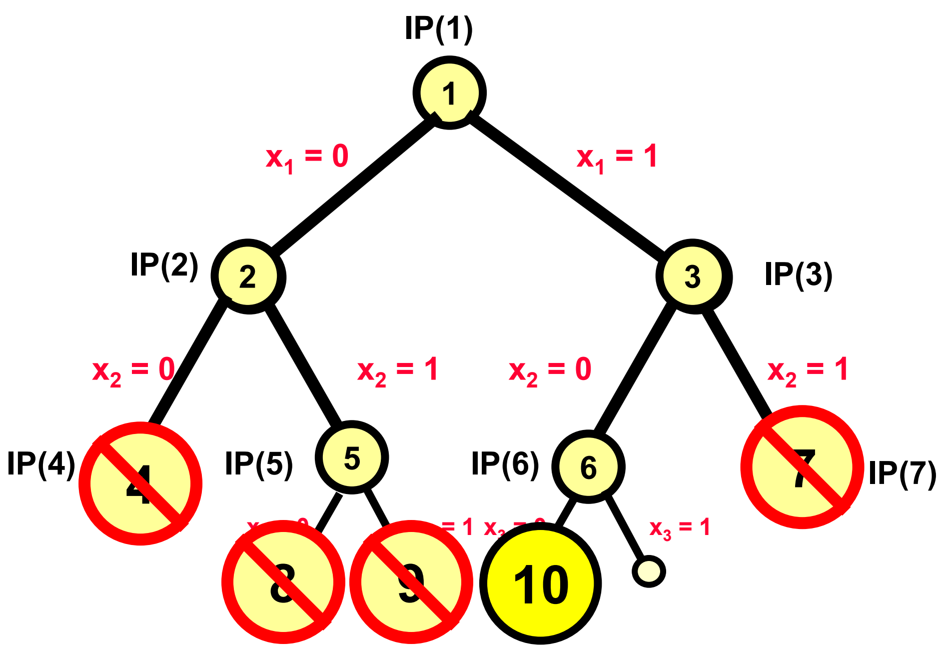
$z_I = 26$

Incumbent

Opt solution for LP(10):

$x_1 = 1; \qquad x_2 = 0,$
$x_3 = 0; \qquad x_4 = 1/4$

$z_{LP}(10) = 25$



Branch and Bound: Node 11

$$\text{Maximize} \quad 24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$$

$$\text{subject to} \quad 8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \le 9$$

$$x_1 = 1,\ x_2 = 0,\ x_3 = 1$$

$$0 \le x_4 \le 1$$

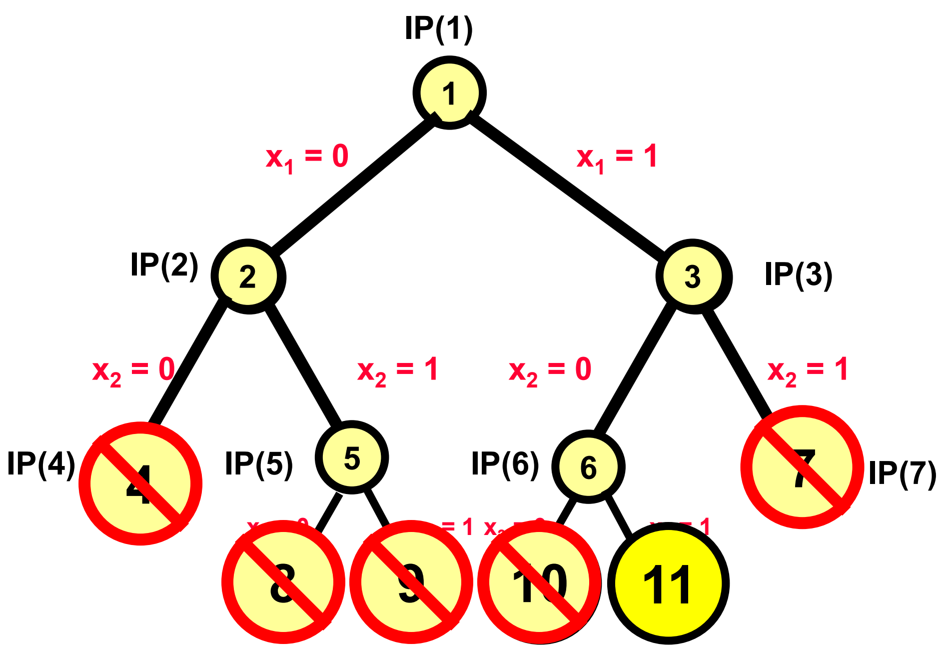
$x_1 = 1, \quad x_2 = 1,$
$x_3 = 0, \quad x_4 = 0$

$z_I = 26$

Incumbent

Opt solution for LP(11):

There is no feasible solution



Branch and Bound: the end

Maximize $24\,x_1 + 2\,x_2 + 20\,x_3 + 4\,x_4$

subject to $8\,x_1 + 1\,x_2 + 5\,x_3 + 4\,x_4 \le 9$

$x_1 = 1\,,\ x_2 = 0,\ x_3 = 1$

$0 \le x_4 \le 1$

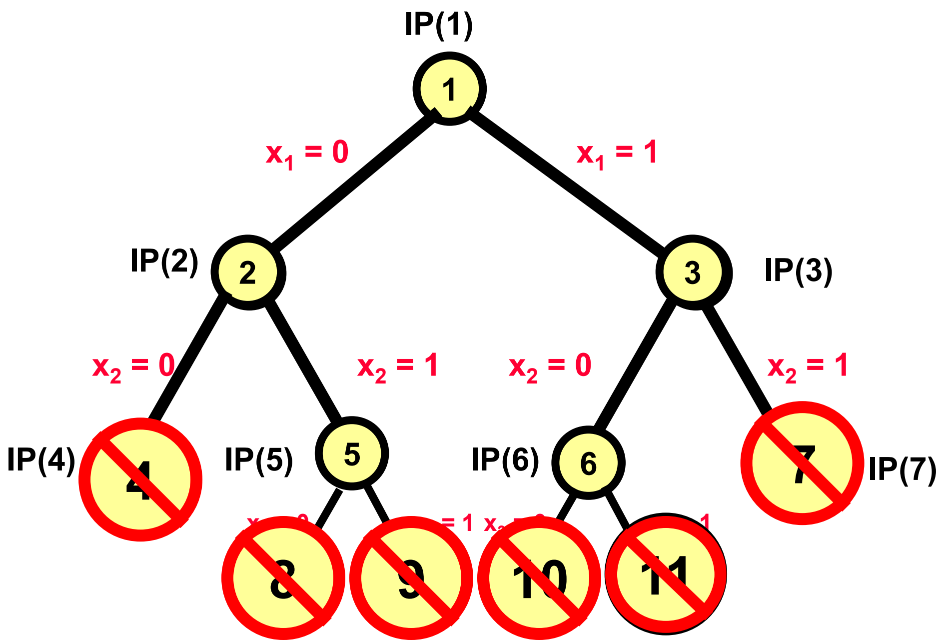
$x_1 = 1, \quad x_2 = 1,$
$x_3 = 0, \quad x_4 = 0$

$z_I = 26$

Incumbent

Opt solution for LP(11):

There is no feasible solution



The end

of B&B

Lessons Learned

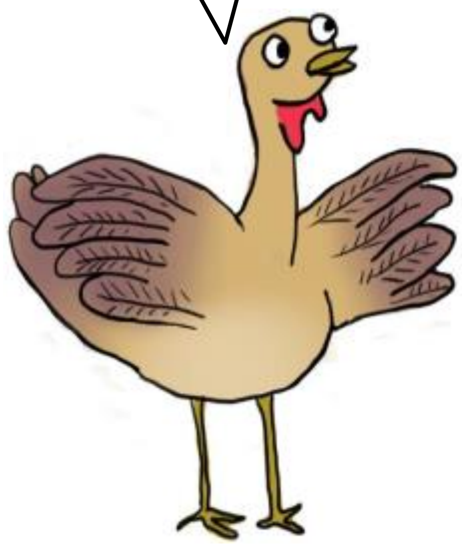
- Branch and Bound can speed up the search
  - Only 11 nodes (LPs) out of 31 were evaluated.

- Branch and Bound relies on eliminating subtrees, either because the IP at the node was solved, or else because the IP solution cannot possibly be optimum.

- Complete enumerations not possible (because of the running time) if there are more than 100 variables. (Even 50 variables would take too long.)

- In practice, there are lots of ways to make Branch and Bound even faster.

OK. I'll bite (so to speak). How can one speed up Branch and Bound?



There are several ways. One way is for the B&B algorithm to have heuristics that "intelligently" choose the best variable to branch on.

Another technique is to use "rounding." Ella explains this on the next slide.

The best technique is to obtain better bounds by adding valid inequalities. Ella explains this two slides from now. This is what the next lecture is all about.

Rounding down to improve bounds

If all cost coefficients of a maximization problem are integer valued, then the optimal objective value (for the IP) is integer. And $z_{IP}(j) \le \lfloor z_{LP}(j) \rfloor$.

$$\begin{array}{ll} \text{Maximize} & 4x_1 + 3x_2 + 3x_3 + 3x_4 \\ \text{subject to} & 2x_1 + 2x_2 + 2x_3 + 2x_4 \le 3 \\ & x_i \in \{0,1\} \quad \text{for } i = 1 \text{ to } 4. \end{array}$$

Opt LP Solution

$x_1 = 1$; $x_2 = 1/2$,
$x_3 = 0$; $x_4 = 0$

$z_{LP} = 5.5$

We conclude that $z_{IP} \le z_{LP} = 5.5$.

But z_{IP} must be integer valued.

So, $z_{IP} \le 5$.

Adding constraints to improve bounds

- A constraint is called a valid inequality if it is satisfied by all integer solutions of an IP (but possibly not the linear solutions of its LP relaxation.)
- Adding a valid inequality might improve the bound.

We illustrate valid inequalities on the next slide. It is the focus of the next lecture.

$$\begin{aligned} \text{Max} \quad & 4x_1 + 3x_2 + 3x_3 + 3x_4 \\ \text{s.t.} \quad & 2x_1 + 2x_2 + 2x_3 + 2x_4 \le 3 \\ & x_i \in \{0,1\} \quad \text{for } i = 1 \text{ to } 4. \end{aligned} \qquad \textbf{A}$$

Opt LP Solution

$x_1 = 1$; $x_2 = 1/2$,
$x_3 = 0$; $x_4 = 0$

$z_{LP} = 5.5$

$$\begin{aligned} \text{Max} \quad & 4x_1 + 3x_2 + 3x_3 + 3x_4 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 \le 1.5 \\ & x_i \in \{0,1\} \quad \text{for } i = 1 \text{ to } 4. \end{aligned} \qquad \textbf{B}$$

$$\begin{aligned} \text{Max} \quad & 4x_1 + 3x_2 + 3x_3 + 3x_4 \\ \text{s.t.} \quad & x_1 + x_2 + x_3 + x_4 \le 1 \\ & x_i \in \{0,1\} \quad \text{for } i = 1 \text{ to } 4. \end{aligned} \qquad \textbf{C}$$

Opt LP Solution

$x_1 = 1$; $x_2 = 0$,
$x_3 = 0$; $x_4 = 0$

$z_{LP} = 4$

The solution for LP(C) is optimal for IP(C)!

Summary

- **Making Branch and Bound work well in practice requires lots of good ideas.**

- **There was not time in class to cover all of these ideas in any detail.**

- **The best idea for speeding up Branch and Bound is to add valid inequalities, or improve the inequalities. We cover this in the next lecture.**