Programming III

Ricardo Wehbe

UADE

4 de noviembre de 2021

Programme

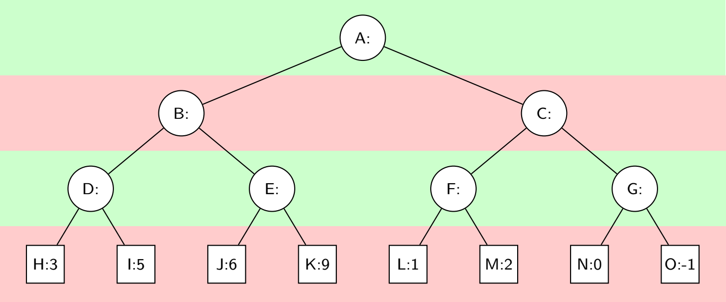
1. Review of the Previous Class

2. Branch and Bound
    - FIFO-BB
    - LIFO-BB
    - LC-BB

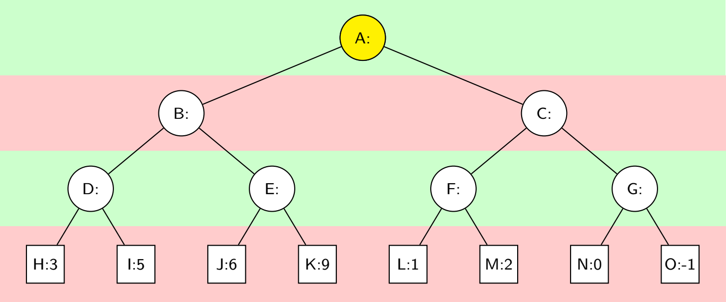
1. Review of the Previous Class

2. Branch and Bound
   - FIFO-BB
   - LIFO-BB
   - LC-BB

The Min-Max Algorithm. An Example

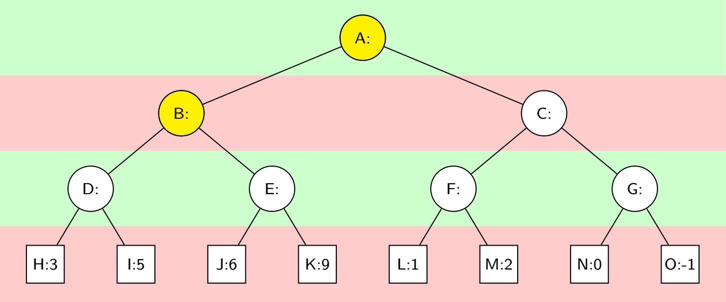


The Min-Max Algorithm. An Example



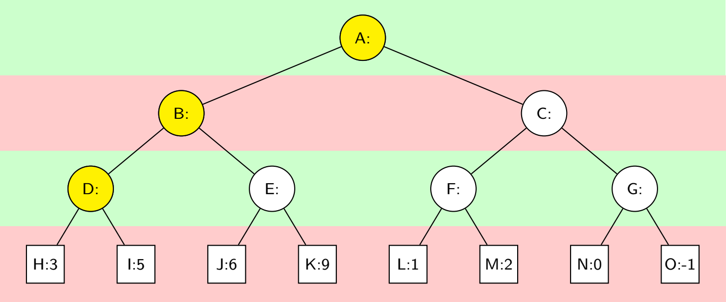
Max chooses the maximum between *B* and *C*; it calls first *B*.

The Min-Max Algorithm. An Example



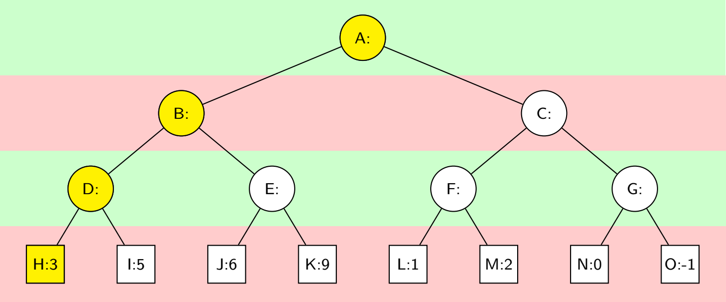
Min chooses the minimum between *D* and *E*; it calls first *D*.

The Min-Max Algorithm. An Example



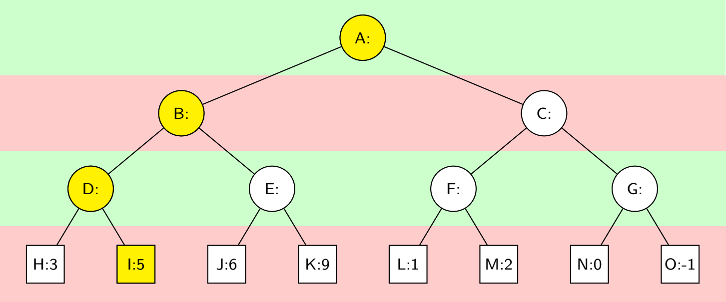
Max chooses the maximum between *H* and *I*; it calls first *H*.

The Min-Max Algorithm. An Example



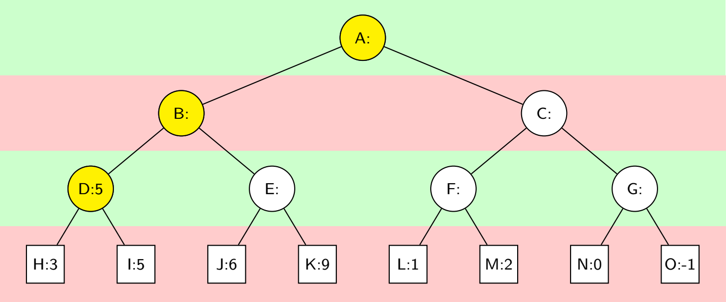
Node *H* yields 3.

The Min-Max Algorithm. An Example



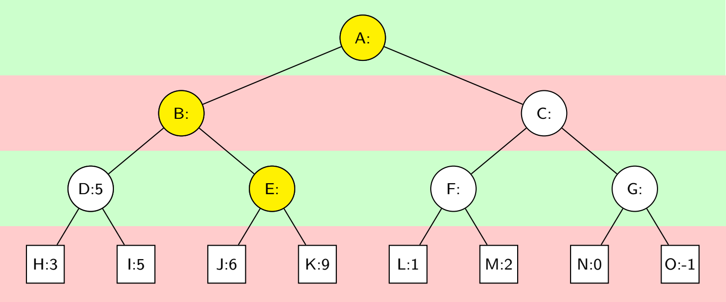
Node *I* yields 5.

The Min-Max Algorithm. An Example



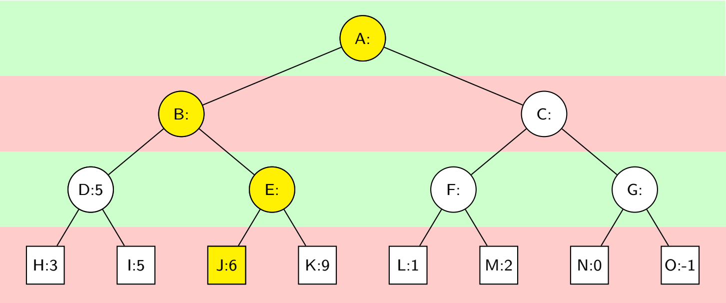
Node D yields $\texttt{max}(3,5) = 5$.

The Min-Max Algorithm. An Example



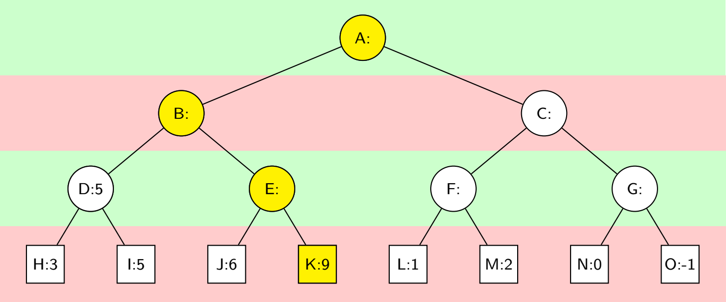
Max chooses the maximum between *J* and *K*; it calls first *J*.

The Min-Max Algorithm. An Example



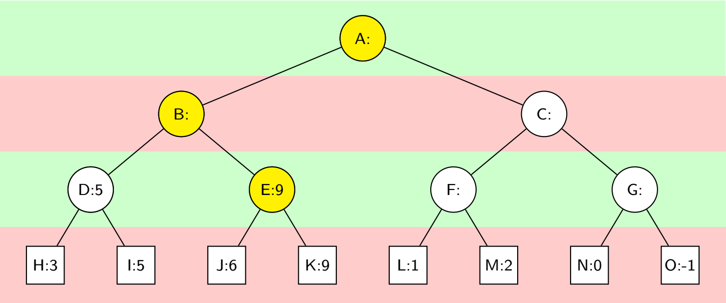
Node J yields 6.

The Min-Max Algorithm. An Example



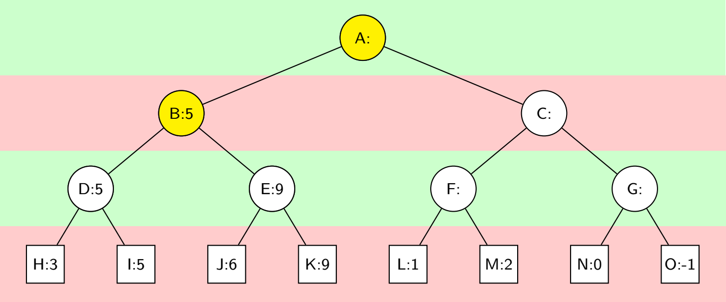
Node K yields 9.

The Min-Max Algorithm. An Example



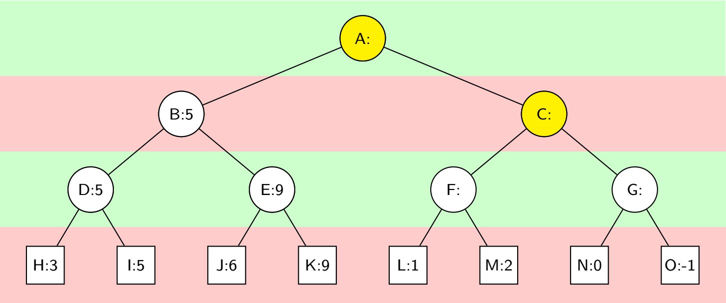
Node E yields $\mathtt{max}(6, 9) = 9$.

The Min-Max Algorithm. An Example



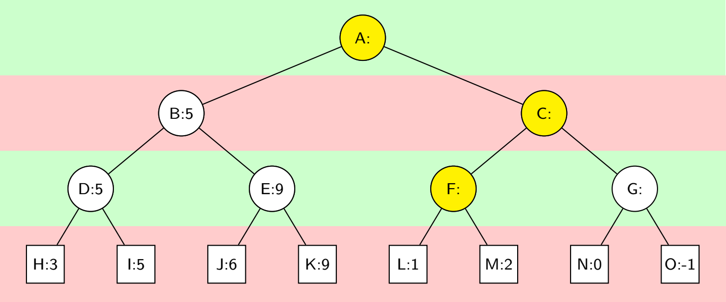
Node B yields $\mathtt{min}(5, 9) = 5$.

The Min-Max Algorithm. An Example



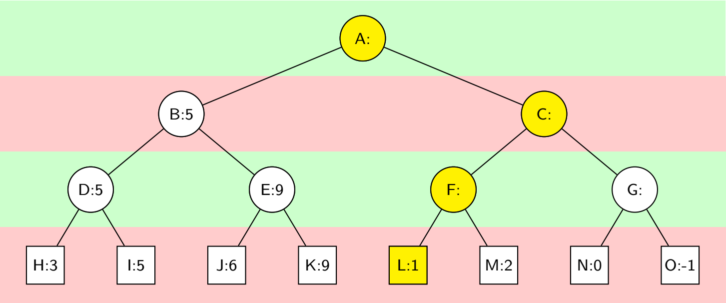
Min chooses the minimum between *F* and *G*; it calls first *F*.

The Min-Max Algorithm. An Example



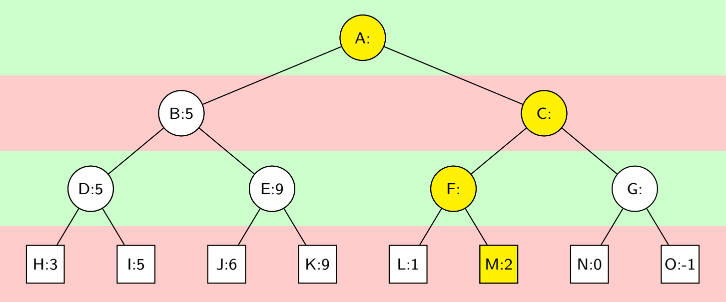
Max chooses the maximum between *L* and *M*; it calls first *L*.

The Min-Max Algorithm. An Example



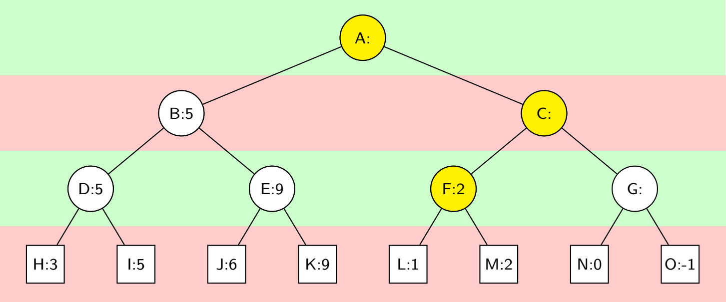
Node L yields 1.

The Min-Max Algorithm. An Example



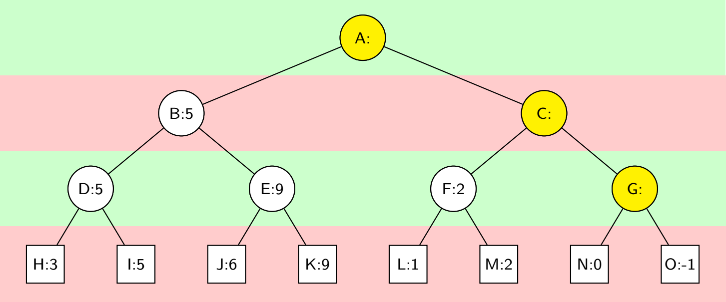
Node M yields 2.

The Min-Max Algorithm. An Example



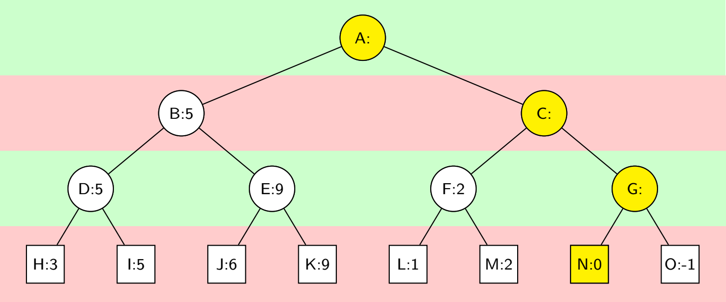
Node F yields $\mathtt{max}(1, 2) = 2$.

The Min-Max Algorithm. An Example



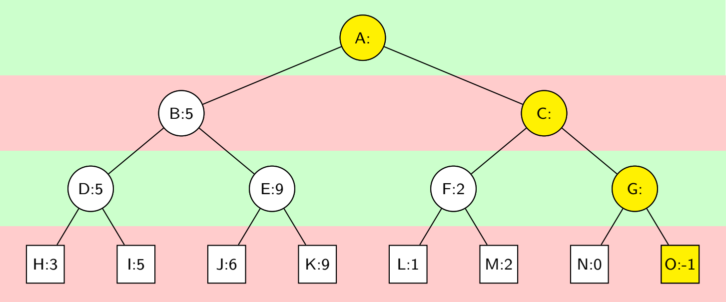
Max chooses the maximum between *N* and *O*; it calls first *N*.

The Min-Max Algorithm. An Example



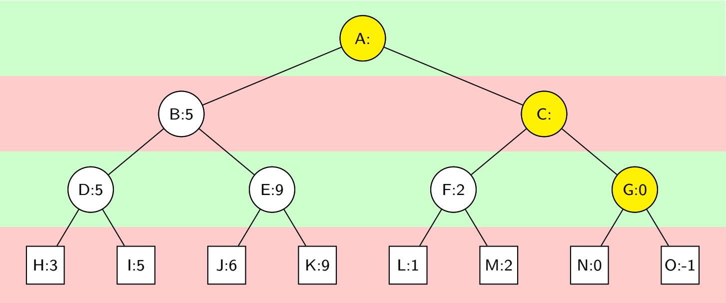
Node N yields 0.

The Min-Max Algorithm. An Example



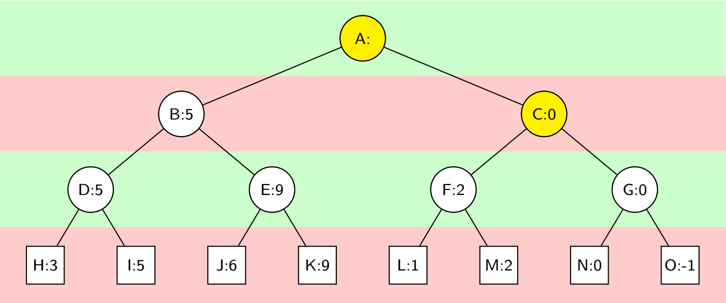
Node *O* yields -1.

The Min-Max Algorithm. An Example



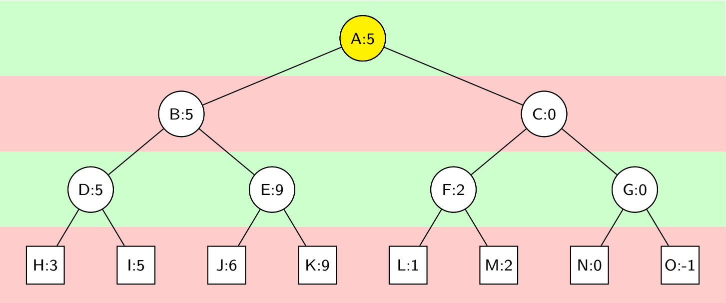
Node F yields $\max(0, -1) = 0$.

The Min-Max Algorithm. An Example



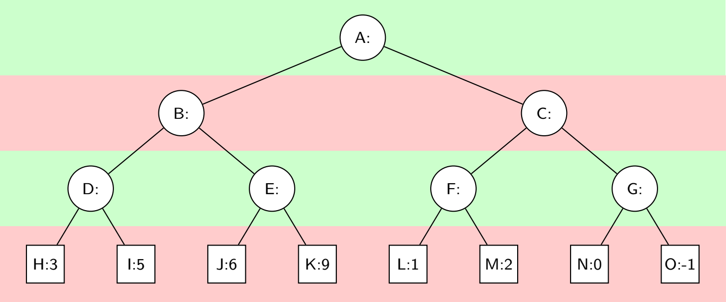
Node C yields $\mathtt{min}(2, 0) = 0$.

The Min-Max Algorithm. An Example

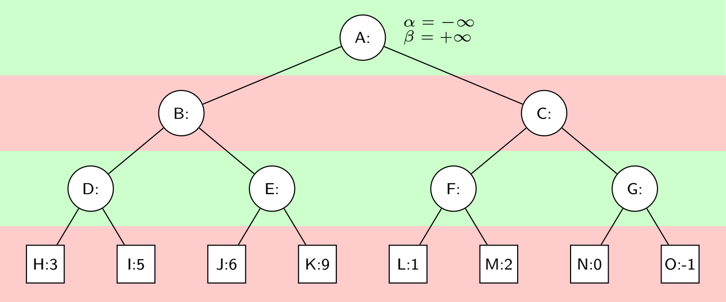


Node A yields $\texttt{min}(5, 0) = 5$.

Alpha–Beta Pruning. An Example

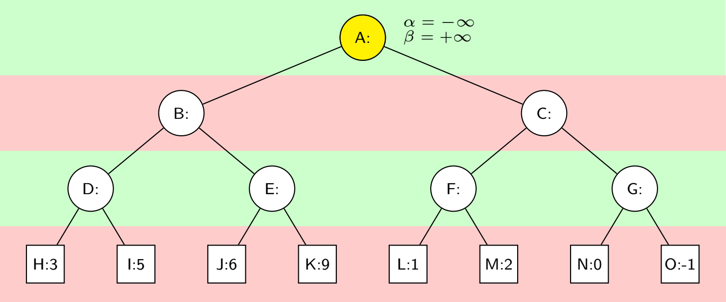


Alpha–Beta Pruning. An Example



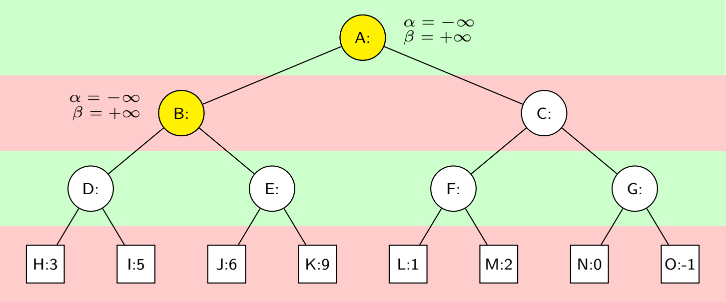
The process starts at A
The initial values are $\alpha = -\infty$ and $\beta = \infty$.

Alpha–Beta Pruning. An Example



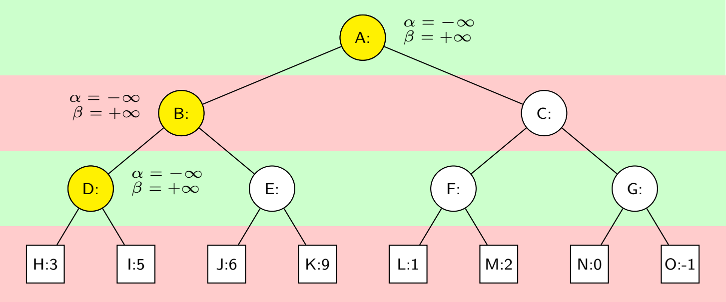
Max chooses the maximum between *B* and *C*; it calls first *B*.

Alpha–Beta Pruning. An Example



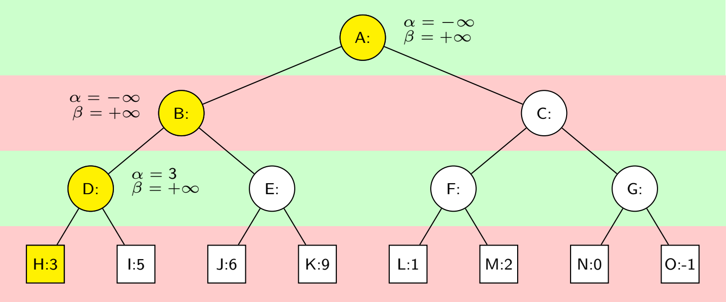
The values of α and β are passed downwards to *B*.
Min chooses the minimum between *D* and *E*; it calls first *D*.

Alpha–Beta Pruning. An Example



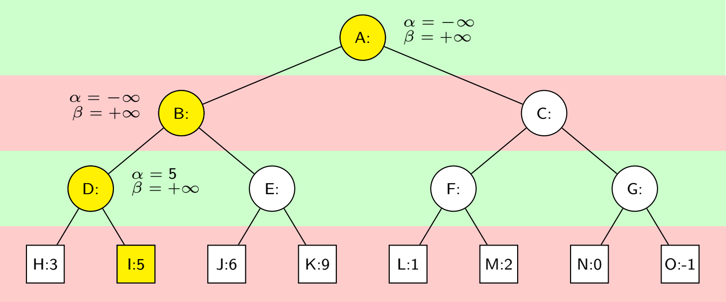
The values of α and β are passed downwards to *D*.
Max chooses the minimum between *H* and *I*; it calls first *H*.

Alpha–Beta Pruning. An Example



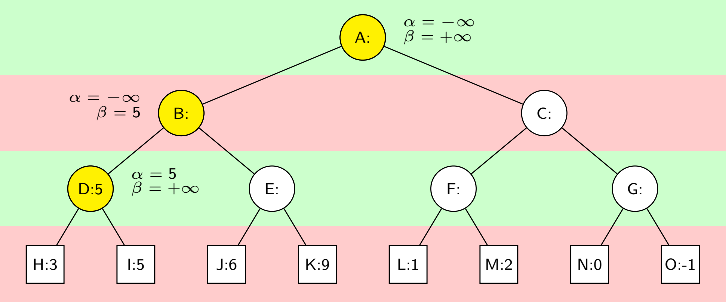
Node H yields 3.
The value of α at D is updated to 3.

Alpha–Beta Pruning. An Example



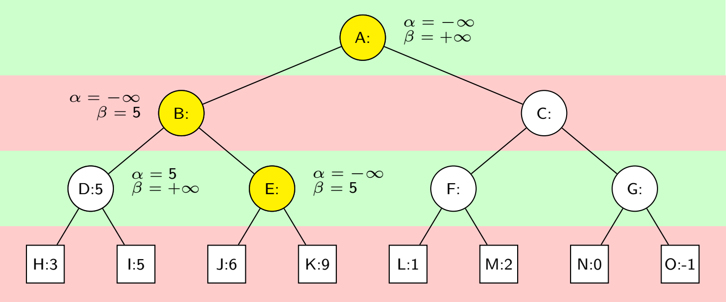
Node *I* yields 5.
The value of α at *D* is updated to 5.

Alpha–Beta Pruning. An Example



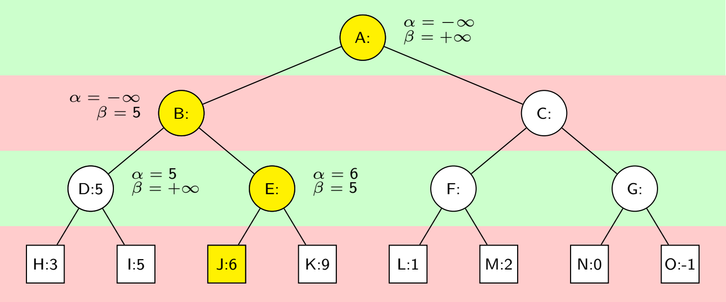
Node D yields $\texttt{max}(3, 5) = 5$.
The value of β at B is updated to 5.

Alpha–Beta Pruning. An Example



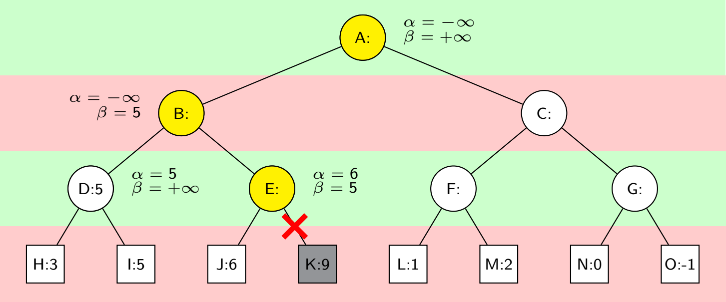
The values of α and β are passed downwards to *E*.
Max chooses the maximum between *J* and *K*; it calls first *J*.

Alpha–Beta Pruning. An Example



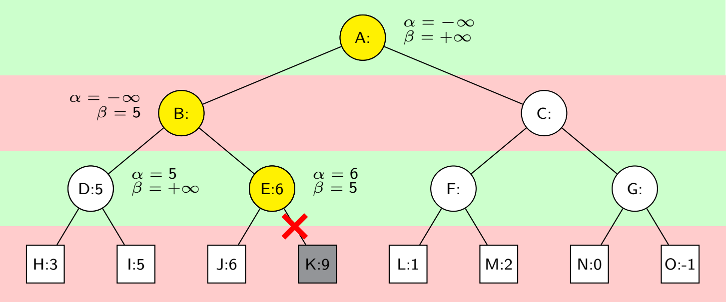
Node J yields 6.
The value of α at E is updated to 6.

Alpha–Beta Pruning. An Example



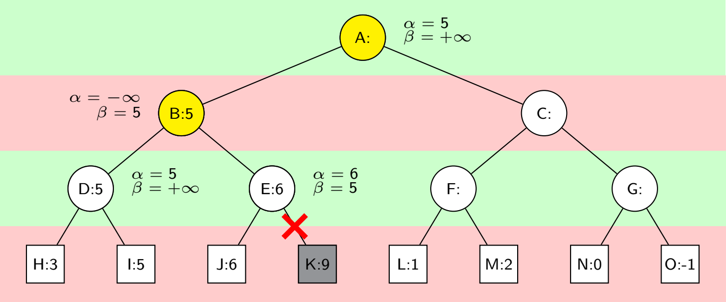
We have now $\alpha \geq \beta$
The exploration finishes. The branch is pruned.

Alpha–Beta Pruning. An Example



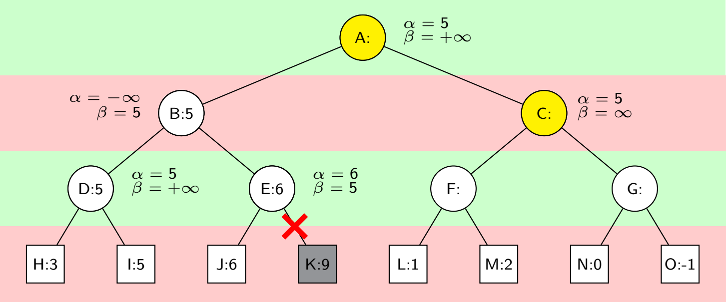
Node *E* yields 6.

Alpha–Beta Pruning. An Example



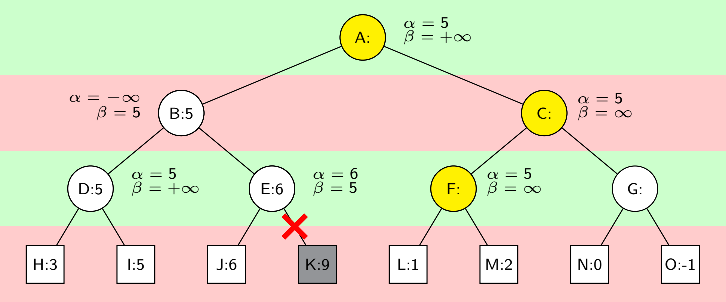
Node B yields $\mathtt{min}(5, 6) = 5$.
The value of α at A is updated to 5.

Alpha–Beta Pruning. An Example



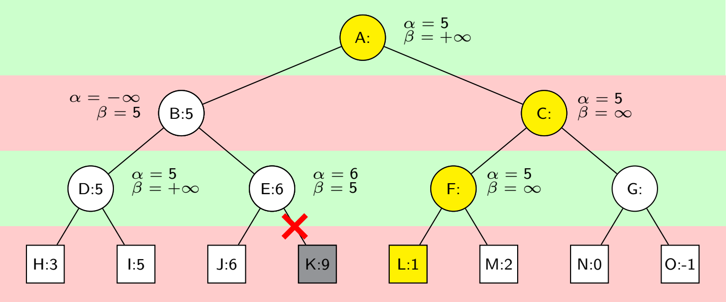
The values of α and β are passed downwards to *C*.
Min chooses the minimum between *F* and *G*; it calls first *F*.

Alpha–Beta Pruning. An Example



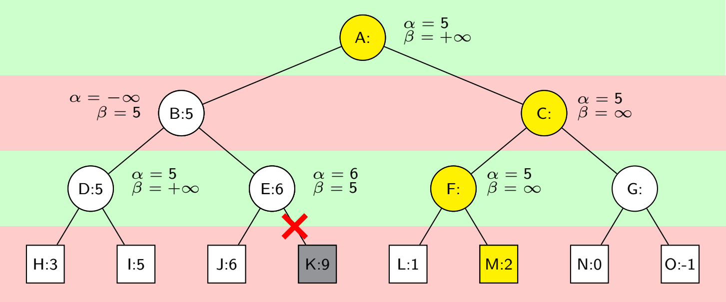
The values of α and β are passed downwards to F.
Max chooses the maximum between L and M; it calls first L.

Alpha–Beta Pruning. An Example



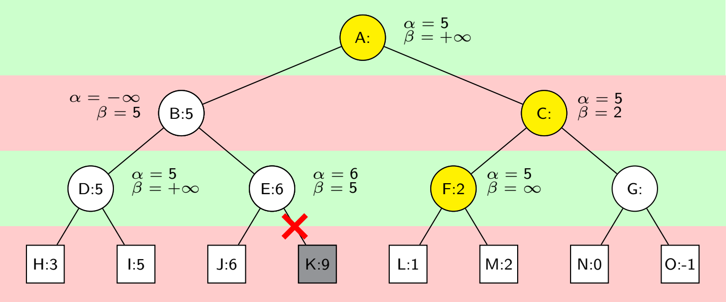
Node L yields 1.
The value of α does not change at F.

Alpha–Beta Pruning. An Example



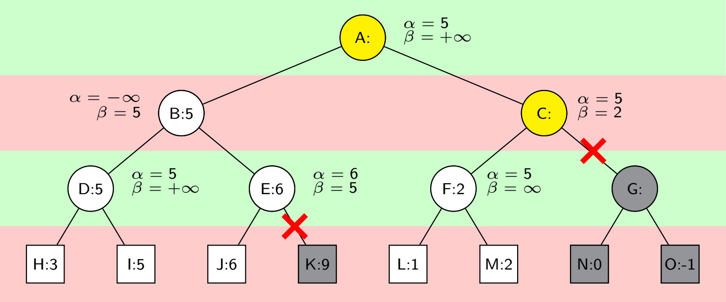
Node M yields 2.
The value of α does not change at F.

Alpha–Beta Pruning. An Example



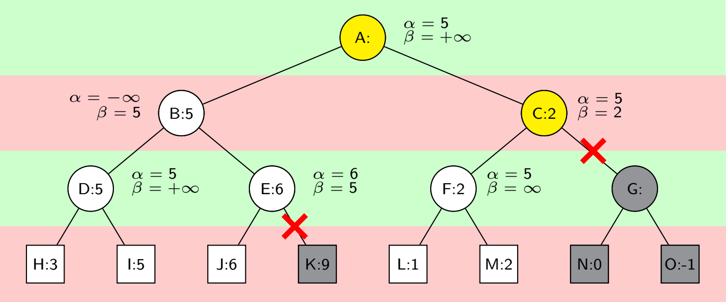
Node F yields $\texttt{min}(1, 2) = 2$.
The value of β at C is updated to 2.

Alpha–Beta Pruning. An Example



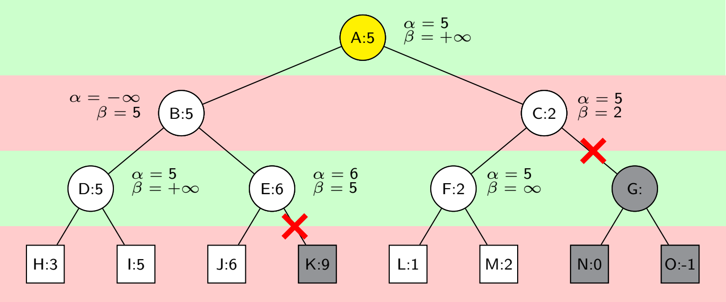
We have now $\alpha \geq \beta$
The exploration finishes. The branch is pruned.

Alpha–Beta Pruning. An Example



Node C yields 2.

Alpha–Beta Pruning. An Example



Node A yields $\mathtt{max}(5, 2) = 5$.

Branch and Bound

Branch and Bound

- This technique is used mainly in optimisation problems, more specifically minimisation problems. Maximisation problems may be easily adapted.

Branch and Bound

- This technique is used mainly in optimisation problems, more specifically minimisation problems. Maximisation problems may be easily adapted.
- There is a global variable, which we call *upper*, that contains the minimal value that it is so far guaranteed.

Branch and Bound

- This technique is used mainly in optimisation problems, more specifically minimisation problems. Maximisation problems may be easily adapted.
- There is a global variable, which we call *upper*, that contains the minimal value that it is so far guaranteed.
- For each node, two magnitudes are calculated: u (the *top*)and c (the *cost*.)

Branch and Bound

- This technique is used mainly in optimisation problems, more specifically minimisation problems. Maximisation problems may be easily adapted.
- There is a global variable, which we call *upper*, that contains the minimal value that it is so far guaranteed.
- For each node, two magnitudes are calculated: u (the *top*)and c (the *cost*.)
- The number c represents the cost to reach the vertex; the top u is the minimum score that we have guaranteed at that point. It could improve but not get worse.

Branch and Bound

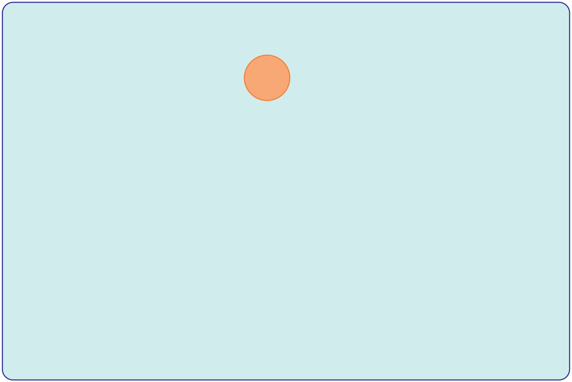
- This technique is used mainly in optimisation problems, more specifically minimisation problems. Maximisation problems may be easily adapted.
- There is a global variable, which we call *upper*, that contains the minimal value that it is so far guaranteed.
- For each node, two magnitudes are calculated: u (the *top*)and c (the *cost*.)
- The number c represents the cost to reach the vertex; the top u is the minimum score that we have guaranteed at that point. It could improve but not get worse.
- if $u < upper$, the variable *upper* is updated to the new minimum guaranteed result.

Branch and Bound

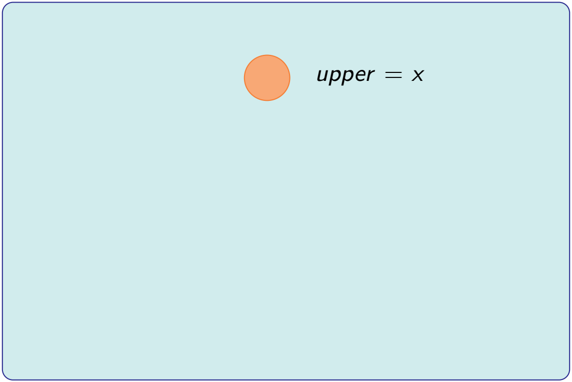
- This technique is used mainly in optimisation problems, more specifically minimisation problems. Maximisation problems may be easily adapted.
- There is a global variable, which we call *upper*, that contains the minimal value that it is so far guaranteed.
- For each node, two magnitudes are calculated: u (the *top*)and c (the *cost*.)
- The number c represents the cost to reach the vertex; the top u is the minimum score that we have guaranteed at that point. It could improve but not get worse.
- if $u < upper$, the variable *upper* is updated to the new minimum guaranteed result.
- If $c > upper$, the branch is killed, since it not worth further exploring.

Branch and Bound: the Central Idea

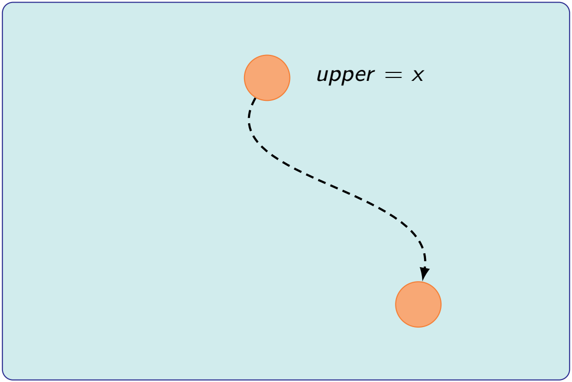
Branch and Bound: the Central Idea



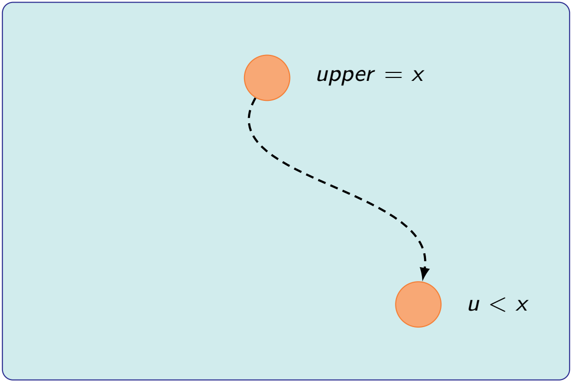
Branch and Bound: the Central Idea



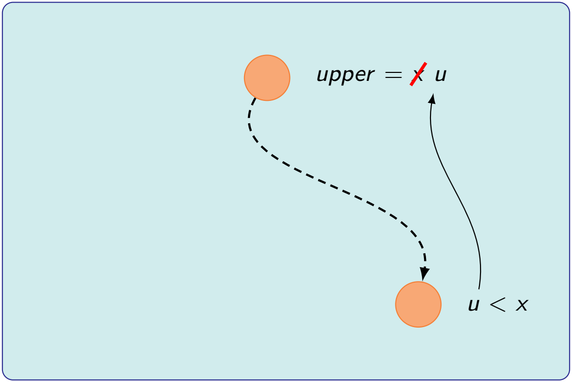
Branch and Bound: the Central Idea



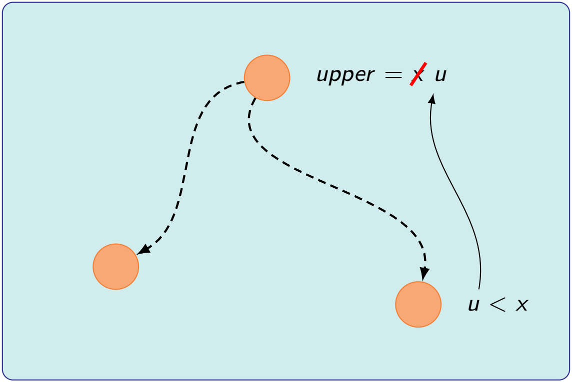
Branch and Bound: the Central Idea



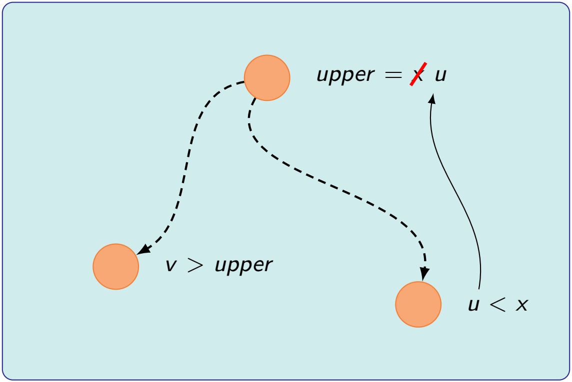
Branch and Bound: the Central Idea



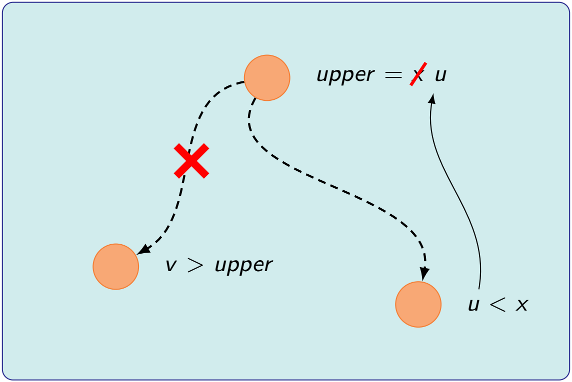
Branch and Bound: the Central Idea



Branch and Bound: the Central Idea

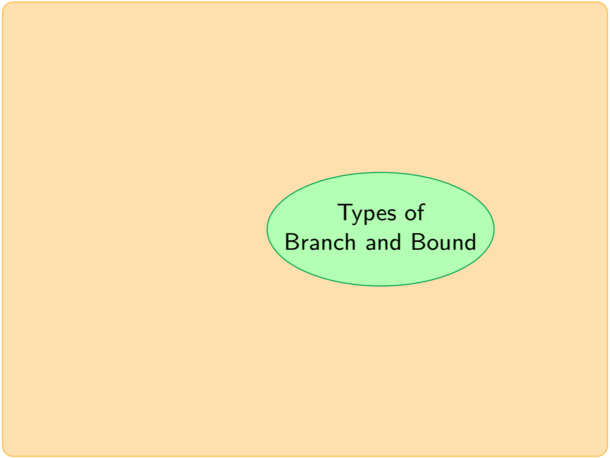


Branch and Bound: the Central Idea

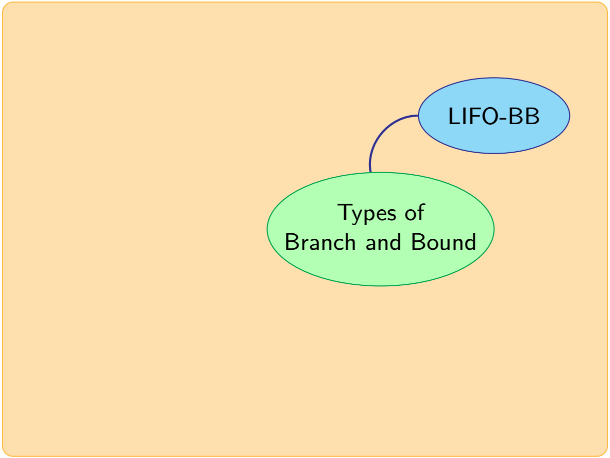


Types of Branch and Bound

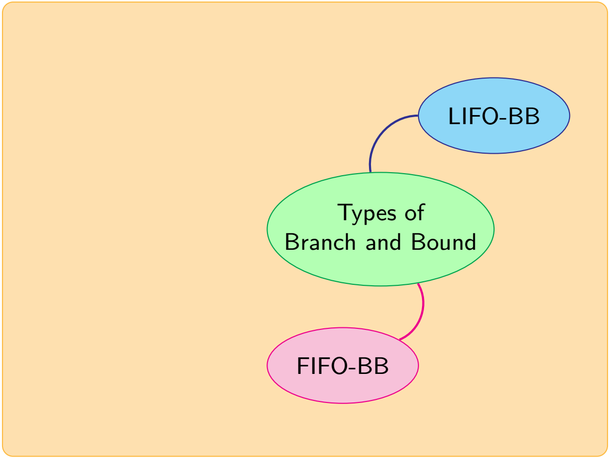
Types of Branch and Bound



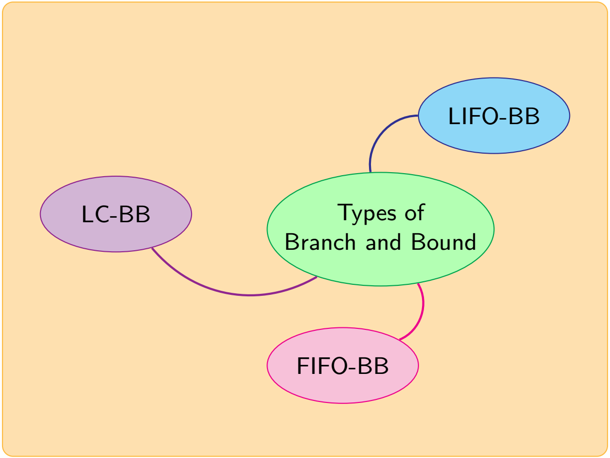
Types of Branch and Bound



Types of Branch and Bound



Types of Branch and Bound

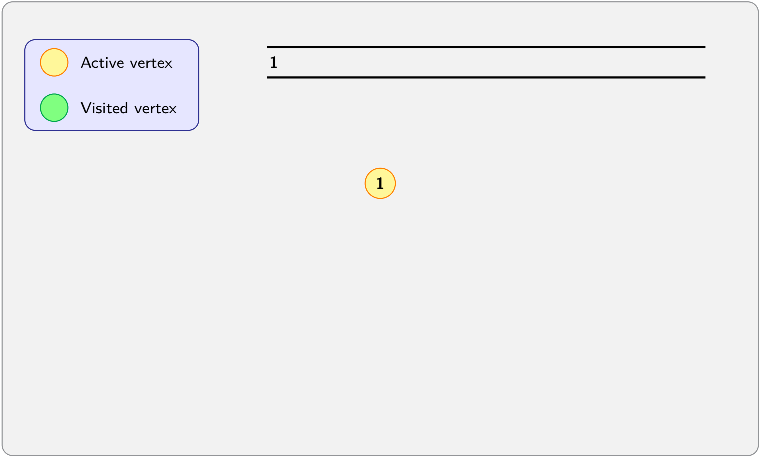


1. Review of the Previous Class

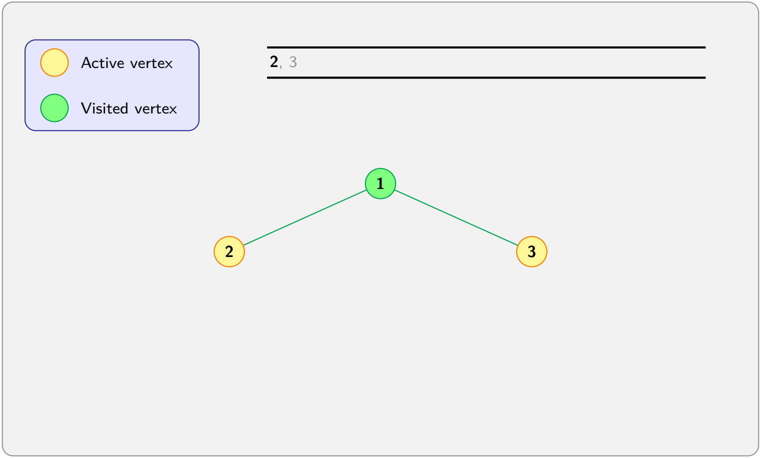
2. Branch and Bound
   - FIFO-BB
   - LIFO-BB
   - LC-BB

First-In-First-Out-Branch and Bound (FIFO-BB)

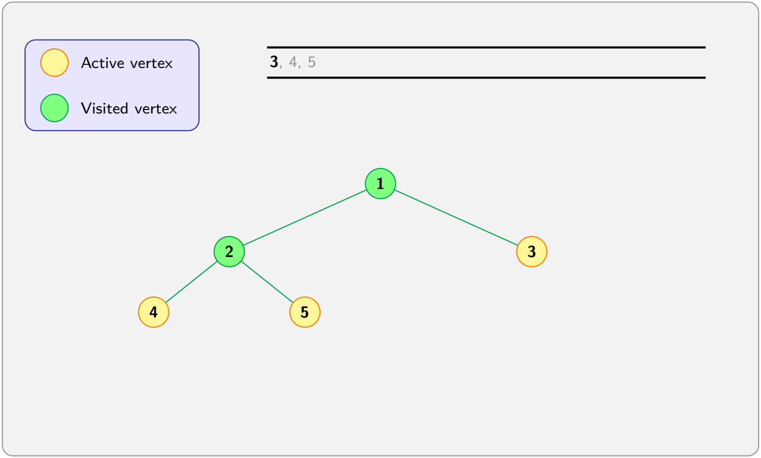
First-In-First-Out-Branch and Bound (FIFO-BB)



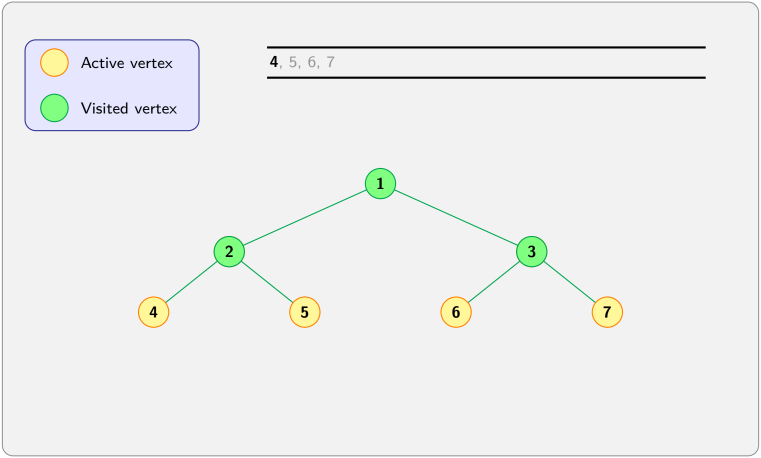
First-In-First-Out-Branch and Bound (FIFO-BB)



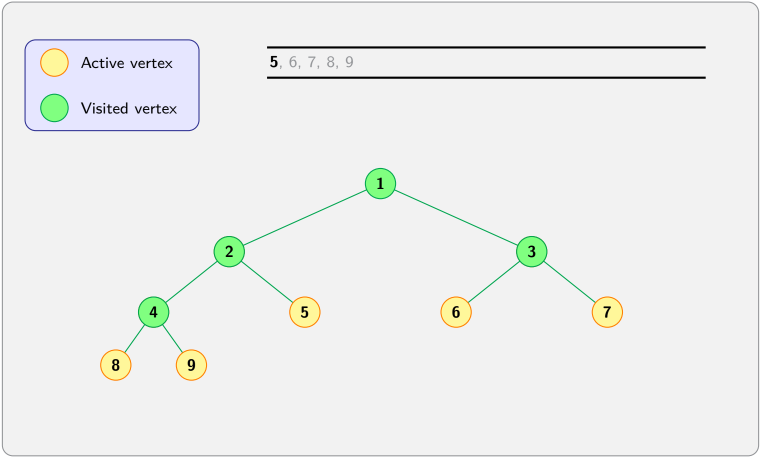
First-In-First-Out-Branch and Bound (FIFO-BB)



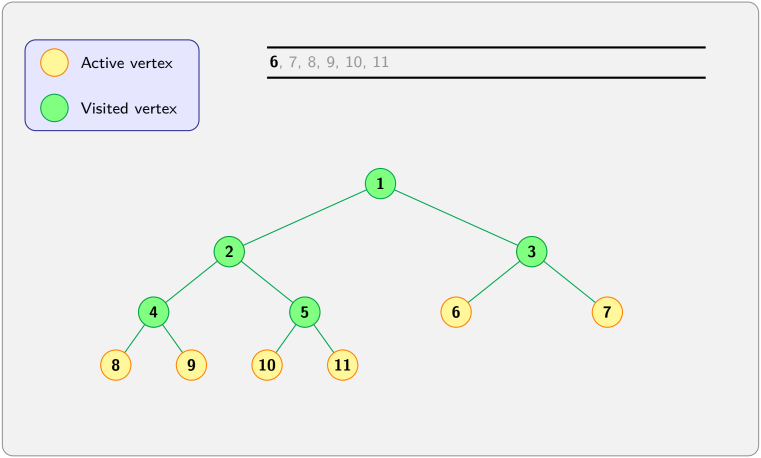
First-In-First-Out-Branch and Bound (FIFO-BB)



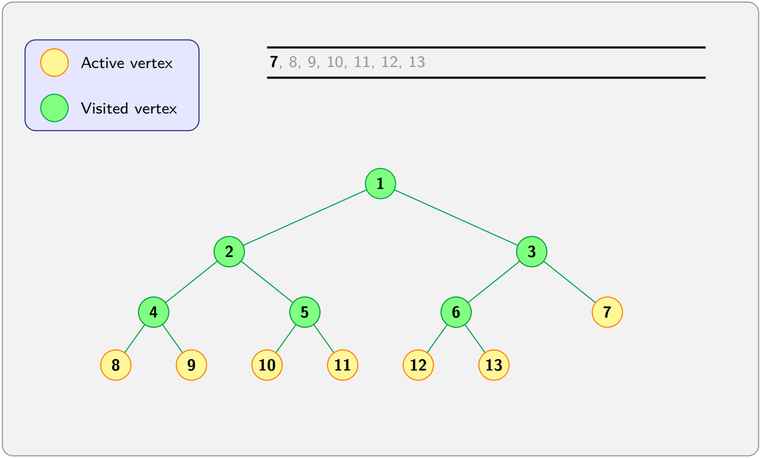
First-In-First-Out-Branch and Bound (FIFO-BB)



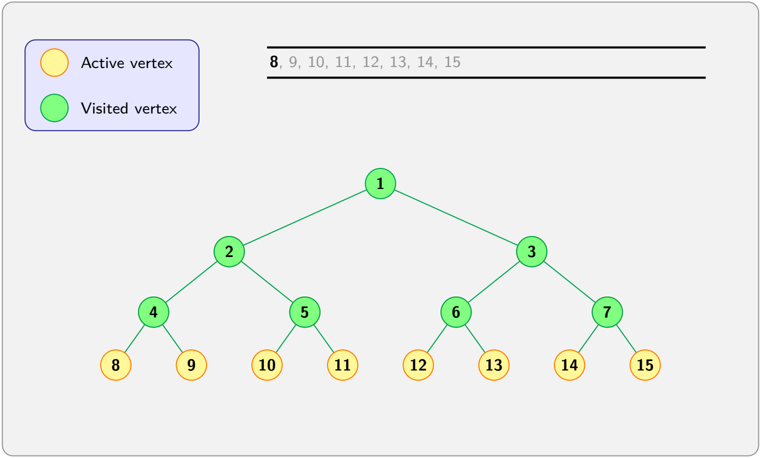
First-In-First-Out-Branch and Bound (FIFO-BB)



First-In-First-Out-Branch and Bound (FIFO-BB)



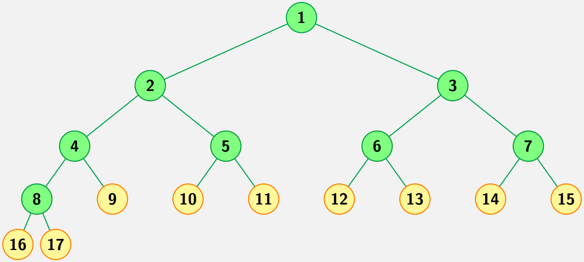
First-In-First-Out-Branch and Bound (FIFO-BB)



First-In-First-Out-Branch and Bound (FIFO-BB)

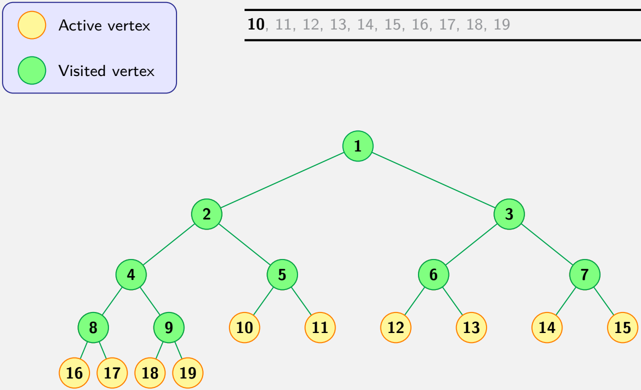
Active vertex
Visited vertex

9, 10, 11, 12, 13, 14, 15, 16, 17

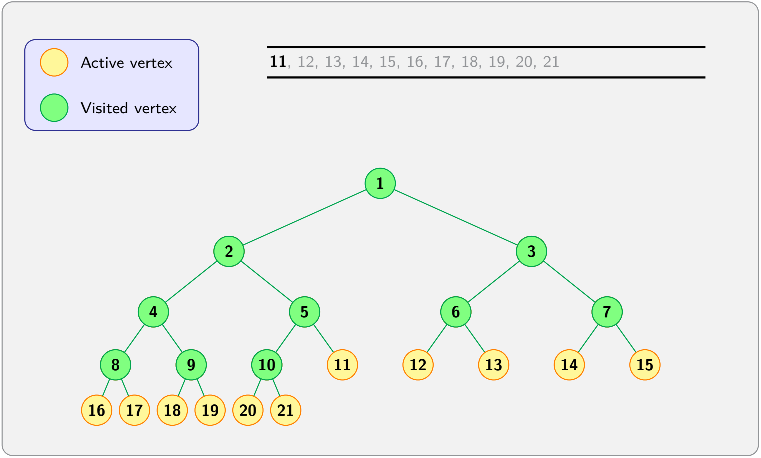


First-In-First-Out-Branch and Bound (FIFO-BB)

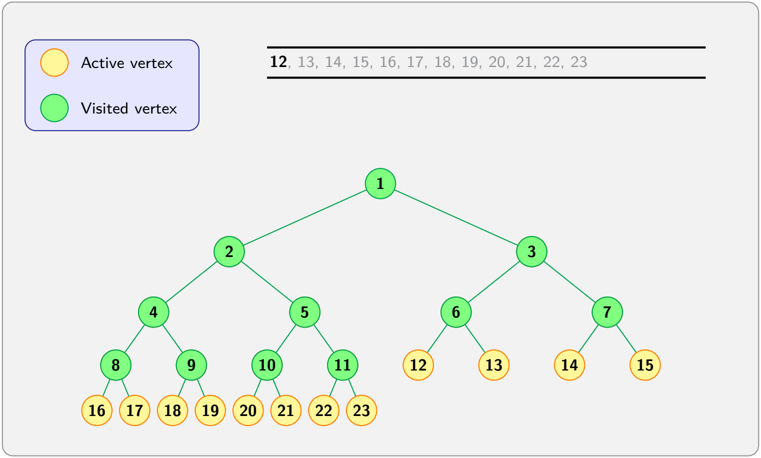
10, 11, 12, 13, 14, 15, 16, 17, 18, 19



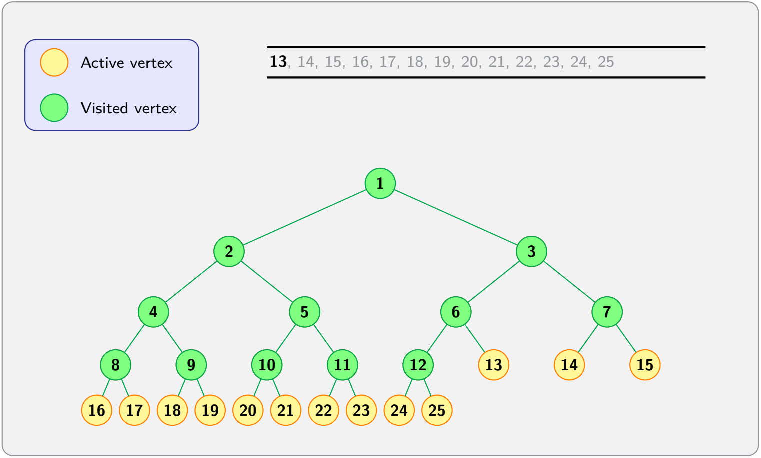
First-In-First-Out-Branch and Bound (FIFO-BB)



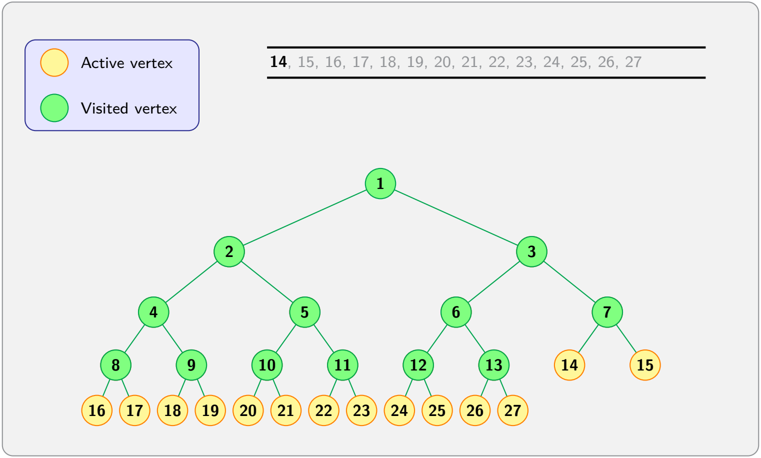
First-In-First-Out-Branch and Bound (FIFO-BB)



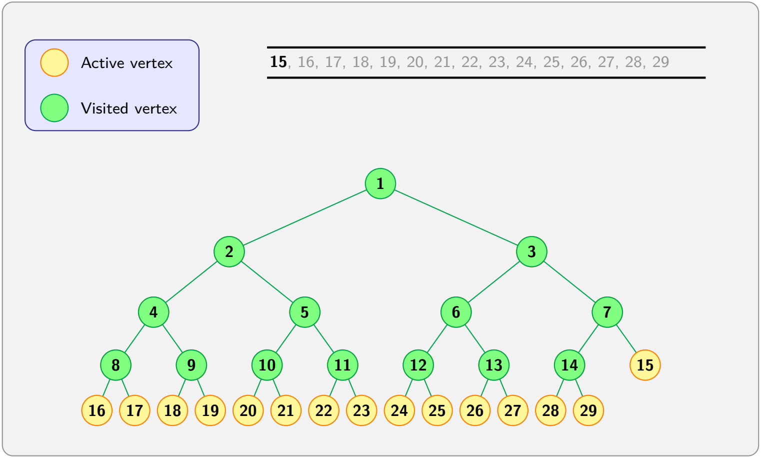
First-In-First-Out-Branch and Bound (FIFO-BB)



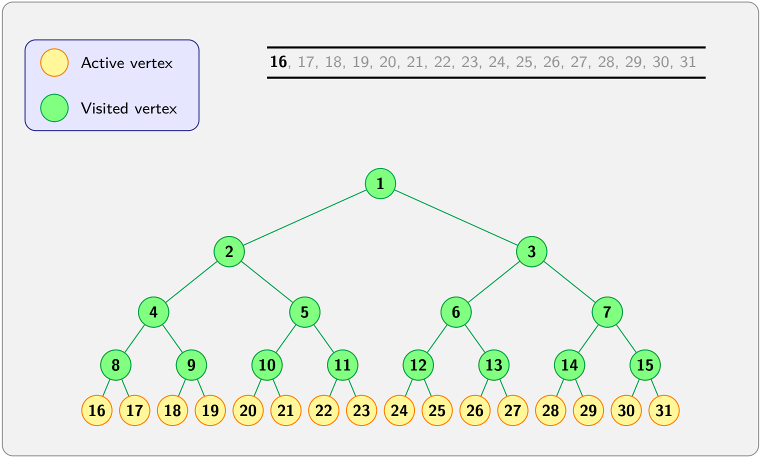
First-In-First-Out-Branch and Bound (FIFO-BB)



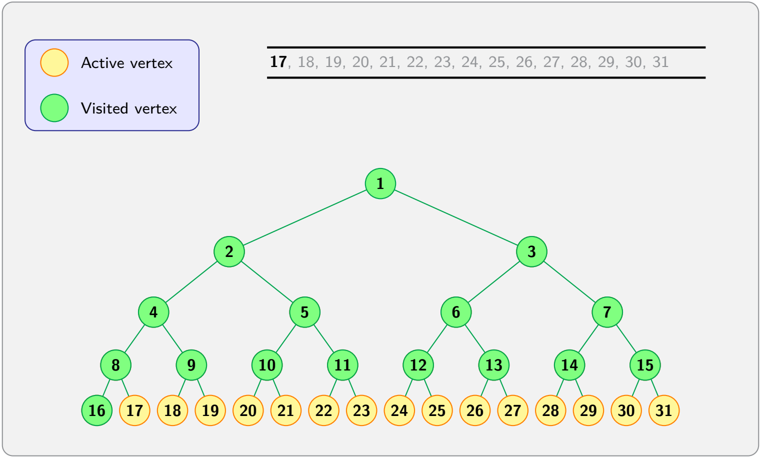
First-In-First-Out-Branch and Bound (FIFO-BB)



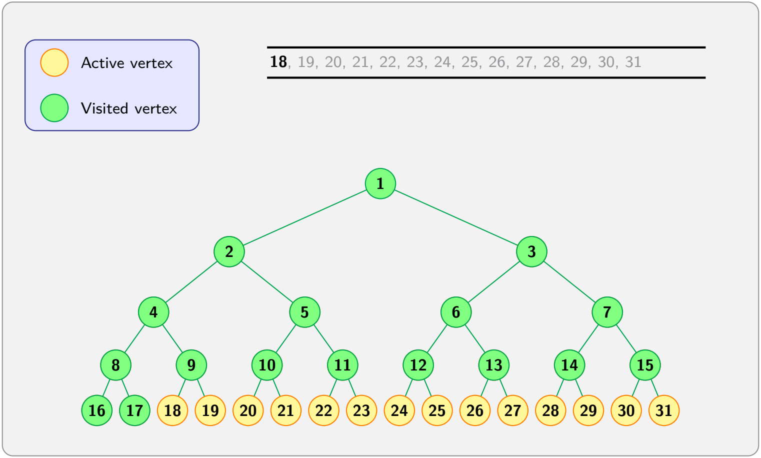
First-In-First-Out-Branch and Bound (FIFO-BB)



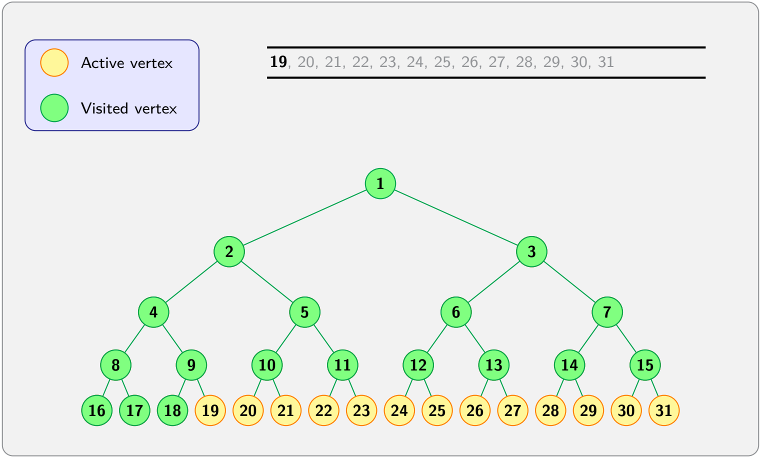
First-In-First-Out-Branch and Bound (FIFO-BB)



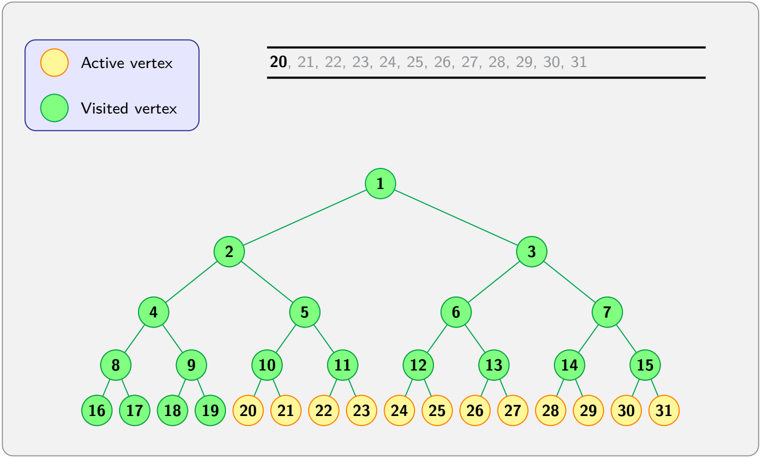
First-In-First-Out-Branch and Bound (FIFO-BB)



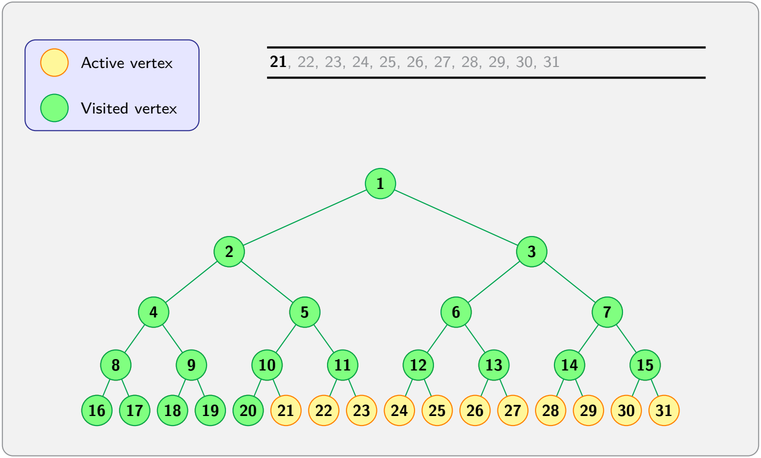
First-In-First-Out-Branch and Bound (FIFO-BB)



First-In-First-Out-Branch and Bound (FIFO-BB)



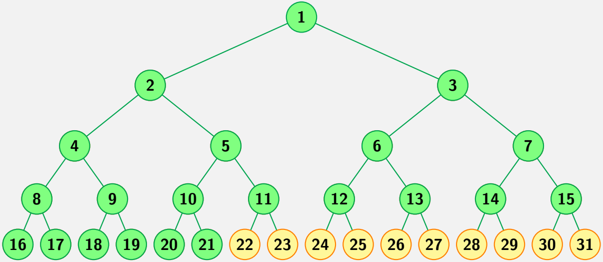
First-In-First-Out-Branch and Bound (FIFO-BB)



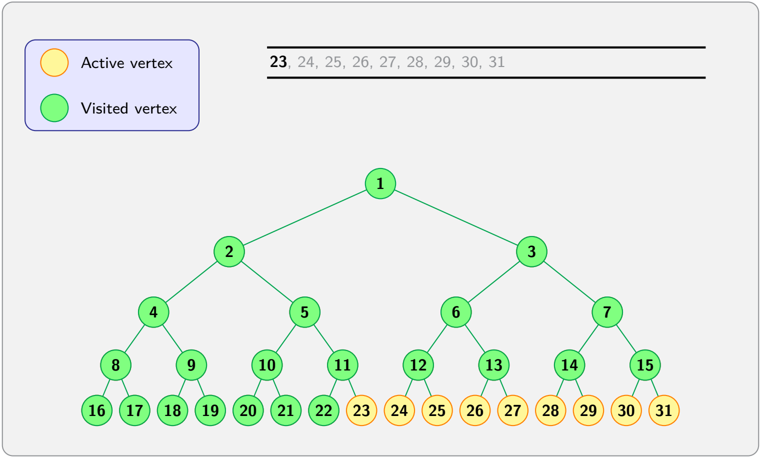
First-In-First-Out-Branch and Bound (FIFO-BB)

Active vertex
Visited vertex

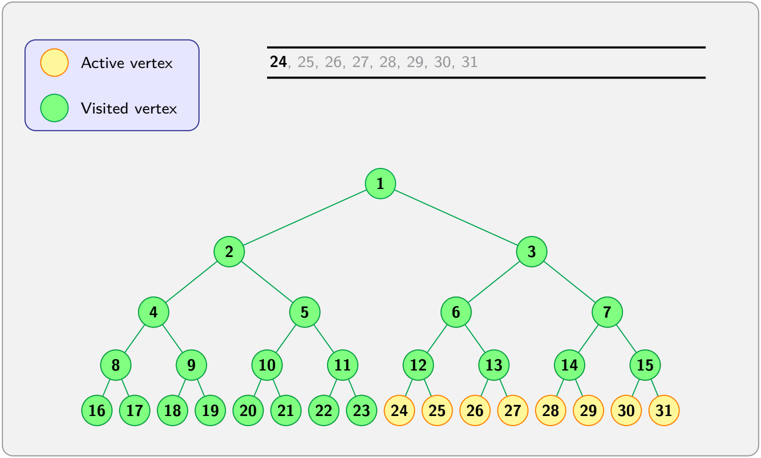
22, 23, 24, 25, 26, 27, 28, 29, 30, 31



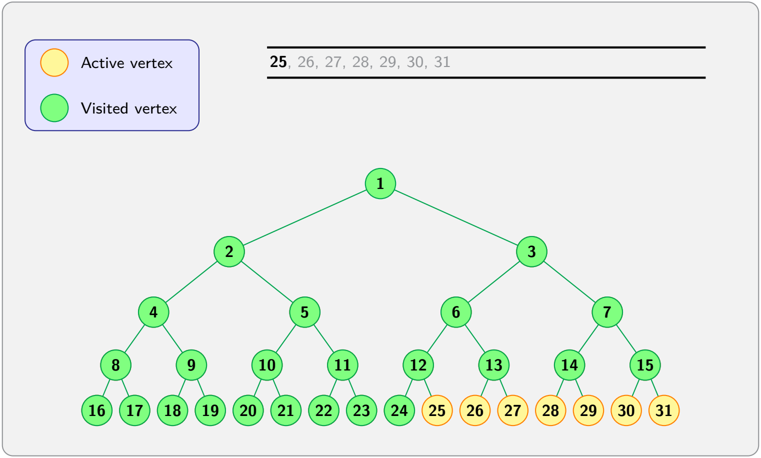
First-In-First-Out-Branch and Bound (FIFO-BB)



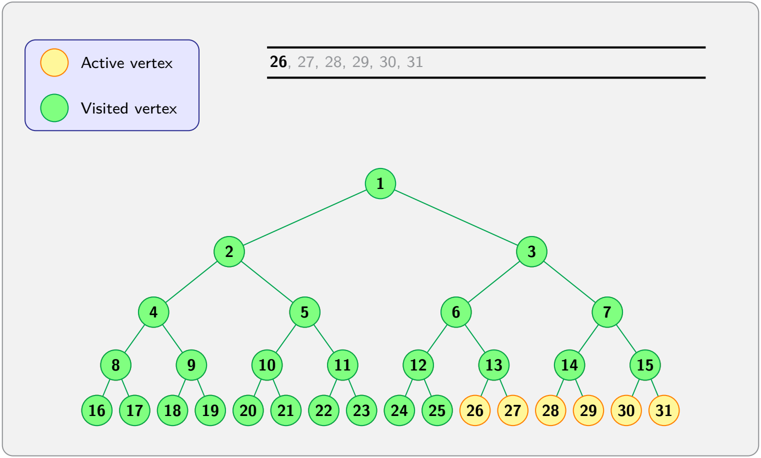
First-In-First-Out-Branch and Bound (FIFO-BB)



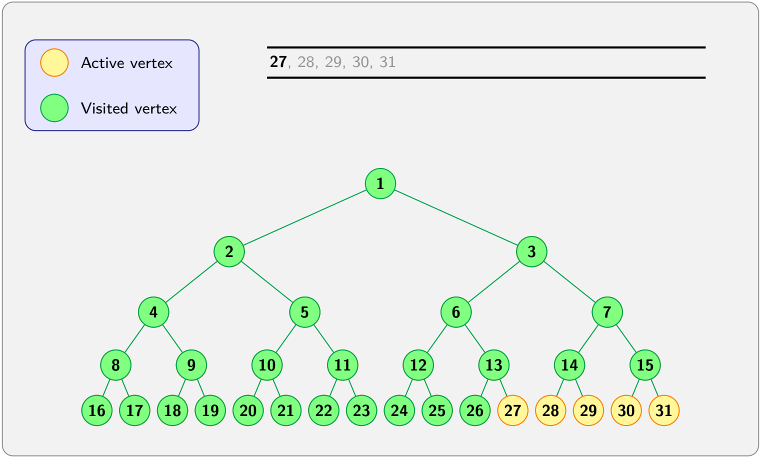
First-In-First-Out-Branch and Bound (FIFO-BB)



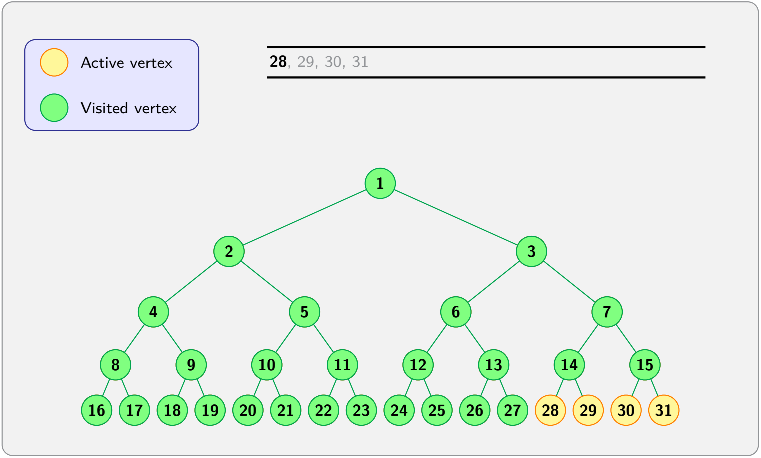
First-In-First-Out-Branch and Bound (FIFO-BB)



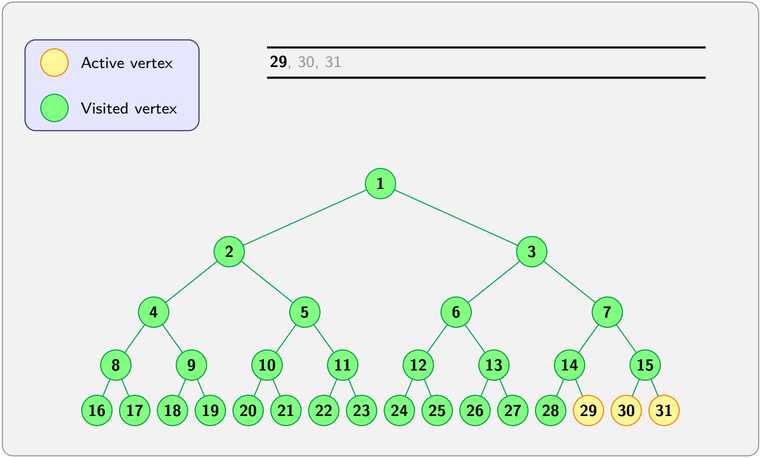
First-In-First-Out-Branch and Bound (FIFO-BB)



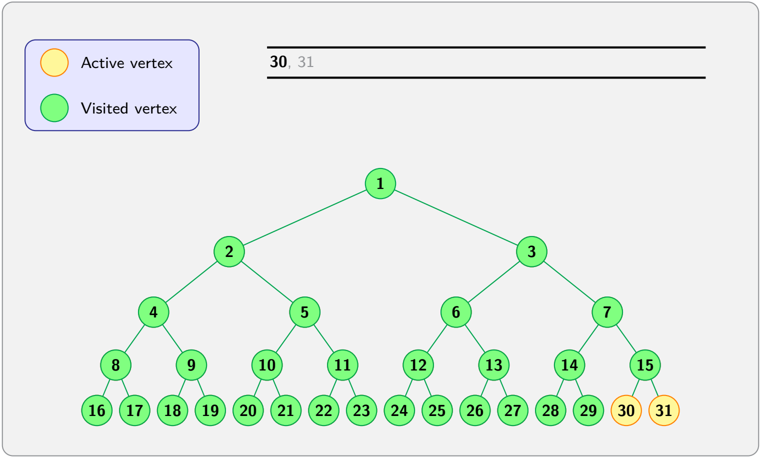
First-In-First-Out-Branch and Bound (FIFO-BB)



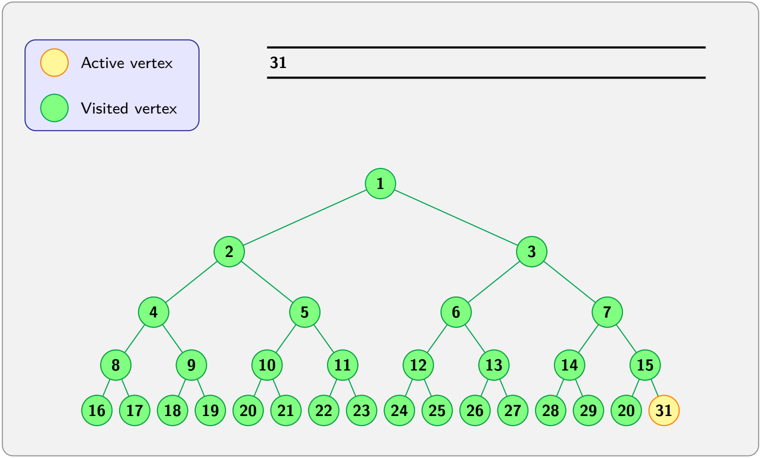
First-In-First-Out-Branch and Bound (FIFO-BB)



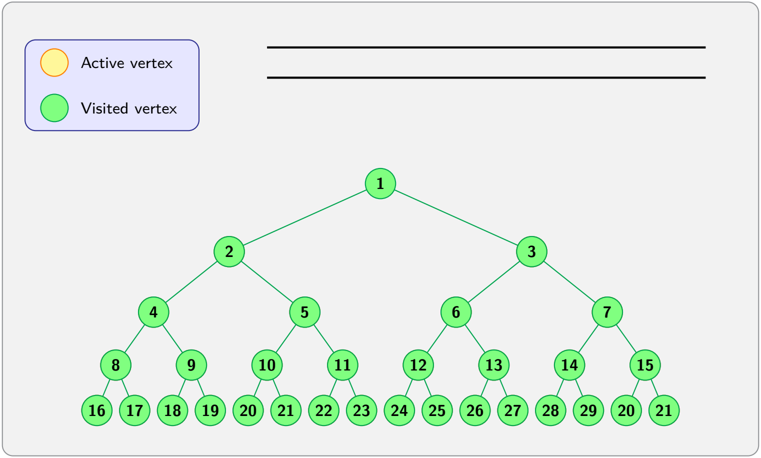
First-In-First-Out-Branch and Bound (FIFO-BB)



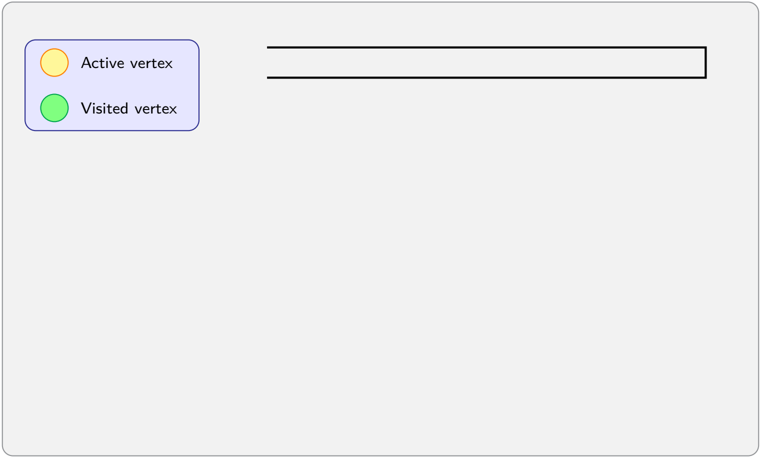
First-In-First-Out-Branch and Bound (FIFO-BB)



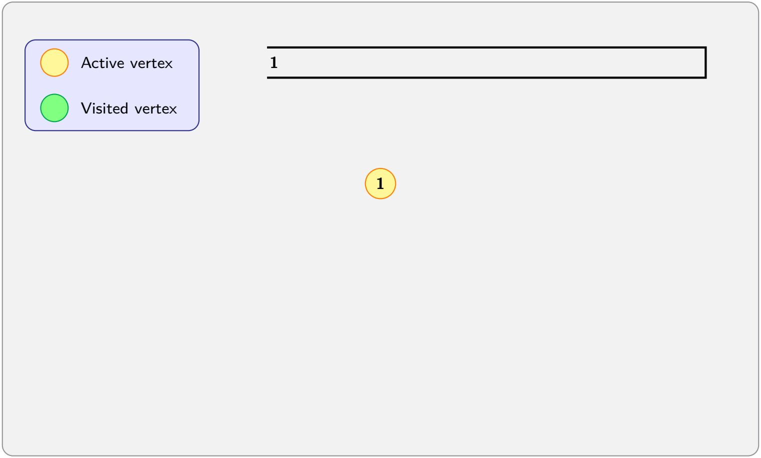
First-In-First-Out-Branch and Bound (FIFO-BB)



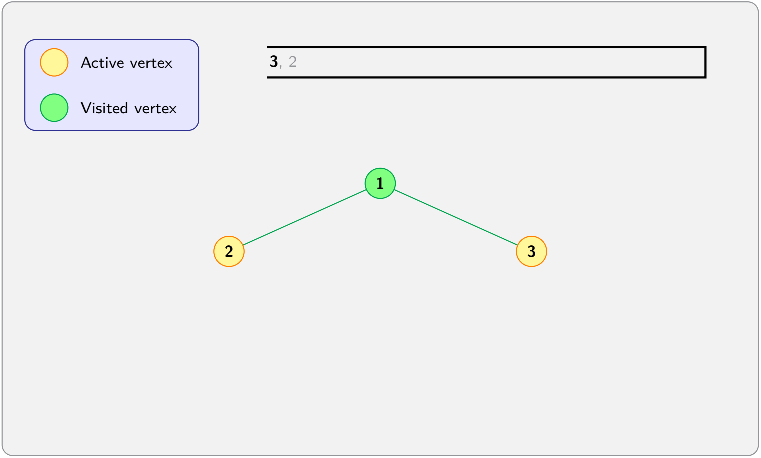
Last-In-First-Out-Branch and Bound (LIFO-BB)



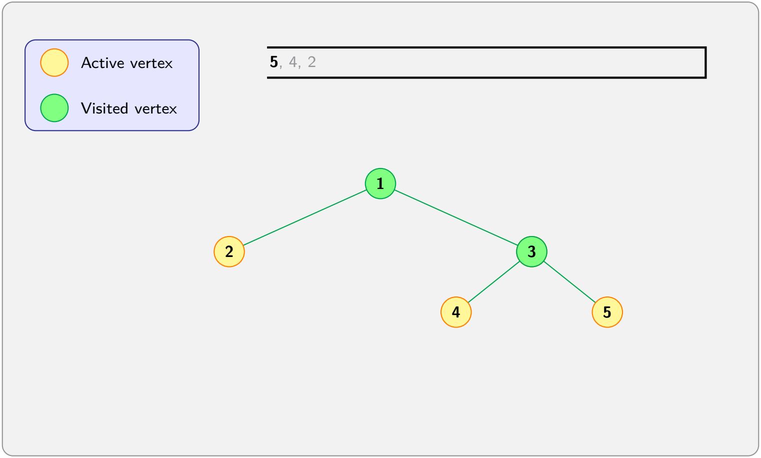
Last-In-First-Out-Branch and Bound (LIFO-BB)



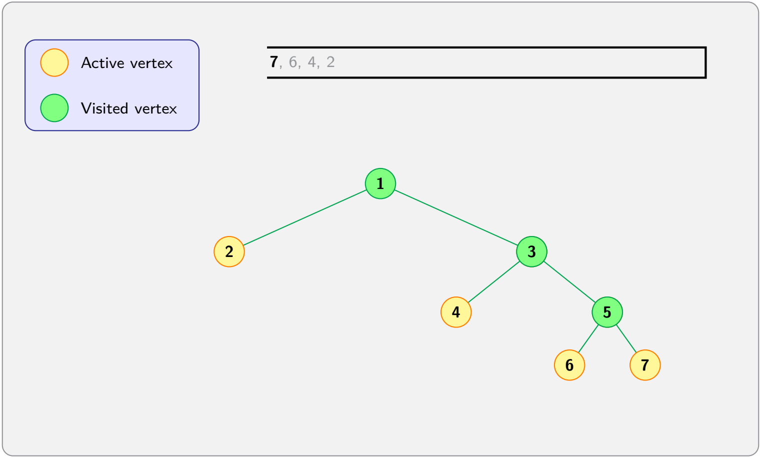
Last-In-First-Out-Branch and Bound (LIFO-BB)



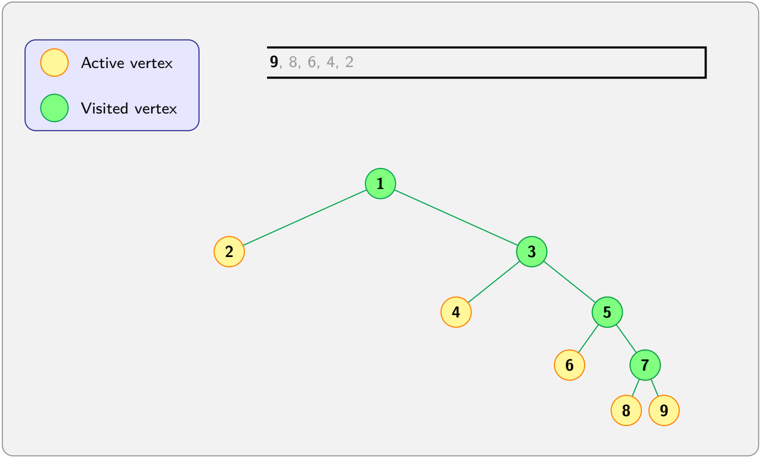
Last-In-First-Out-Branch and Bound (LIFO-BB)



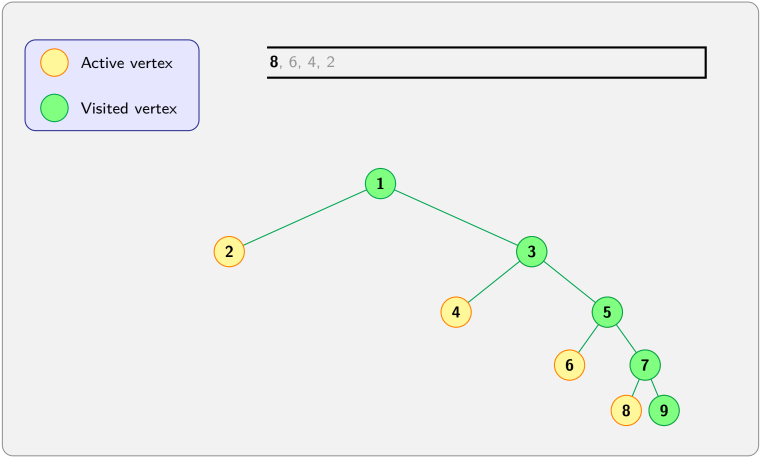
Last-In-First-Out-Branch and Bound (LIFO-BB)



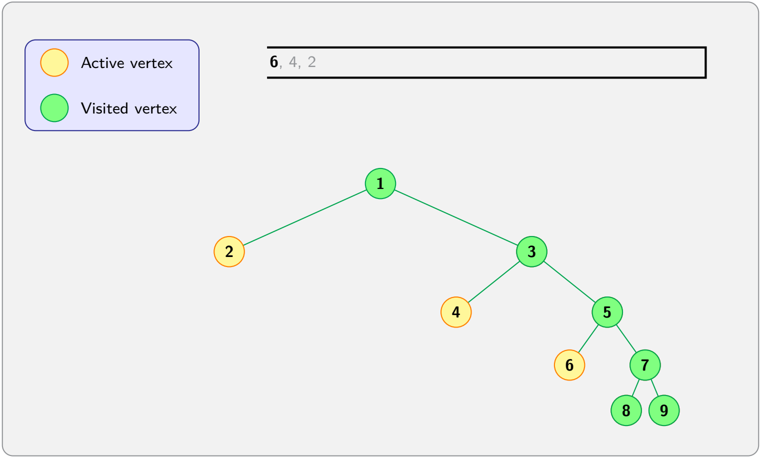
Last-In-First-Out-Branch and Bound (LIFO-BB)



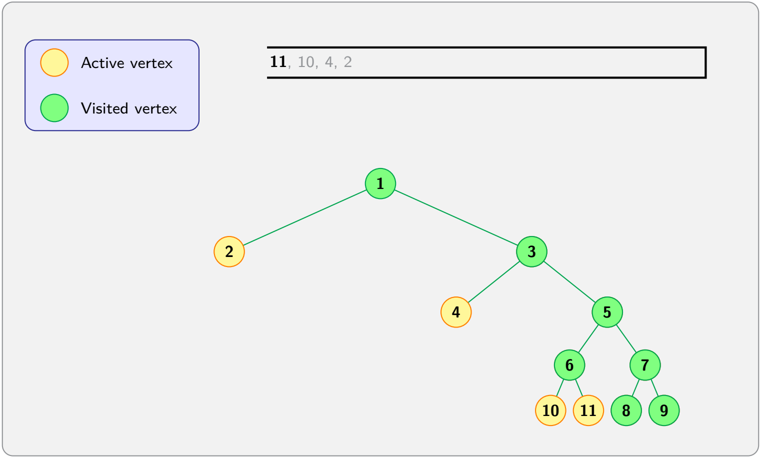
Last-In-First-Out-Branch and Bound (LIFO-BB)



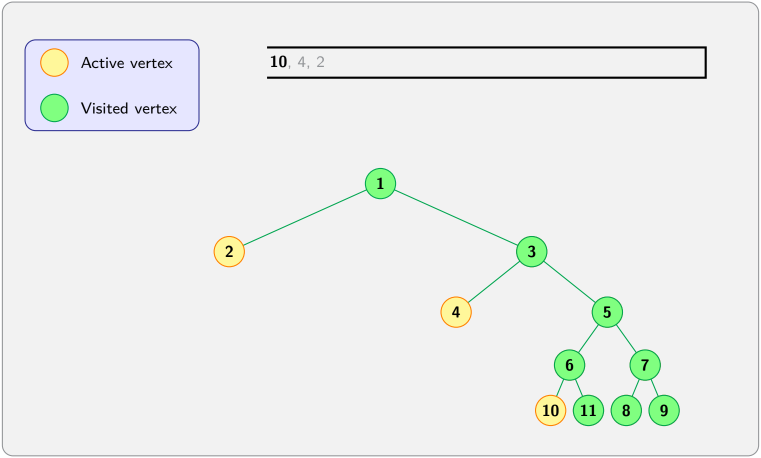
Last-In-First-Out-Branch and Bound (LIFO-BB)



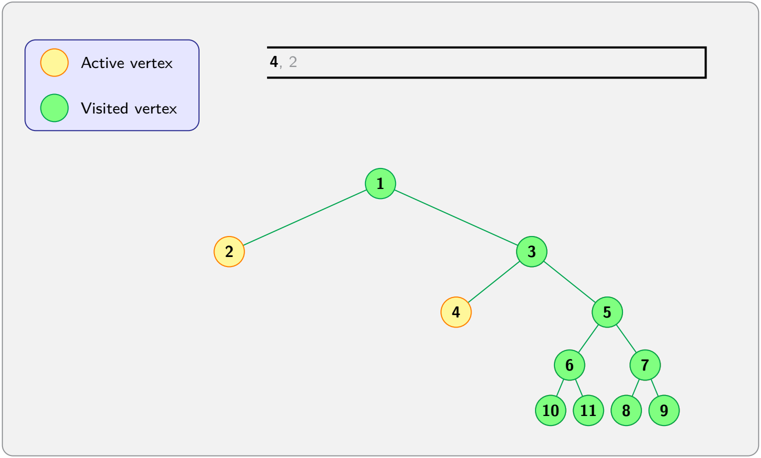
Last-In-First-Out-Branch and Bound (LIFO-BB)



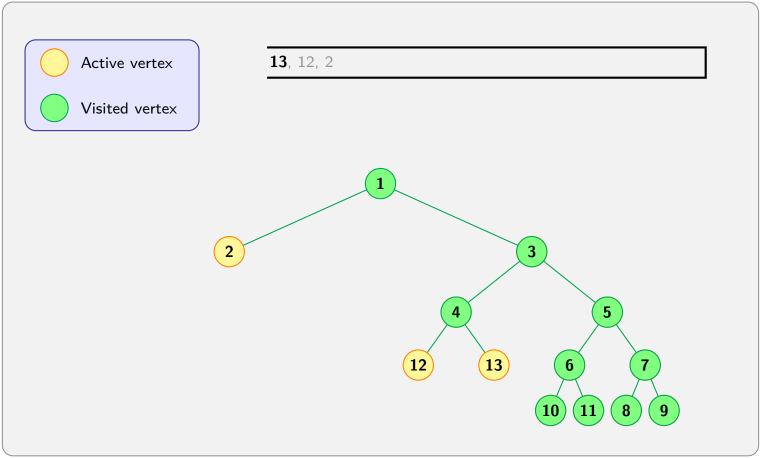
Last-In-First-Out-Branch and Bound (LIFO-BB)



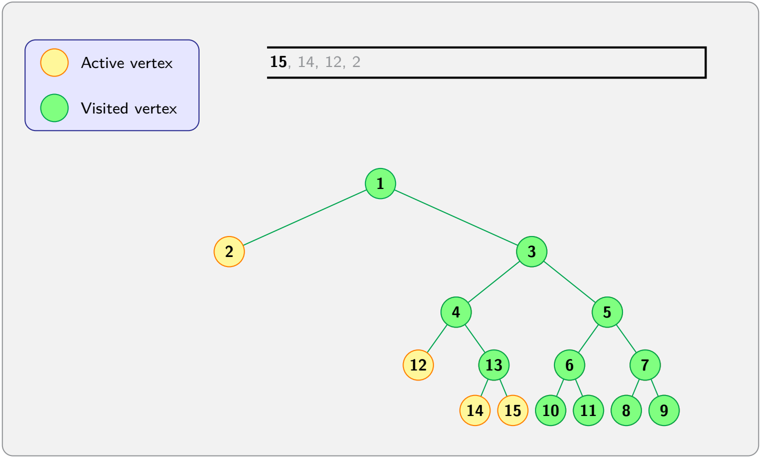
Last-In-First-Out-Branch and Bound (LIFO-BB)



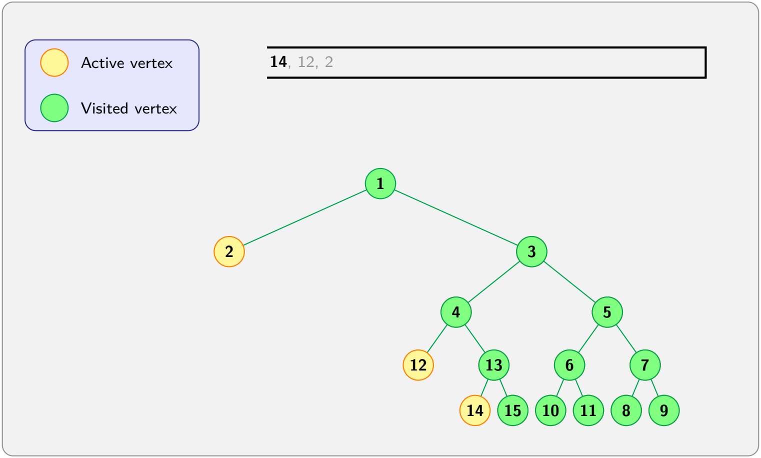
Last-In-First-Out-Branch and Bound (LIFO-BB)



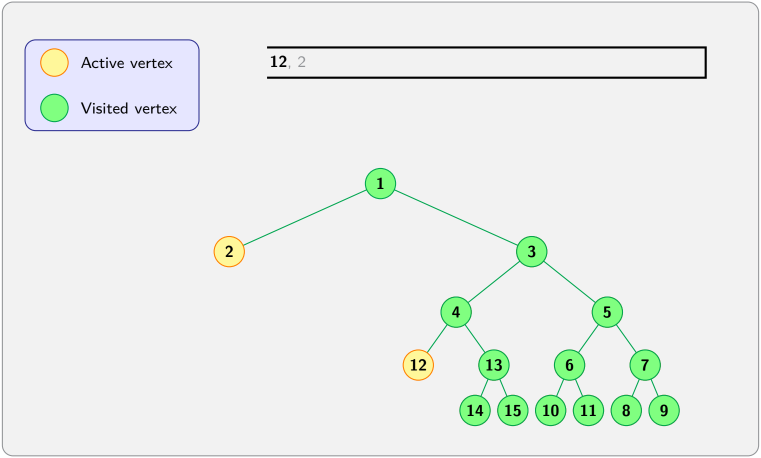
Last-In-First-Out-Branch and Bound (LIFO-BB)



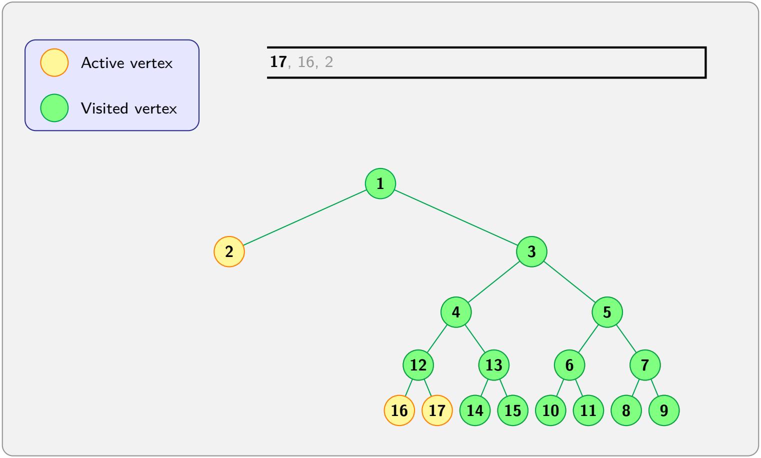
Last-In-First-Out-Branch and Bound (LIFO-BB)



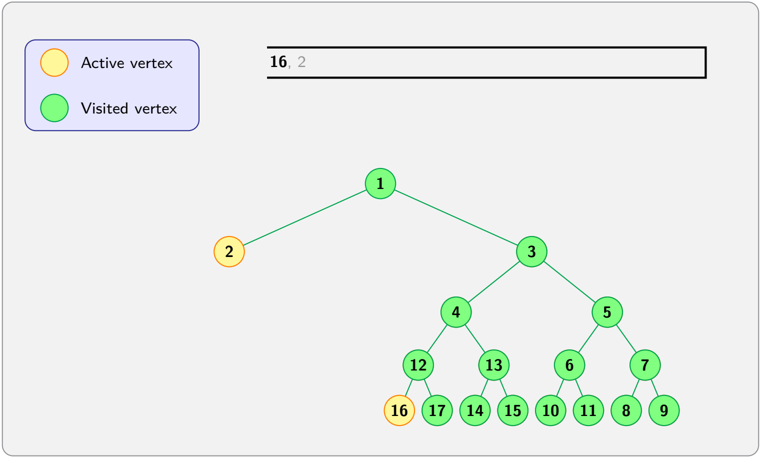
Last-In-First-Out-Branch and Bound (LIFO-BB)



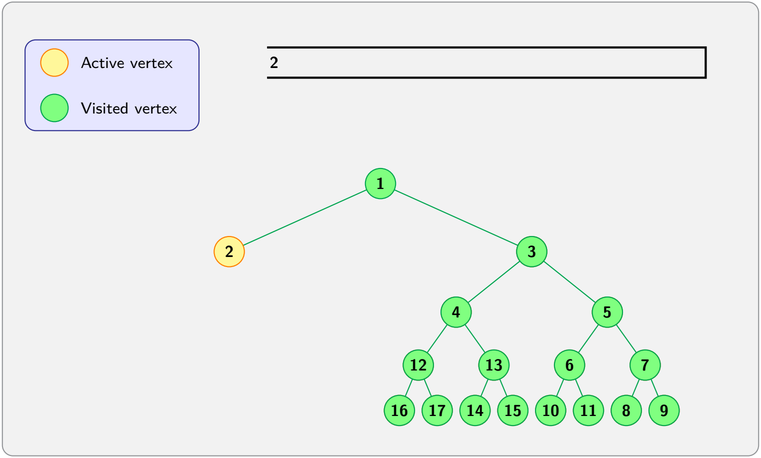
Last-In-First-Out-Branch and Bound (LIFO-BB)



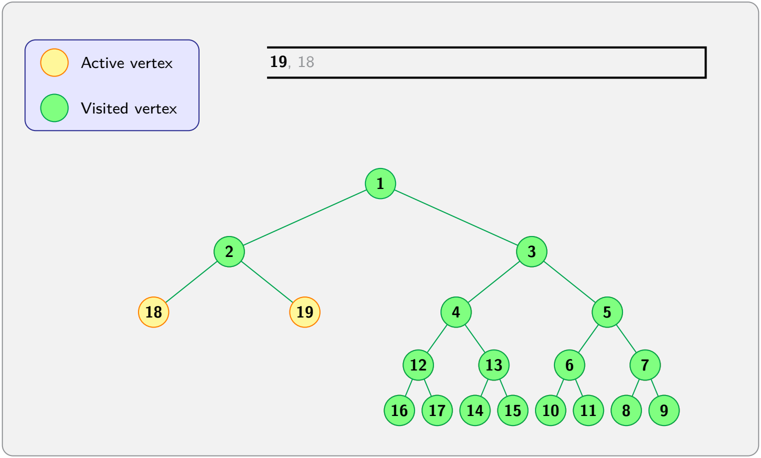
Last-In-First-Out-Branch and Bound (LIFO-BB)



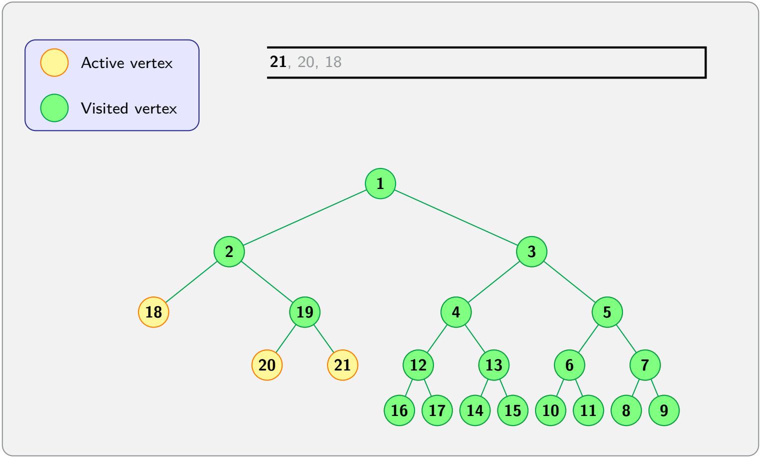
Last-In-First-Out-Branch and Bound (LIFO-BB)



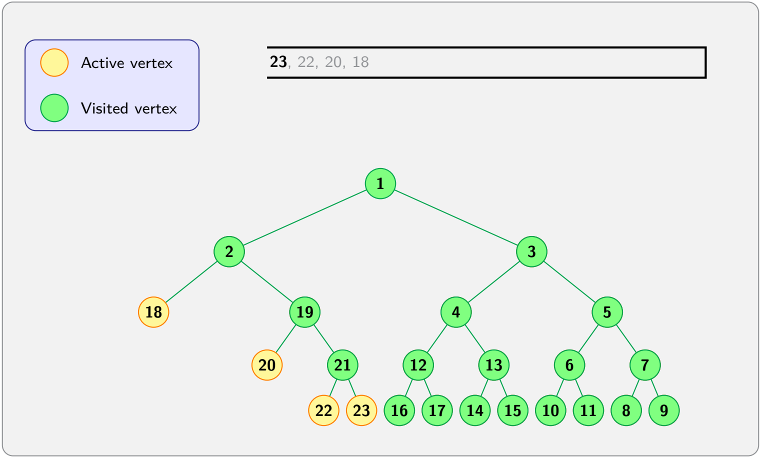
Last-In-First-Out-Branch and Bound (LIFO-BB)



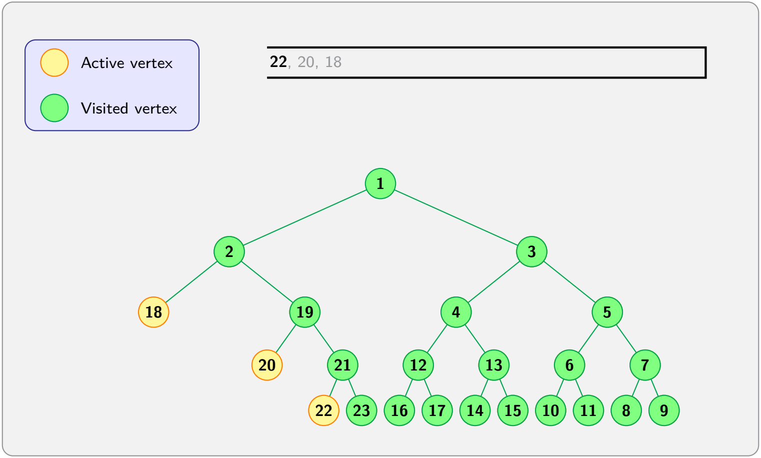
Last-In-First-Out-Branch and Bound (LIFO-BB)



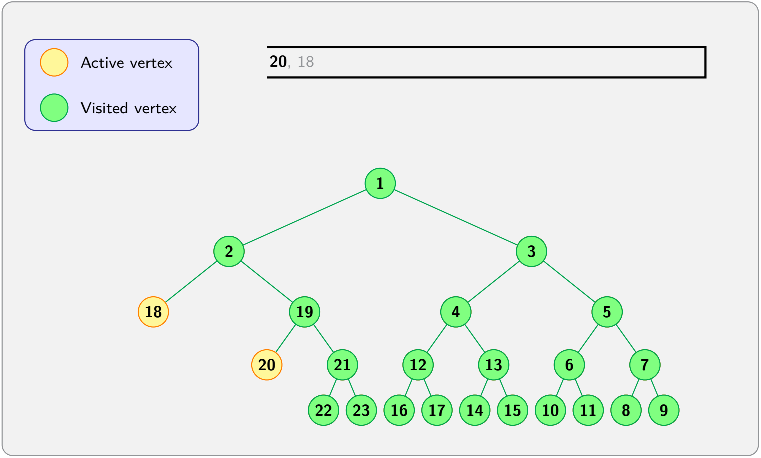
Last-In-First-Out-Branch and Bound (LIFO-BB)



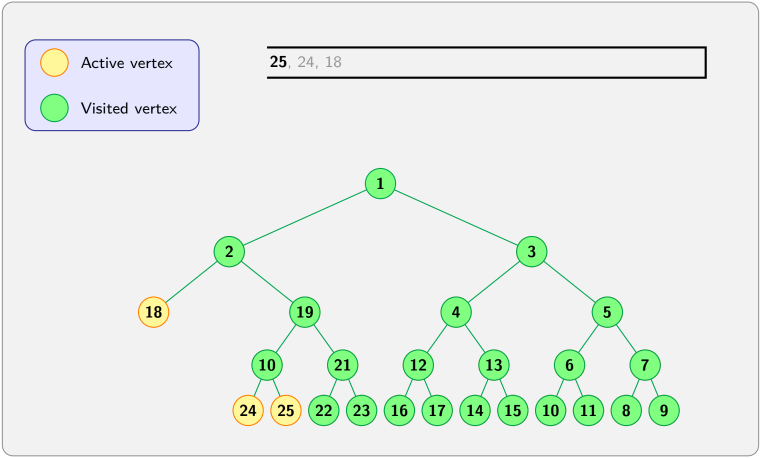
Last-In-First-Out-Branch and Bound (LIFO-BB)



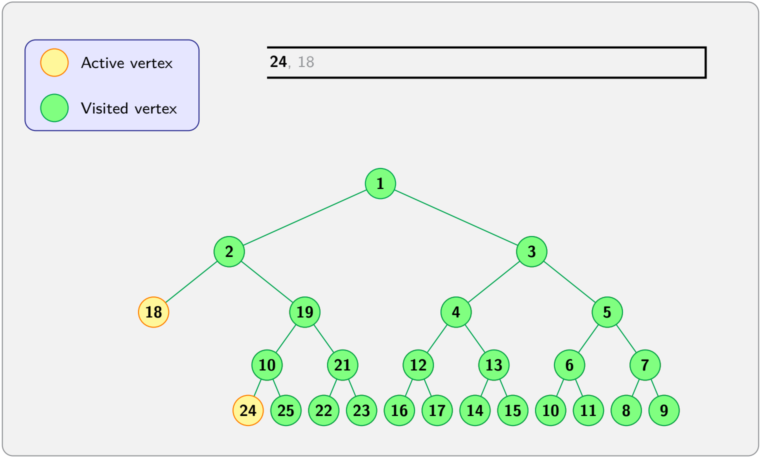
Last-In-First-Out-Branch and Bound (LIFO-BB)



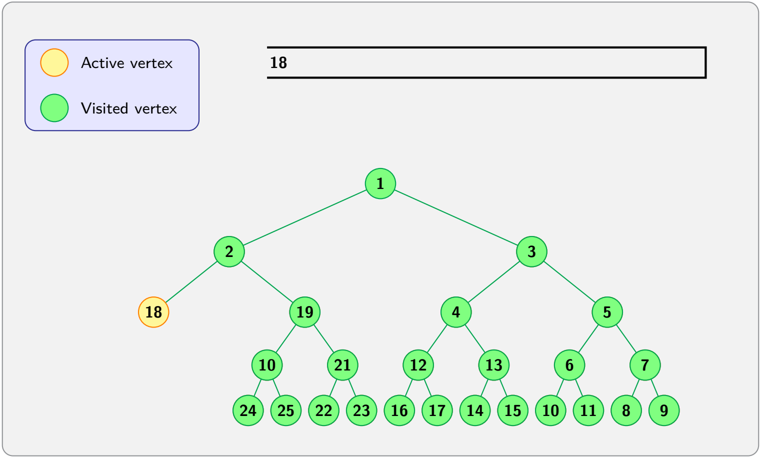
Last-In-First-Out-Branch and Bound (LIFO-BB)



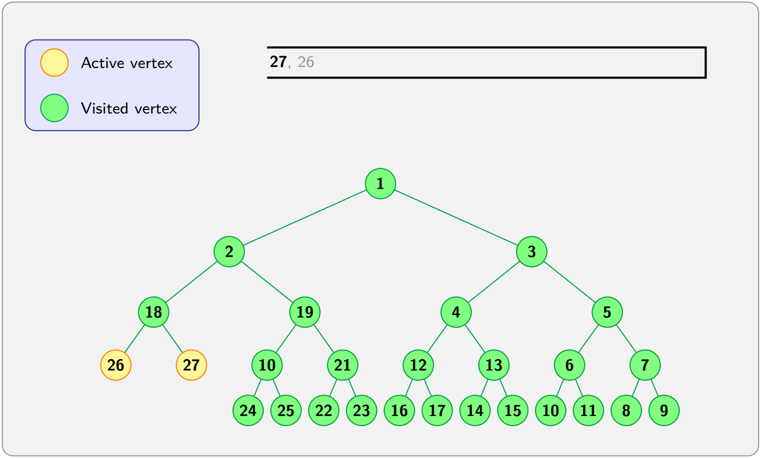
Last-In-First-Out-Branch and Bound (LIFO-BB)



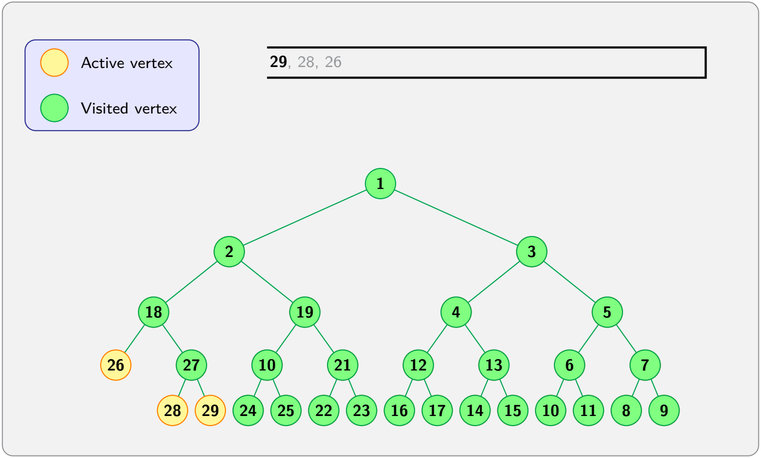
Last-In-First-Out-Branch and Bound (LIFO-BB)



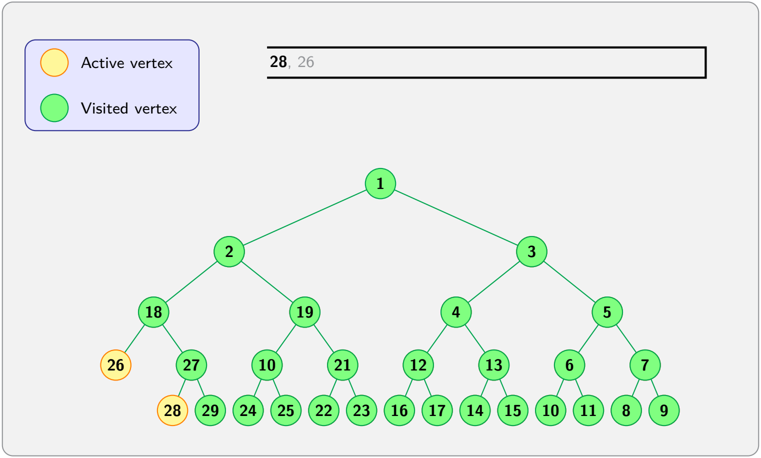
Last-In-First-Out-Branch and Bound (LIFO-BB)



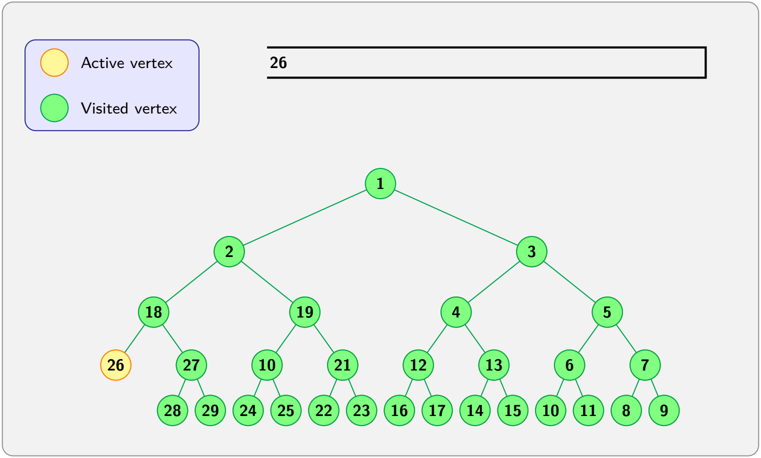
Last-In-First-Out-Branch and Bound (LIFO-BB)



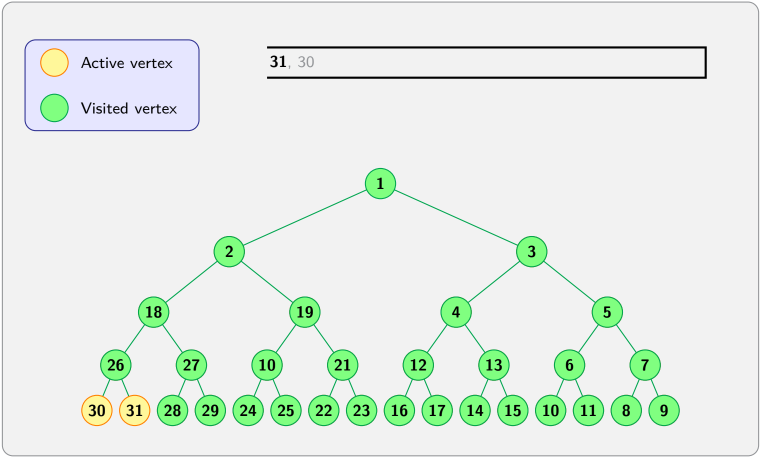
Last-In-First-Out-Branch and Bound (LIFO-BB)



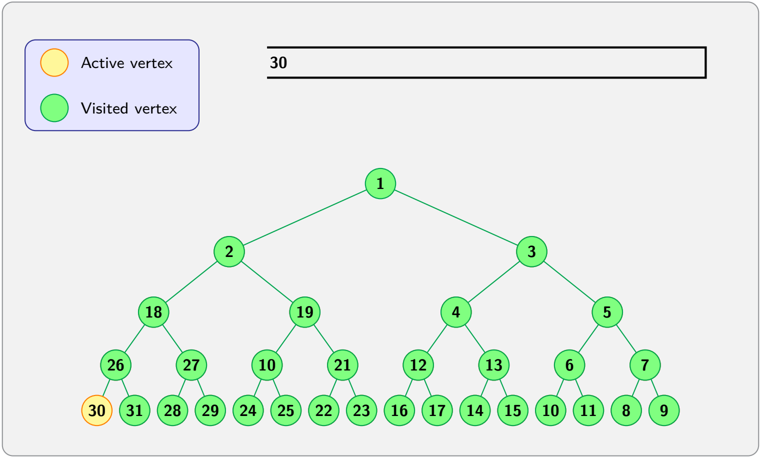
Last-In-First-Out-Branch and Bound (LIFO-BB)



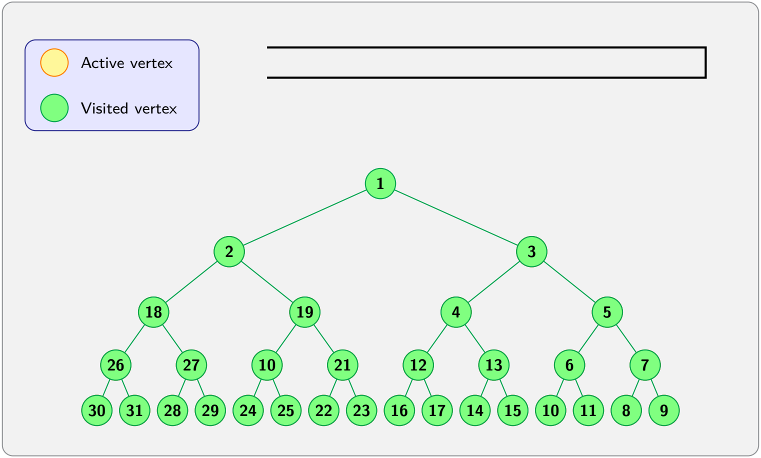
Last-In-First-Out-Branch and Bound (LIFO-BB)



Last-In-First-Out-Branch and Bound (LIFO-BB)



Last-In-First-Out-Branch and Bound (LIFO-BB)



1. Review of the Previous Class
2. Branch and Bound
   - FIFO-BB
   - LIFO-BB
   - LC-BB

The Problem of Job Sequencing with Deadlines

The Problem of Job Sequencing with Deadlines

- There is a set of jobs j_i, each one with a time duration t_i, a deadline d_i and a penalty p_i if the job is not finished by deadline d_i.

The Problem of Job Sequencing with Deadlines

- There is a set of jobs j_i, each one with a time duration t_i, a deadline d_i and a penalty p_i if the job is not finished by deadline d_i.
- The goal is to minimise the penalty paid.

The Problem of Job Sequencing with Deadlines

- There is a set of jobs j_i, each one with a time duration t_i, a deadline d_i and a penalty p_i if the job is not finished by deadline d_i.
- The goal is to minimise the penalty paid.
- Each level corresponds to a job; we decide whether to include it or not. The inclusion must be feasible (i.e., it should be possible to complete the jobs in due time.)

The Problem of Job Sequencing with Deadlines

- There is a set of jobs j_i, each one with a time duration t_i, a deadline d_i and a penalty p_i if the job is not finished by deadline d_i.
- The goal is to minimise the penalty paid.
- Each level corresponds to a job; we decide whether to include it or not. The inclusion must be feasible (i.e., it should be possible to complete the jobs in due time.)
- For each node we define two values: cost (c) and upper bound (u).

The Problem of Job Sequencing with Deadlines

- There is a set of jobs j_i, each one with a time duration t_i, a deadline d_i and a penalty p_i if the job is not finished by deadline d_i.
- The goal is to minimise the penalty paid.
- Each level corresponds to a job; we decide whether to include it or not. The inclusion must be feasible (i.e., it should be possible to complete the jobs in due time.)
- For each node we define two values: cost (c) and upper bound (u).
- Cost is defined as the sum of all penalties up to the last job considered.

The Problem of Job Sequencing with Deadlines

- There is a set of jobs j_i, each one with a time duration t_i, a deadline d_i and a penalty p_i if the job is not finished by deadline d_i.
- The goal is to minimise the penalty paid.
- Each level corresponds to a job; we decide whether to include it or not. The inclusion must be feasible (i.e., it should be possible to complete the jobs in due time.)
- For each node we define two values: cost (c) and upper bound (u).
- Cost is defined as the sum of all penalties up to the last job considered.
- The upper bound will be the sum of all penalties except those already included in the solution.

Least Cost Branch and Bound (LC-BB)

Least Cost Branch and Bound (LC-BB)

Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

Least Cost Branch and Bound (LC-BB)

Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

Optimal vertex

Active vertex

Visited vertex

Killed vertex

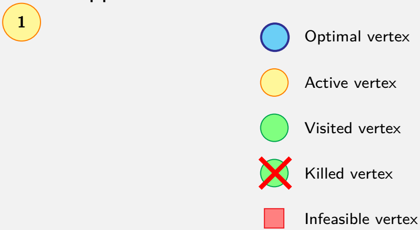
Infeasible vertex

Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper $= 24$

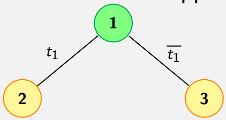


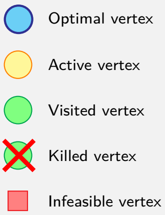
Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper $= 24$



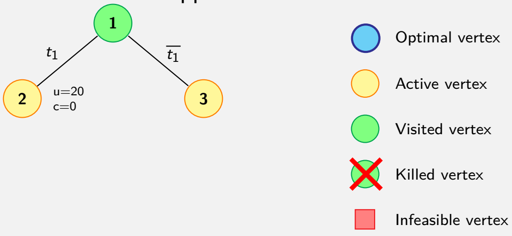


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper = 24

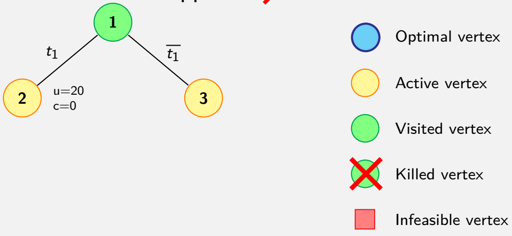


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper = $\cancel{24}$ 20

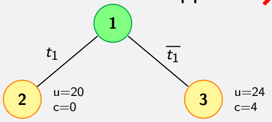


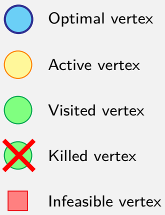
Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper = ~~24~~ 20



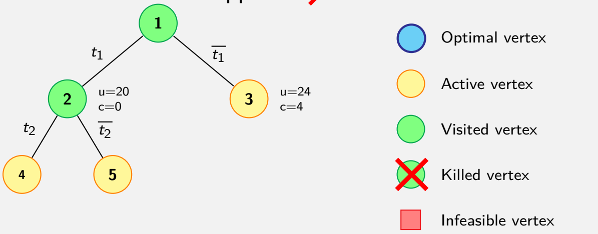


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper $= \cancel{24}$ 20

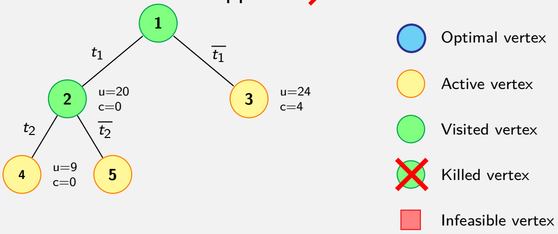


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper $= \cancel{24}\ 20$

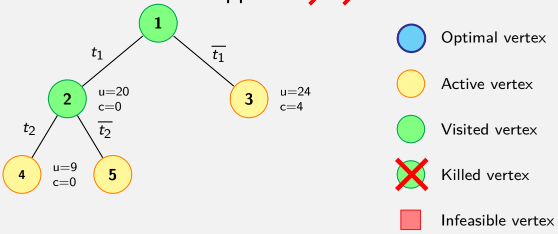


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper $= \cancel{24}\ \cancel{20}\ 9$

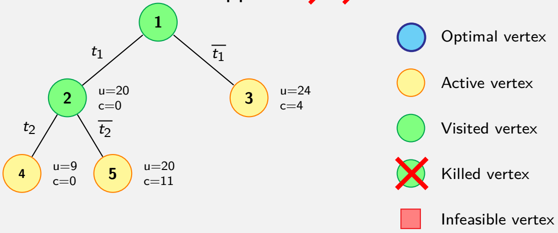


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper $= \cancel{24}\ \cancel{20}\ 9$

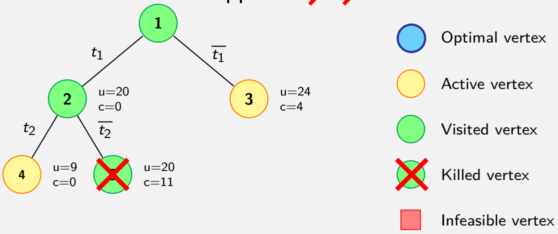


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper $= \cancel{24}\ \cancel{20}\ 9$

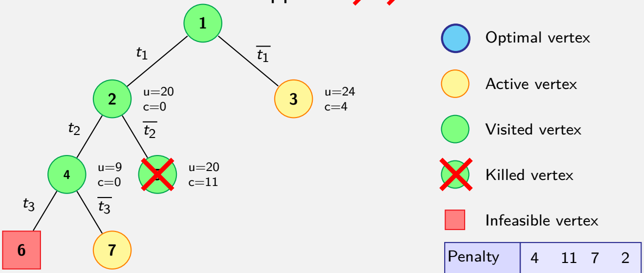


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper $= \cancel{24}\ \cancel{20}\ 9$

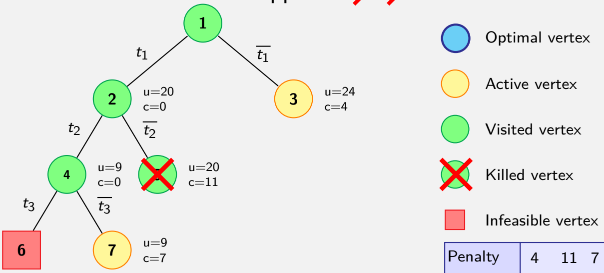


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

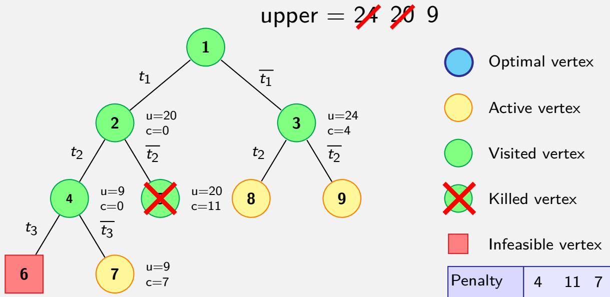
upper = ~~24~~ ~~20~~ 9



Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

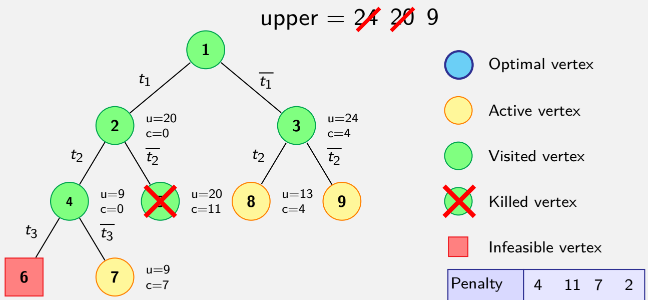
Least Cost Branch and Bound (LC-BB)



Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

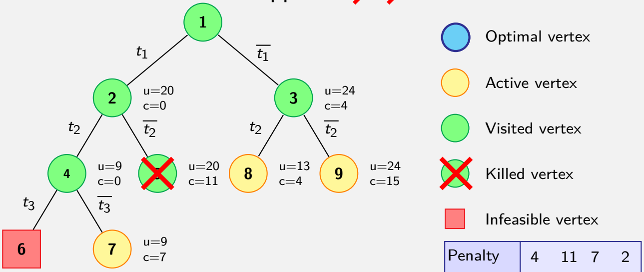


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

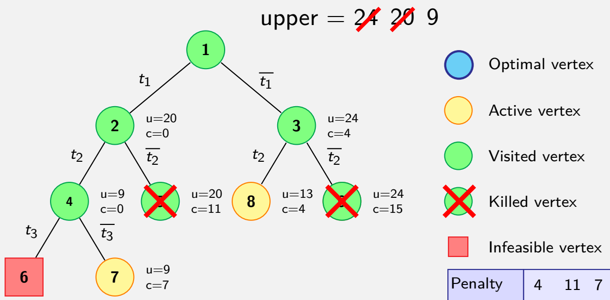
upper $=$ ~~24~~ ~~20~~ 9



Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

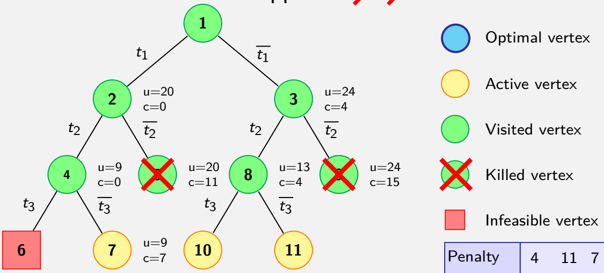


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

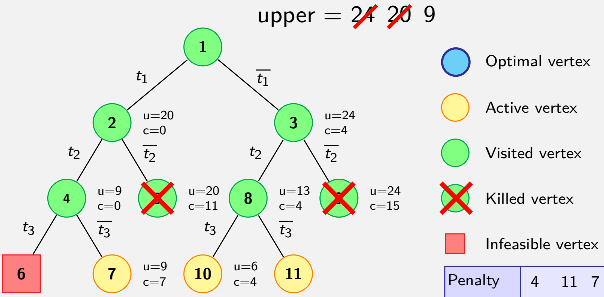
upper = ~~24~~ ~~20~~ 9



Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

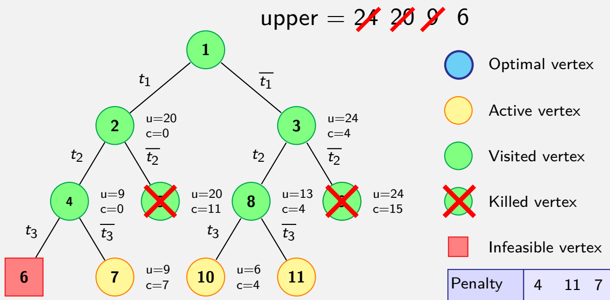
Least Cost Branch and Bound (LC-BB)



Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

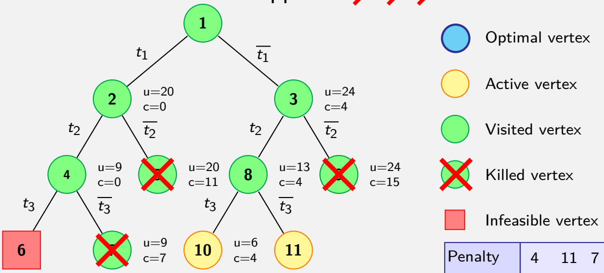


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

$\text{upper} = \cancel{24}\ \cancel{20}\ \cancel{9}\ 6$

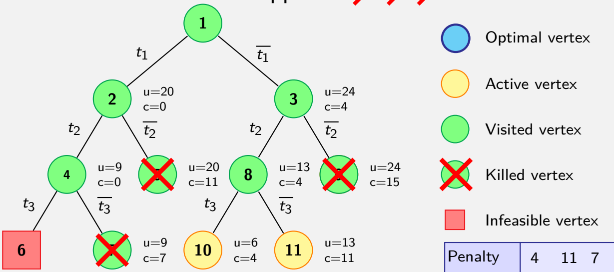


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

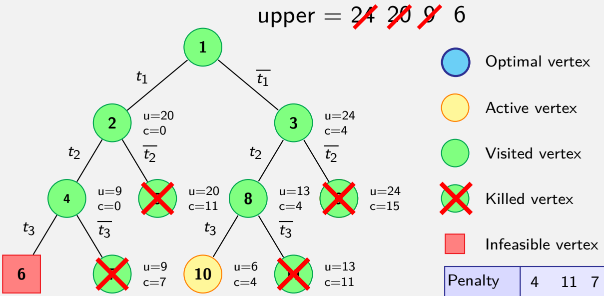
upper = $\cancel{24}$ $\cancel{20}$ $\cancel{9}$ 6



Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

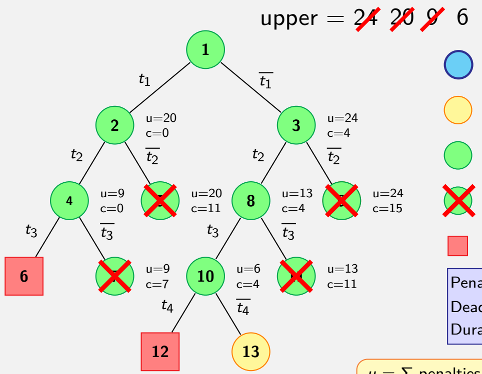
Least Cost Branch and Bound (LC-BB)

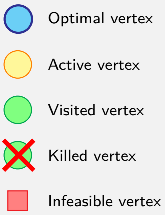


Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

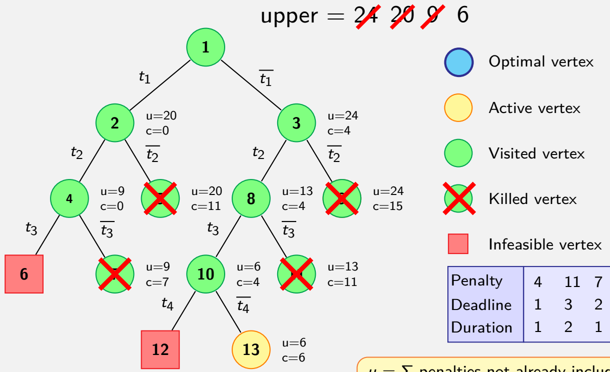




Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

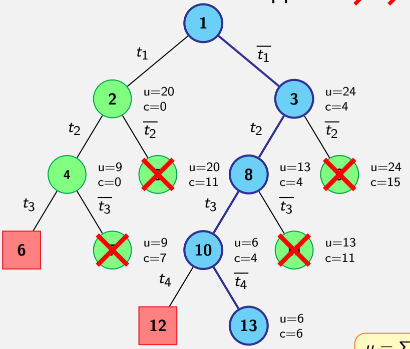


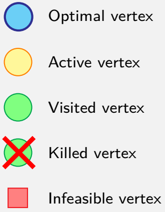
Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far

Least Cost Branch and Bound (LC-BB)

upper $= \cancel{24}\ \cancel{20}\ \cancel{9}\ 6$





Penalty	4	11	7	2
Deadline	1	3	2	1
Duration	1	2	1	1

$u = \Sigma$ penalties not already included
$c = \Sigma$ penalties not included so far