Programming III

Ricardo Wehbe

UADE

13 de noviembre de 2021

Programme

1. A First Example: 0-1 Knapsack
2. A Second Example: Task Assignment
3. A Third Example: Travelling Salesman Problem (TSP)

A First Example: 0-1 Knapsack

- This is the same problem we have already seen in the Dynamic Programming part of the course.

A First Example: 0-1 Knapsack

- This is the same problem we have already seen in the Dynamic Programming part of the course.
- Le us recall: there is a set of objects o_i, each one having a weight w_i and a value v_i. We have a knapsack of maximum capacity c and we want to maximise the value we put therein without exceeding the capacity of the knapsack.

A First Example: 0-1 Knapsack

- This is the same problem we have already seen in the Dynamic Programming part of the course.
- Le us recall: there is a set of objects o_i, each one having a weight w_i and a value v_i. We have a knapsack of maximum capacity c and we want to maximise the value we put therein without exceeding the capacity of the knapsack.
- Since this is a maximisation problem, we consider the values as negative.

A First Example: 0-1 Knapsack

- This is the same problem we have already seen in the Dynamic Programming part of the course.
- Le us recall: there is a set of objects o_i, each one having a weight w_i and a value v_i. We have a knapsack of maximum capacity c and we want to maximise the value we put therein without exceeding the capacity of the knapsack.
- Since this is a maximisation problem, we consider the values as negative.
- For each node we will compute two magnitudes: upper bound (u) and cost (c.)

A First Example: 0-1 Knapsack

- This is the same problem we have already seen in the Dynamic Programming part of the course.
- Le us recall: there is a set of objects o_i, each one having a weight w_i and a value v_i. We have a knapsack of maximum capacity c and we want to maximise the value we put therein without exceeding the capacity of the knapsack.
- Since this is a maximisation problem, we consider the values as negative.
- For each node we will compute two magnitudes: upper bound (u) and cost (c.)
- The upper bound of the node will be the sum of the values not already included; the cost will be the sum of the values not already included *with fractions*.

A First Example: 0-1 Knapsack

object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

A First Example: 0-1 Knapsack

object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$\boldsymbol{m} = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

A First Example: 0-1 Knapsack

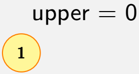
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$\boldsymbol{m} = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

- Active vertex
- Visited vertex
- Killed vertex
- Optimal vertex
- Infeasible vertex

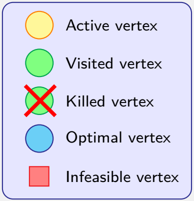
A First Example: 0-1 Knapsack



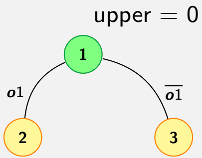
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$\boldsymbol{m} = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



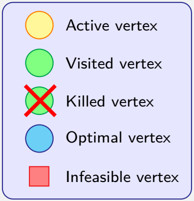
A First Example: 0-1 Knapsack



object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

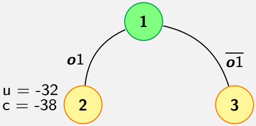
$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



A First Example: 0-1 Knapsack

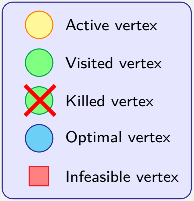
upper $= 0$



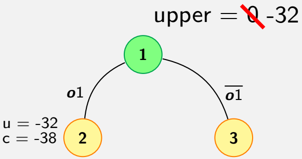
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



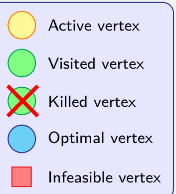
A First Example: 0-1 Knapsack



object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

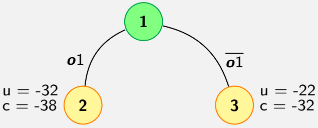
$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



A First Example: 0-1 Knapsack

upper = ~~0~~ -32



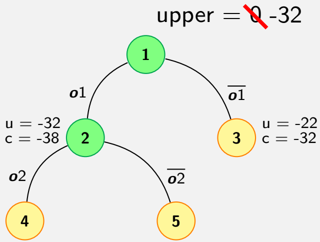
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

- Active vertex
- Visited vertex
- Killed vertex
- Optimal vertex
- Infeasible vertex

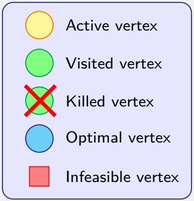
A First Example: 0-1 Knapsack



object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

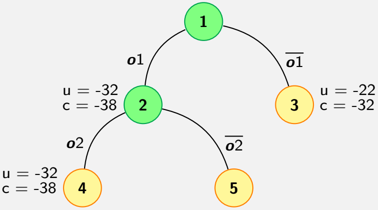
$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



A First Example: 0-1 Knapsack

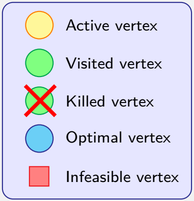
upper = ~~0~~ -32



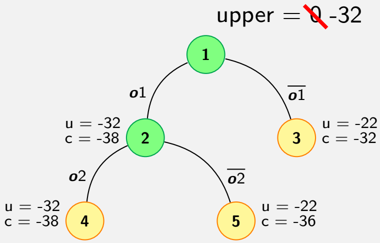
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
m = 15				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



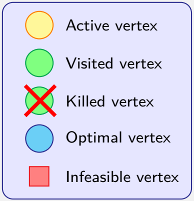
A First Example: 0-1 Knapsack



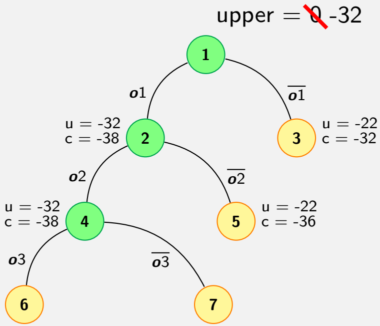
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



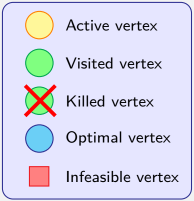
A First Example: 0-1 Knapsack



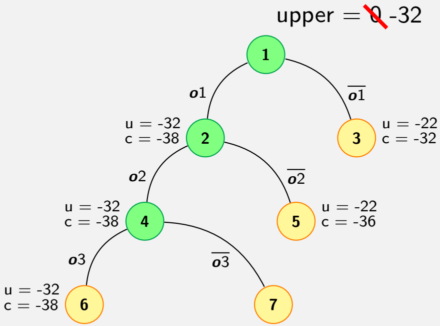
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



A First Example: 0-1 Knapsack



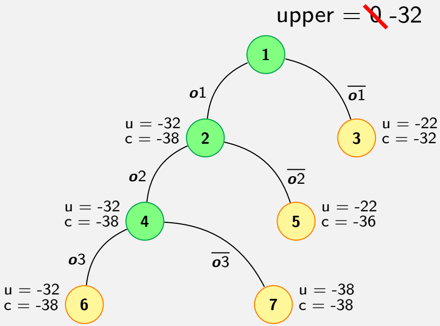
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

- Active vertex
- Visited vertex
- Killed vertex
- Optimal vertex
- Infeasible vertex

A First Example: 0-1 Knapsack



object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

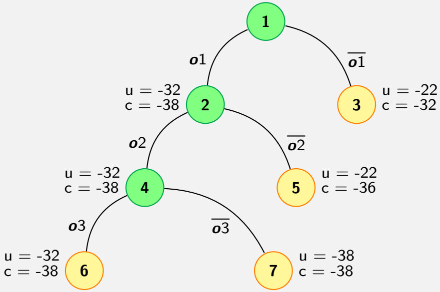
$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

- Active vertex
- Visited vertex
- Killed vertex
- Optimal vertex
- Infeasible vertex

A First Example: 0-1 Knapsack

upper = ~~0~~ ~~-32~~ -38



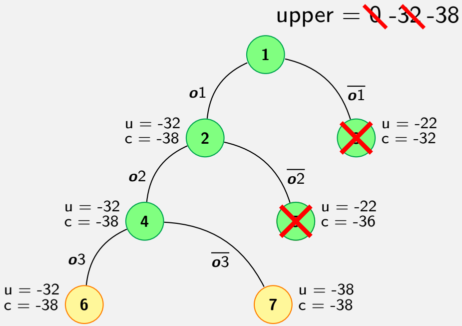
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

- Active vertex
- Visited vertex
- Killed vertex
- Optimal vertex
- Infeasible vertex

A First Example: 0-1 Knapsack



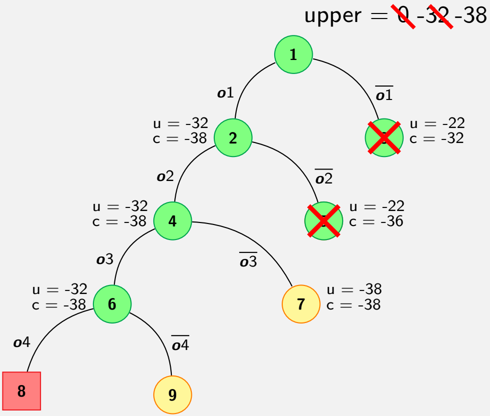
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

- Active vertex
- Visited vertex
- Killed vertex
- Optimal vertex
- Infeasible vertex

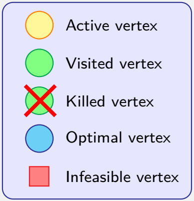
A First Example: 0-1 Knapsack



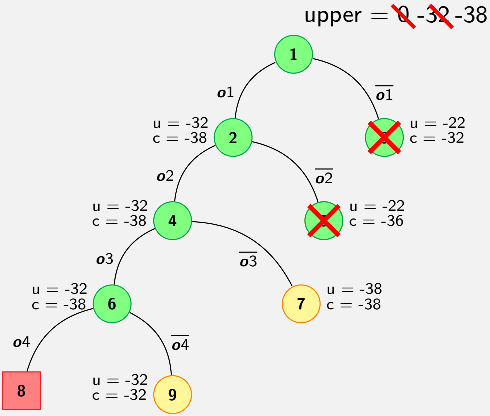
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



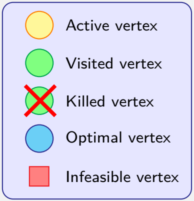
A First Example: 0-1 Knapsack



object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

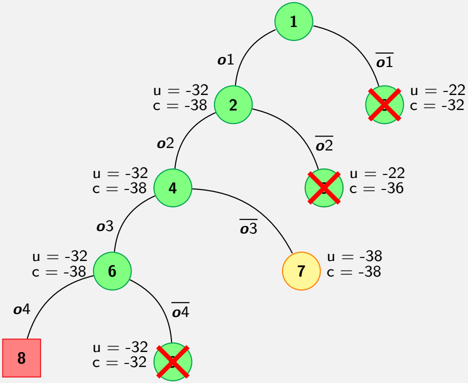
$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



A First Example: 0-1 Knapsack

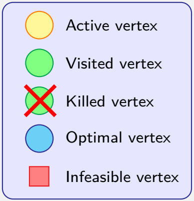
upper = ~~0~~ ~~-32~~ -38



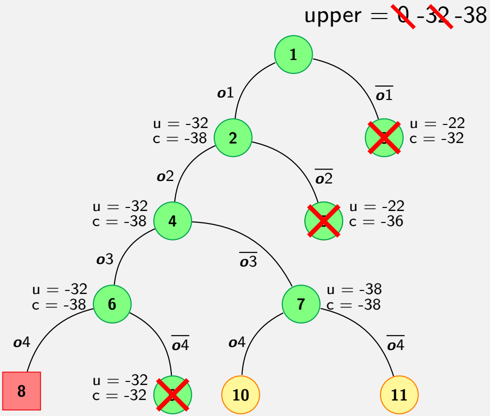
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



A First Example: 0-1 Knapsack



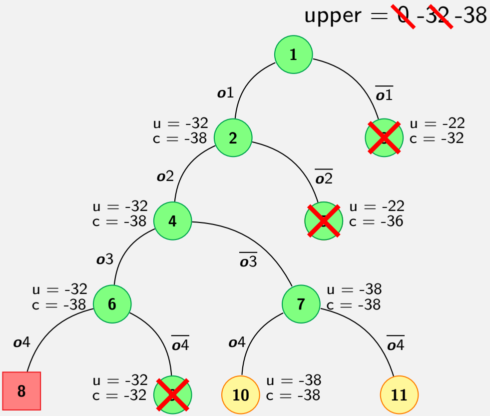
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

- Active vertex
- Visited vertex
- Killed vertex
- Optimal vertex
- Infeasible vertex

A First Example: 0-1 Knapsack



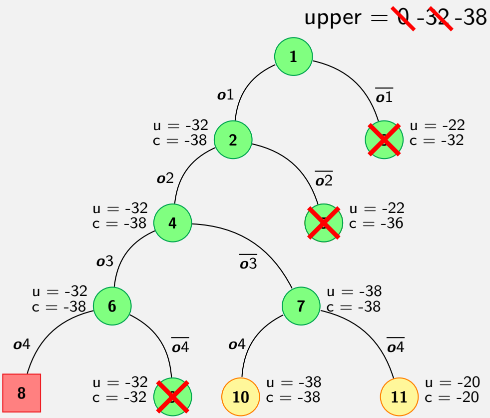
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

- Active vertex
- Visited vertex
- Killed vertex
- Optimal vertex
- Infeasible vertex

A First Example: 0-1 Knapsack



object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

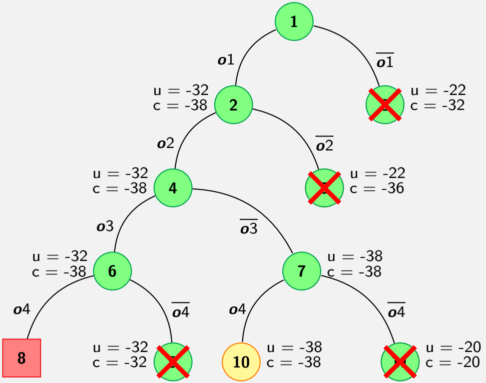
$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)

- Active vertex
- Visited vertex
- Killed vertex
- Optimal vertex
- Infeasible vertex

A First Example: 0-1 Knapsack

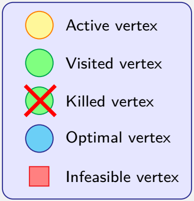
upper = ~~0~~ ~~-32~~ -38



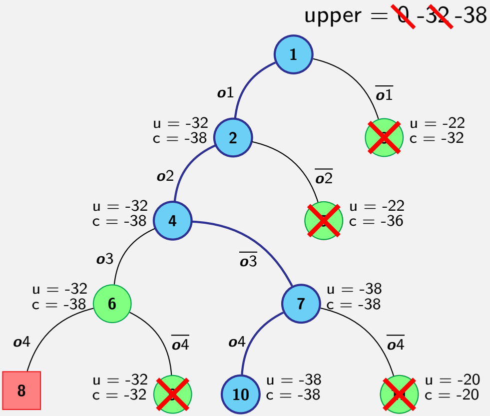
object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



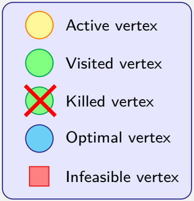
A First Example: 0-1 Knapsack



object	$o1$	$o2$	$o3$	$o4$
value	10	10	12	18
weight	2	4	6	9
$m = 15$				

$u = \Sigma$ values not yet included

$c = \Sigma$ values not yet included (with fractions)



A Second Example: Task Assignment

A Second Example: Task Assignment

- Here we have a set of n jobs that must be done. We have also a team of n persons. Each person can perform any of the jobs at a given cost.

A Second Example: Task Assignment

- Here we have a set of n jobs that must be done. We have also a team of n persons. Each person can perform any of the jobs at a given cost.
- Each person must be assigned to exactly one job. We want to do so with the minimum cost.

A Second Example: Task Assignment

- Here we have a set of n jobs that must be done. We have also a team of n persons. Each person can perform any of the jobs at a given cost.
- Each person must be assigned to exactly one job. We want to do so with the minimum cost.
- The variable *upper* will be initially assigned the cost of an arbitrary assignment.

A Second Example: Task Assignment

- Here we have a set of n jobs that must be done. We have also a team of n persons. Each person can perform any of the jobs at a given cost.
- Each person must be assigned to exactly one job. We want to do so with the minimum cost.
- The variable *upper* will be initially assigned the cost of an arbitrary assignment.
- For each node we will calculate a cost (c), which will be the sum of the minimum costs of the tasks not yet assigned and the costs already incurred.

A Second Example: Task Assignment

- Here we have a set of n jobs that must be done. We have also a team of n persons. Each person can perform any of the jobs at a given cost.
- Each person must be assigned to exactly one job. We want to do so with the minimum cost.
- The variable *upper* will be initially assigned the cost of an arbitrary assignment.
- For each node we will calculate a cost (c), which will be the sum of the minimum costs of the tasks not yet assigned and the costs already incurred.
- The *upper* variable will be only updated when we have completed some assignment that has a better cost.

A Second Example: Task Assignement

A Second Example: Task Assignement

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

A Second Example: Task Assignement

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex
Optimal vertex
Visited vertex
Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

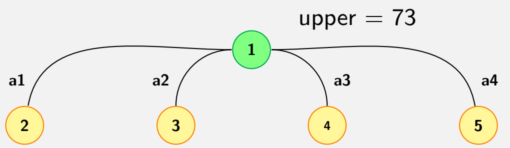
upper $= 73$

	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex
Optimal vertex
Visited vertex
Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

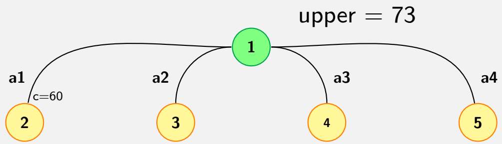


	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex
Optimal vertex
Visited vertex
Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

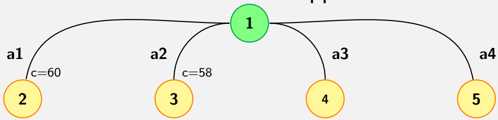


	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

upper = 73

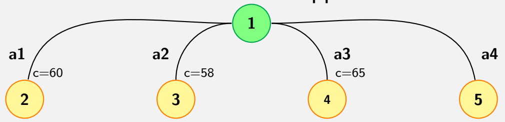


	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

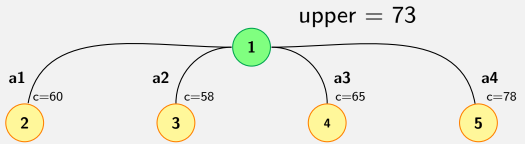
upper = 73



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

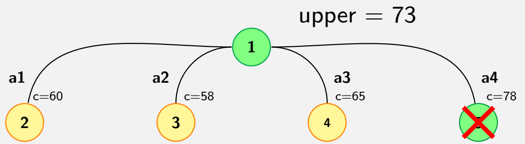
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

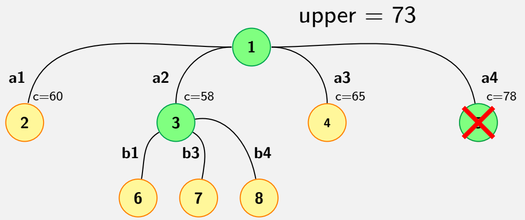
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement



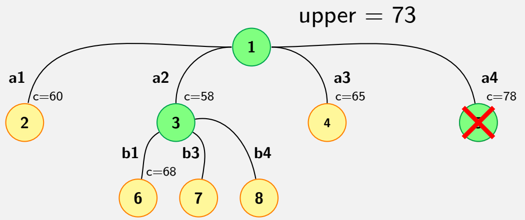
	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex   Optimal vertex

Visited vertex   Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement



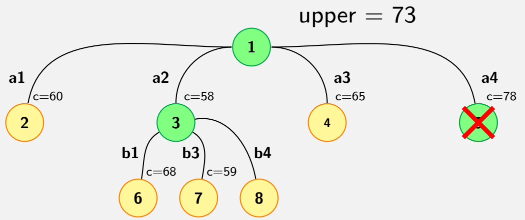
	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex   Optimal vertex

Visited vertex   Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement



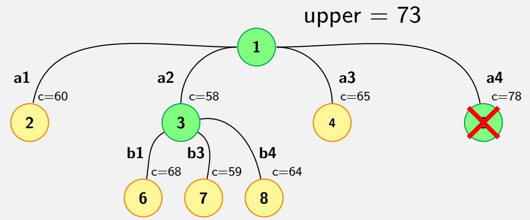
	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex   Optimal vertex

Visited vertex   Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement



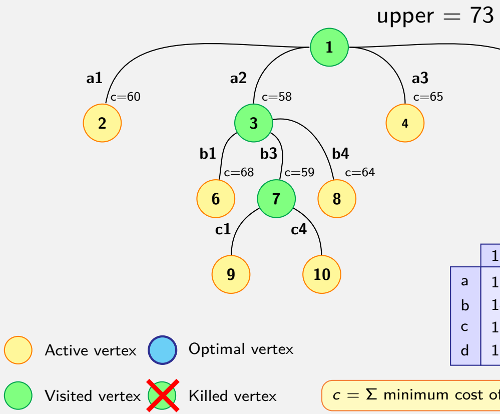
	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex   Optimal vertex

Visited vertex   Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

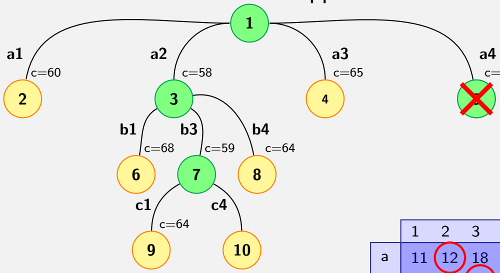


	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

upper = 73



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

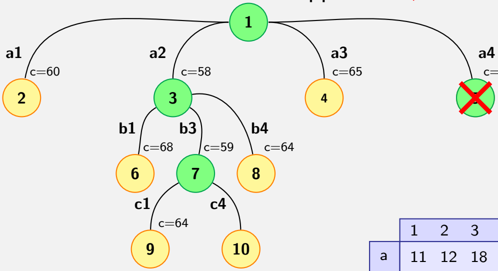
Active vertex Optimal vertex

Visited vertex Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

upper = ~~73~~ 64



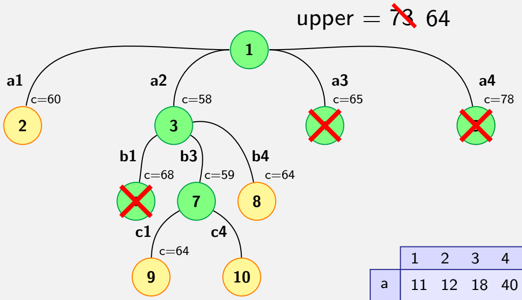
	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex — Optimal vertex

Visited vertex — Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

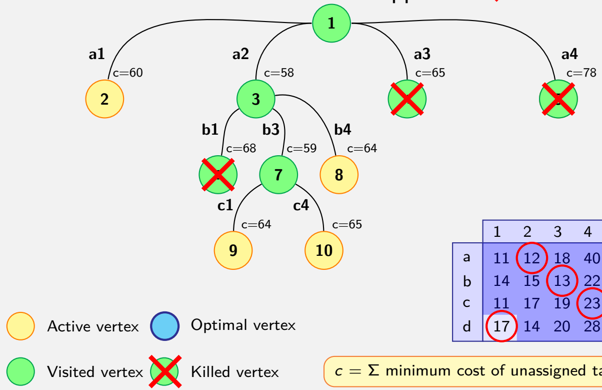


	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement

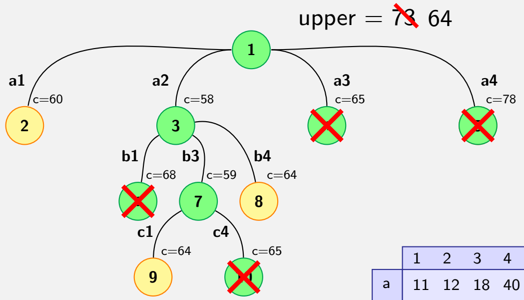
upper = $\cancel{73}$ 64



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

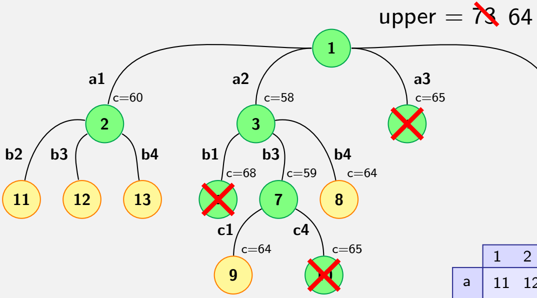
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

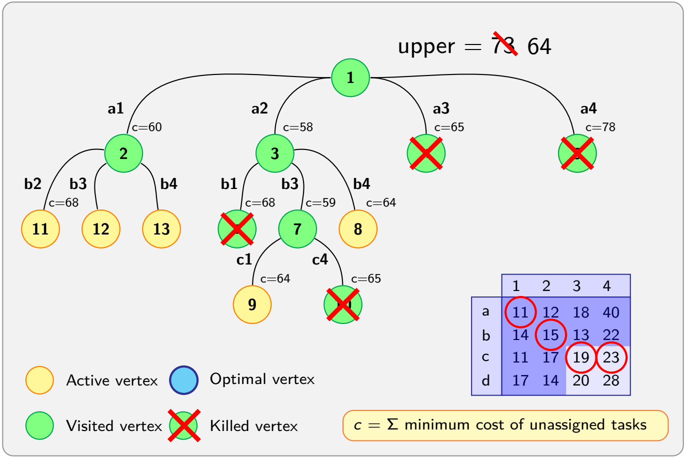
A Second Example: Task Assignement



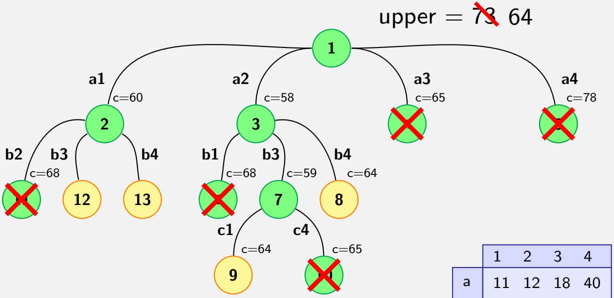
	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement



A Second Example: Task Assignement



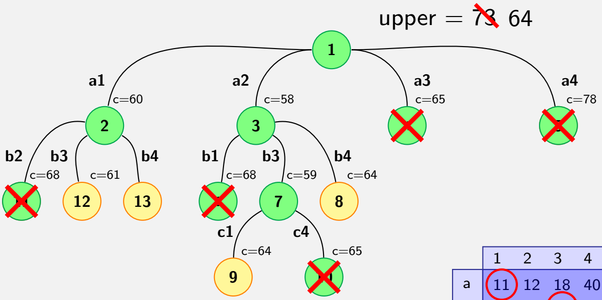
	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex　Optimal vertex

Visited vertex　Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement



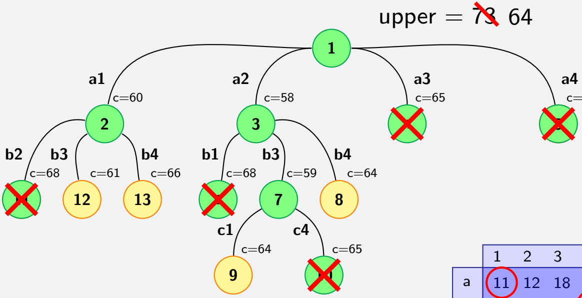
	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

Active vertex Optimal vertex

Visited vertex Killed vertex

$c = \Sigma$ minimum cost of unassigned tasks

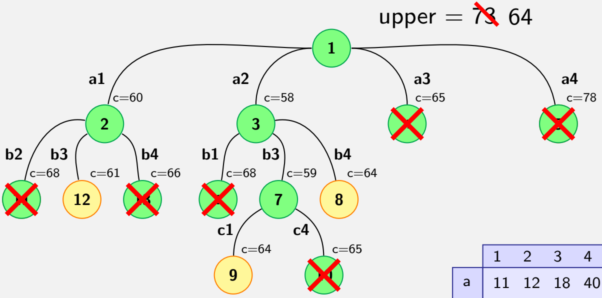
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

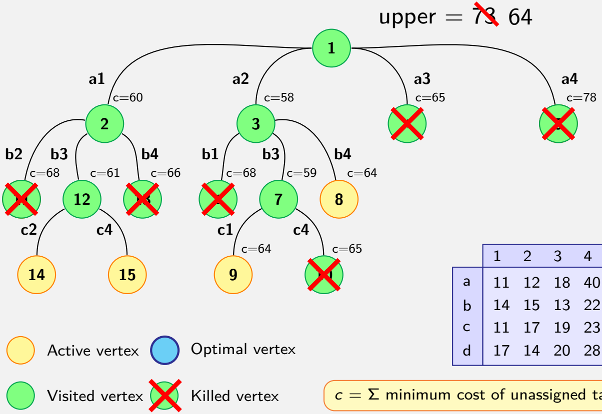
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

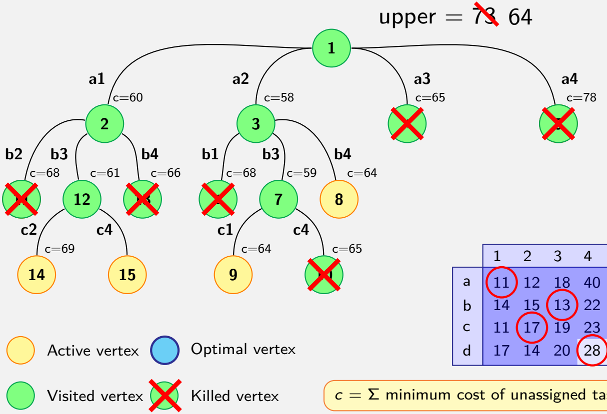
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

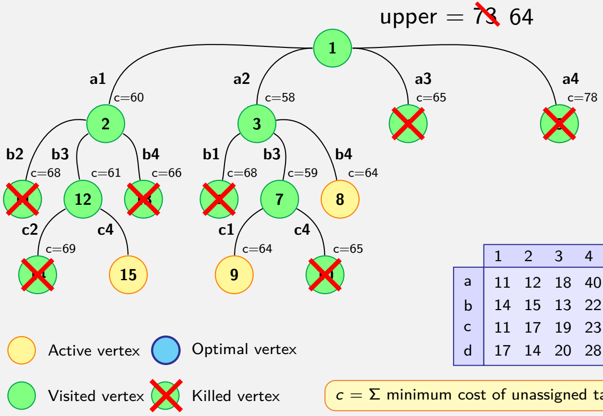
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

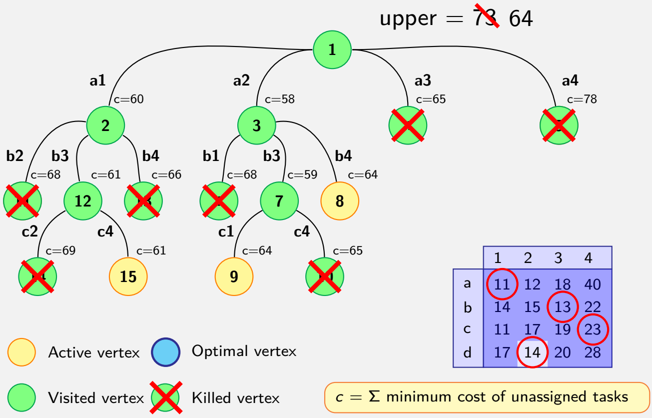
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

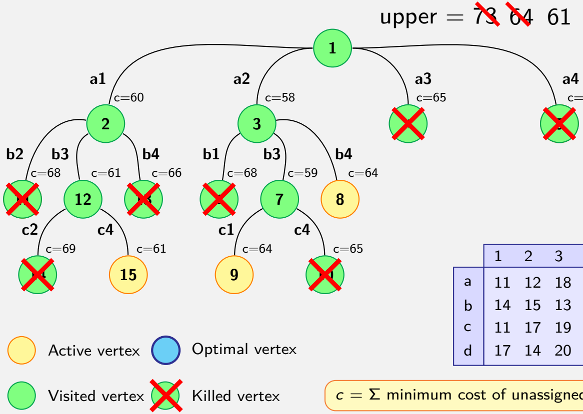
$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

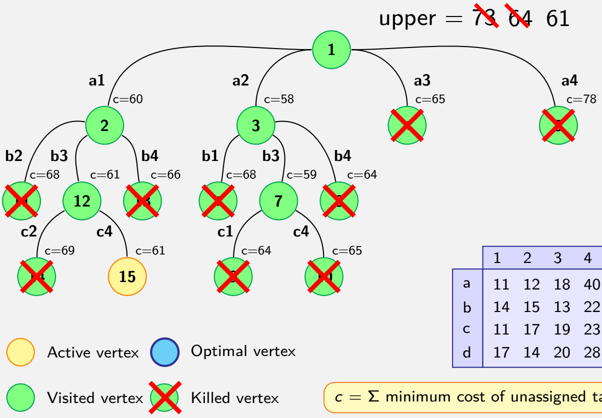
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

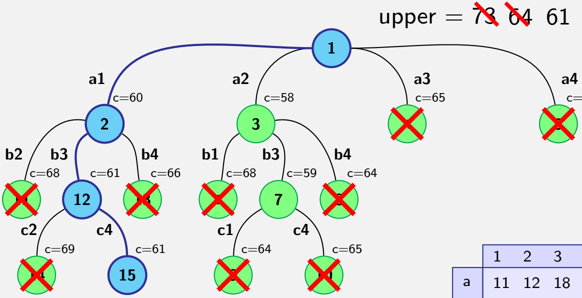
A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

A Second Example: Task Assignement



	1	2	3	4
a	11	12	18	40
b	14	15	13	22
c	11	17	19	23
d	17	14	20	28

$c = \Sigma$ minimum cost of unassigned tasks

1. A First Example: 0-1 Knapsack
2. A Second Example: Task Assignment
3. A Third Example: Travelling Salesman Problem (TSP)

Well, I got on the road, and I went north to Providence. (...) And then I went to Waterbury. Waterbury is a fine city. Big clock city, the famous Waterbury clock. Sold a nice bill there. And then Boston—Boston is the cradle of the Revolution. A fine city. And a couple of other towns in Mass., and on to Portland and Bangor and straight home!

Arthur Miller, *Death of a Salesman*

A Third Example: Travelling Salesman Problem (TSP)

- In this problem, we have a set S of connected cities. The edges of the graph have costs.

A Third Example: Travelling Salesman Problem (TSP)

- In this problem, we have a set S of connected cities. The edges of the graph have costs.
- A salesman wants to visit all cities (each city exactly once) with minimum cost.

A Third Example: Travelling Salesman Problem (TSP)

- In this problem, we have a set S of connected cities. The edges of the graph have costs.
- A salesman wants to visit all cities (each city exactly once) with minimum cost.
- The strategy is the following one: when we have a set V of visited cities with a cost c_1, there remains a set $R = S \backslash V$ of remaining cities.

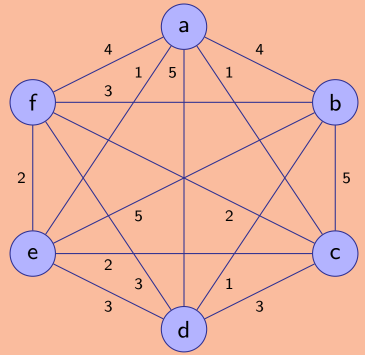
A Third Example: Travelling Salesman Problem (TSP)

- In this problem, we have a set S of connected cities. The edges of the graph have costs.
- A salesman wants to visit all cities (each city exactly once) with minimum cost.
- The strategy is the following one: when we have a set V of visited cities with a cost c_1, there remains a set $R = S \backslash V$ of remaining cities.
- The total cost cannot be better than the sum of c_1 plus the minimum cost c_2 to go from the last city of V to some city in R plus the minimum cost c_3 from some city in R to the first city in V plus the cost c_4 of the MST of R.

A Third Example: Travelling Salesman Problem (TSP)

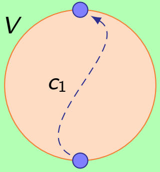
- In this problem, we have a set S of connected cities. The edges of the graph have costs.
- A salesman wants to visit all cities (each city exactly once) with minimum cost.
- The strategy is the following one: when we have a set V of visited cities with a cost c_1, there remains a set $R = S \backslash V$ of remaining cities.
- The total cost cannot be better than the sum of c_1 plus the minimum cost c_2 to go from the last city of V to some city in R plus the minimum cost c_3 from some city in R to the first city in V plus the cost c_4 of the MST of R.
- Can you see why?

The Map for the TSP Example



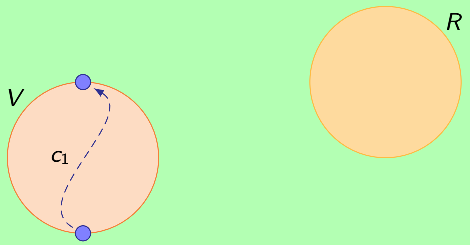
The Strategy for TSP

The Strategy for TSP



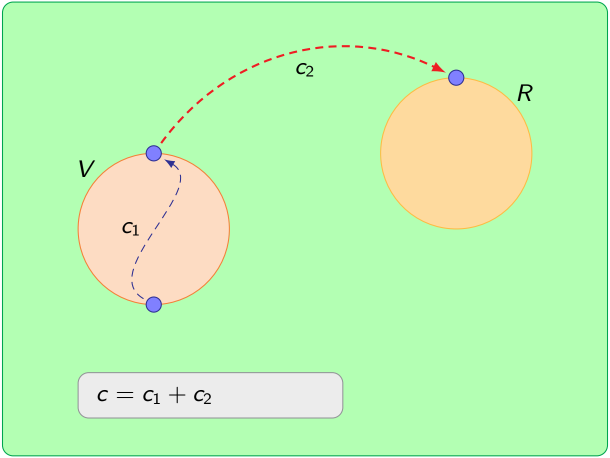
$c = c_1$

The Strategy for TSP

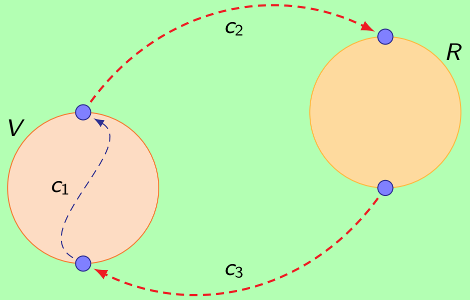


$c = c_1$

The Strategy for TSP

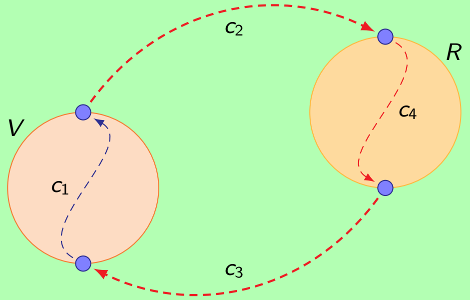


The Strategy for TSP



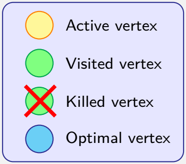
$$c = c_1 + c_2 + c_3$$

The Strategy for TSP

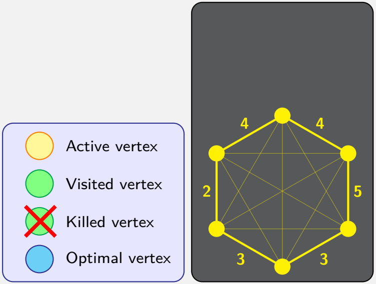


$c = c_1 + c_2 + c_3 + c_4$

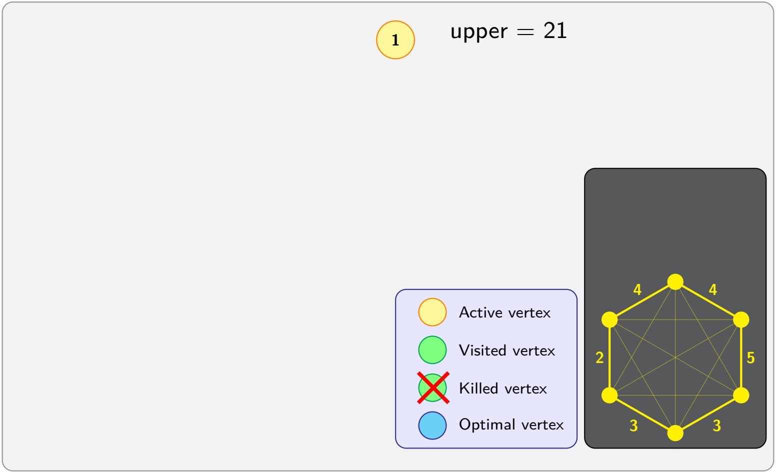
A Third Example: Travelling Salesman Problem (TSP)



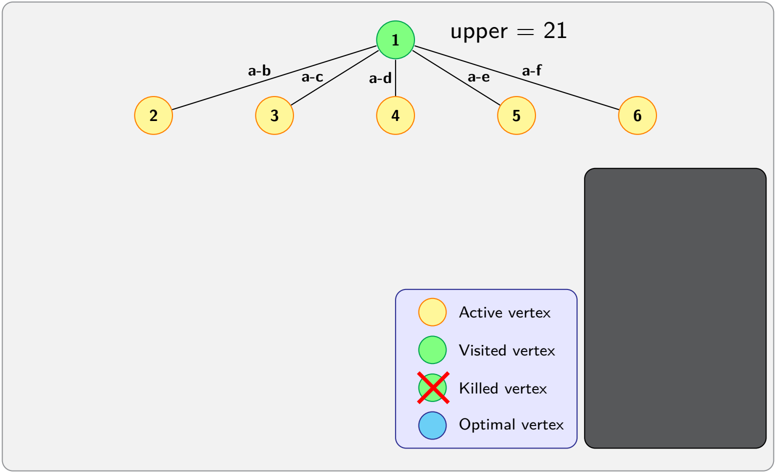
A Third Example: Travelling Salesman Problem (TSP)



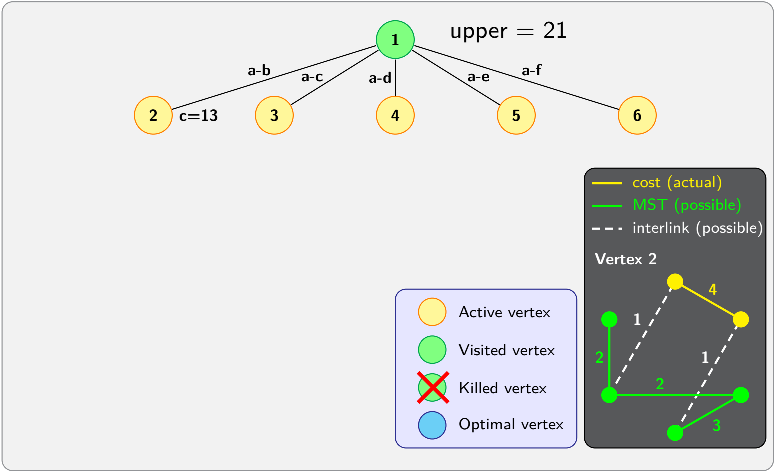
A Third Example: Travelling Salesman Problem (TSP)



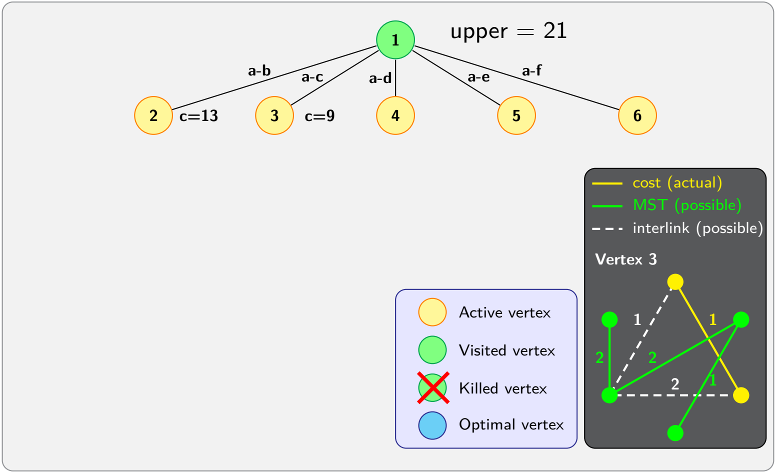
A Third Example: Travelling Salesman Problem (TSP)



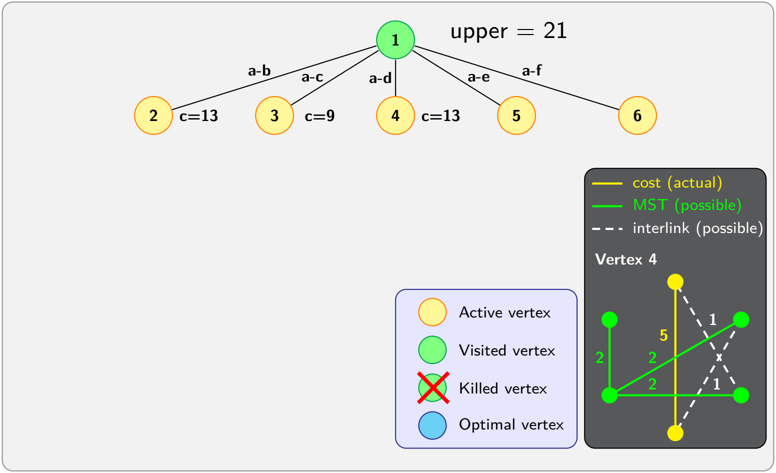
A Third Example: Travelling Salesman Problem (TSP)



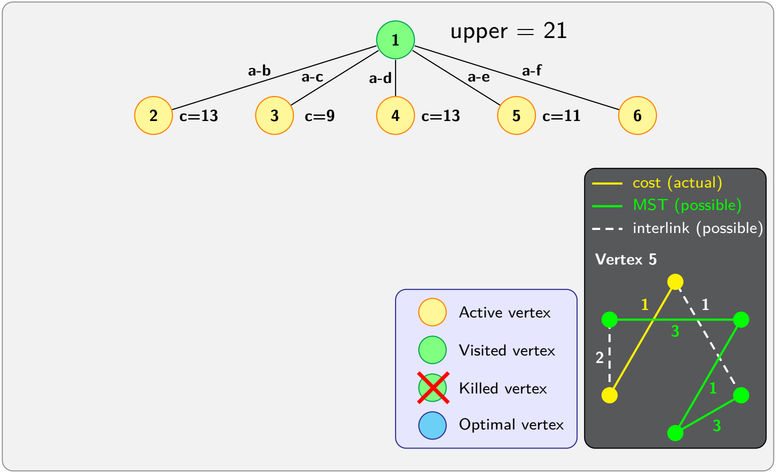
A Third Example: Travelling Salesman Problem (TSP)



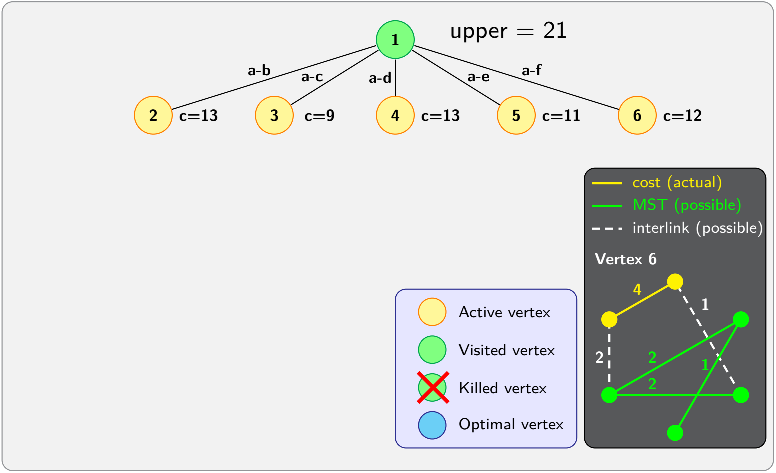
A Third Example: Travelling Salesman Problem (TSP)



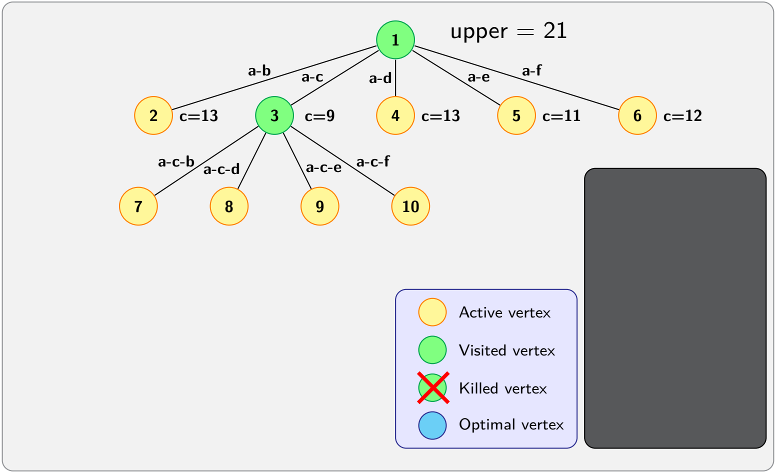
A Third Example: Travelling Salesman Problem (TSP)



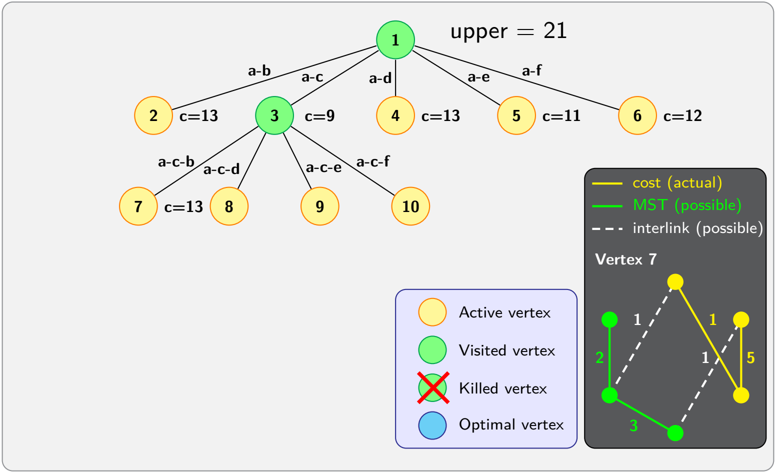
A Third Example: Travelling Salesman Problem (TSP)



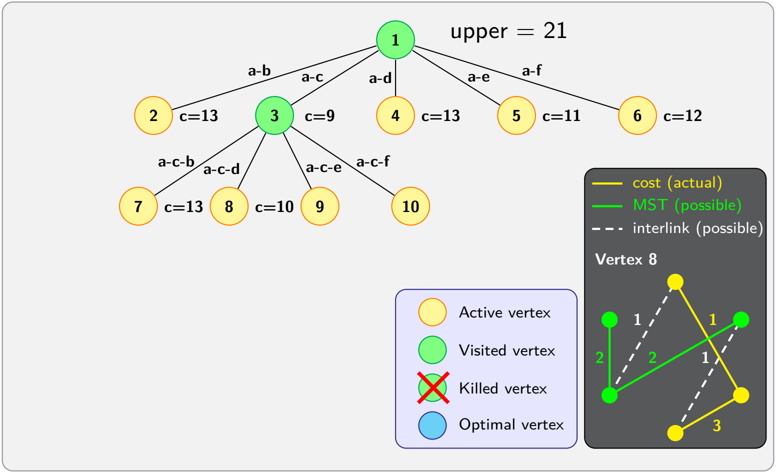
A Third Example: Travelling Salesman Problem (TSP)



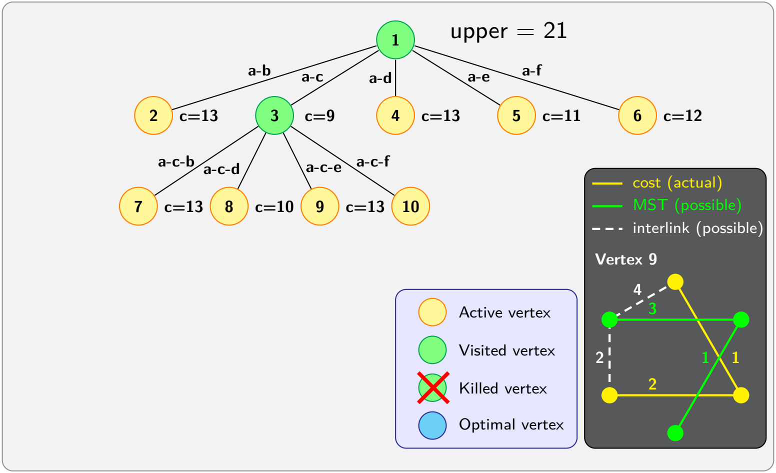
A Third Example: Travelling Salesman Problem (TSP)



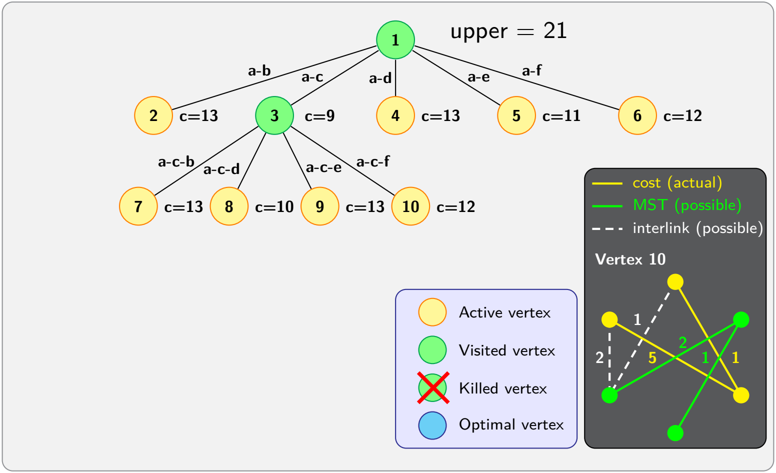
A Third Example: Travelling Salesman Problem (TSP)



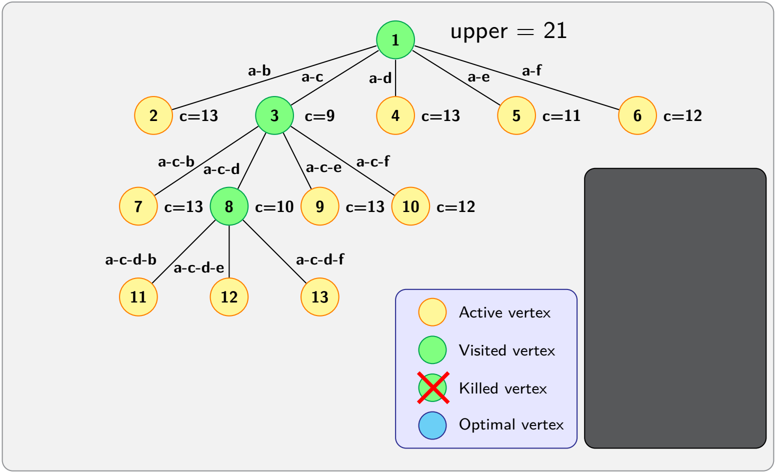
A Third Example: Travelling Salesman Problem (TSP)



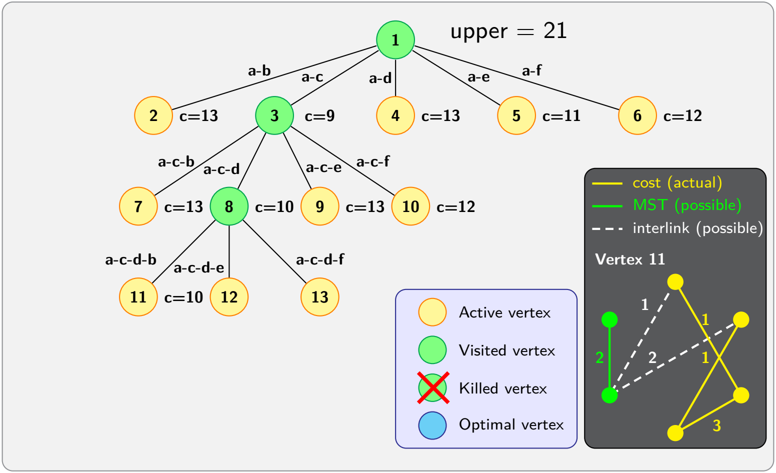
A Third Example: Travelling Salesman Problem (TSP)



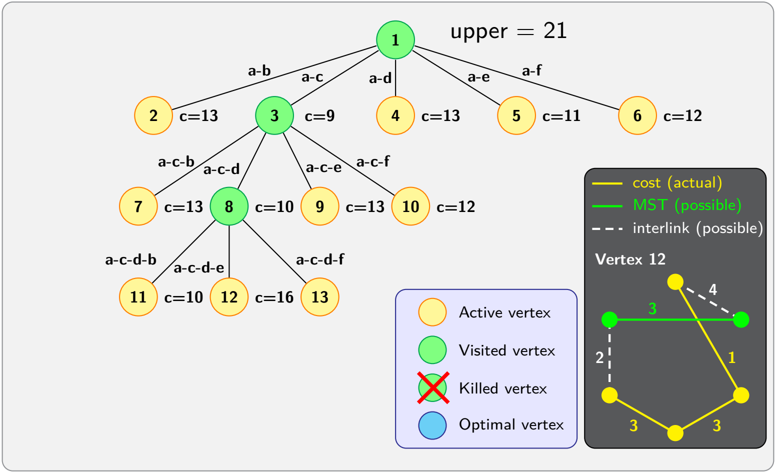
A Third Example: Travelling Salesman Problem (TSP)



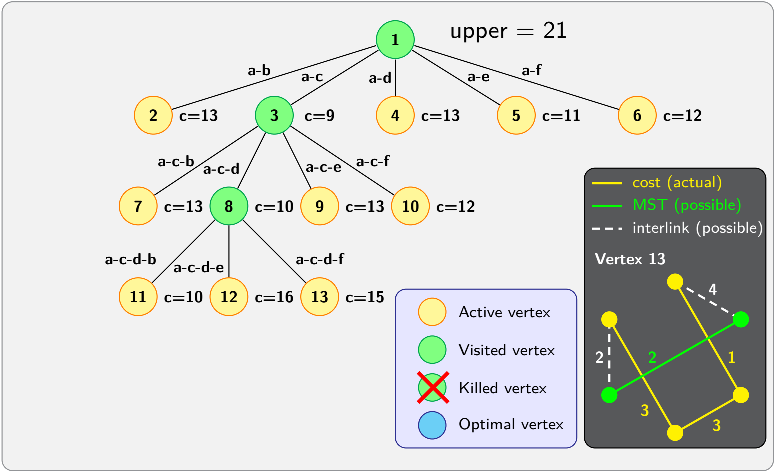
A Third Example: Travelling Salesman Problem (TSP)



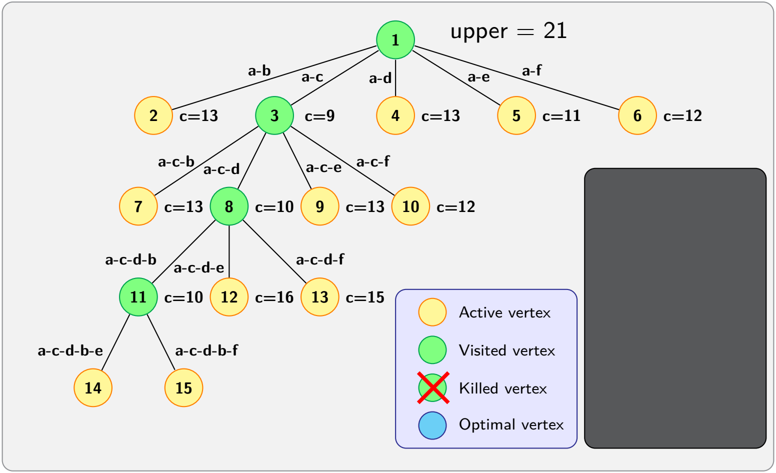
A Third Example: Travelling Salesman Problem (TSP)



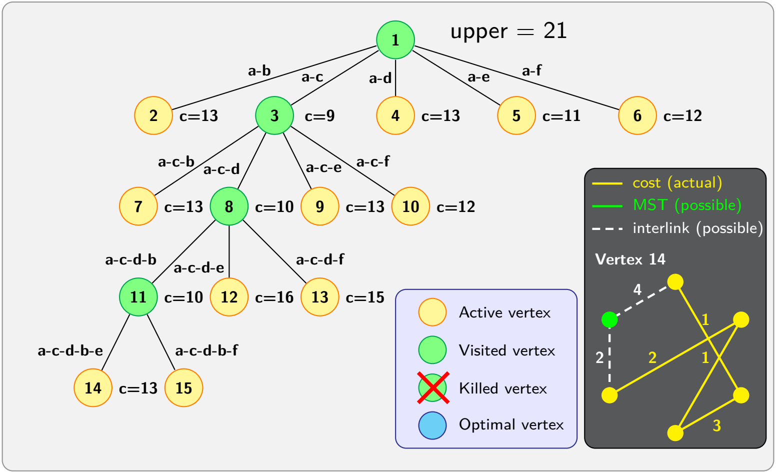
A Third Example: Travelling Salesman Problem (TSP)



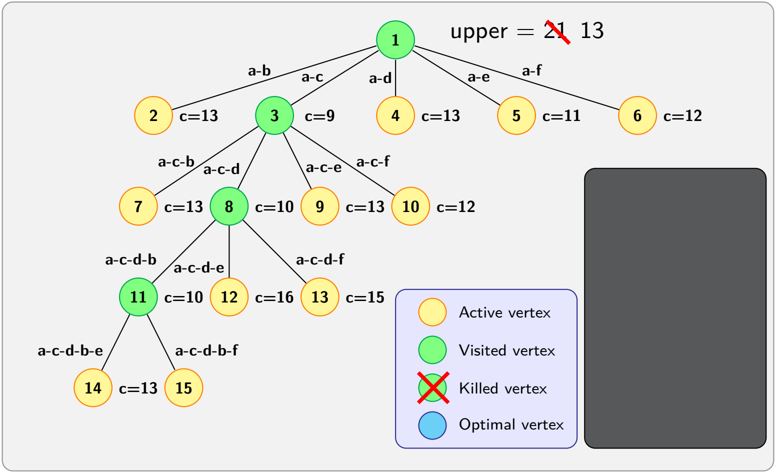
A Third Example: Travelling Salesman Problem (TSP)



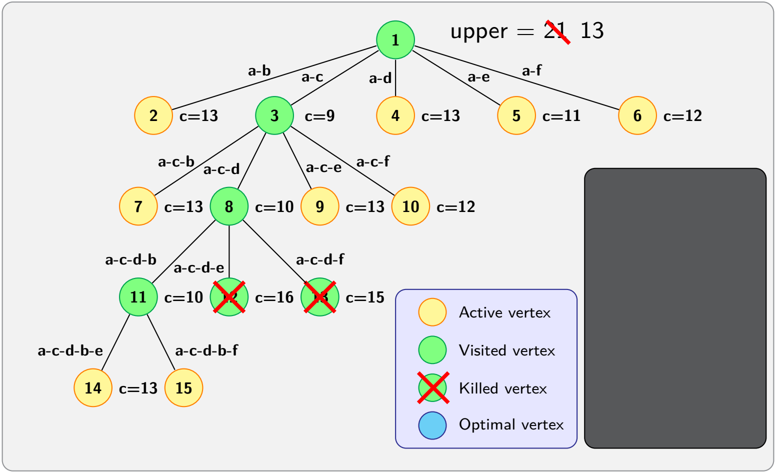
A Third Example: Travelling Salesman Problem (TSP)



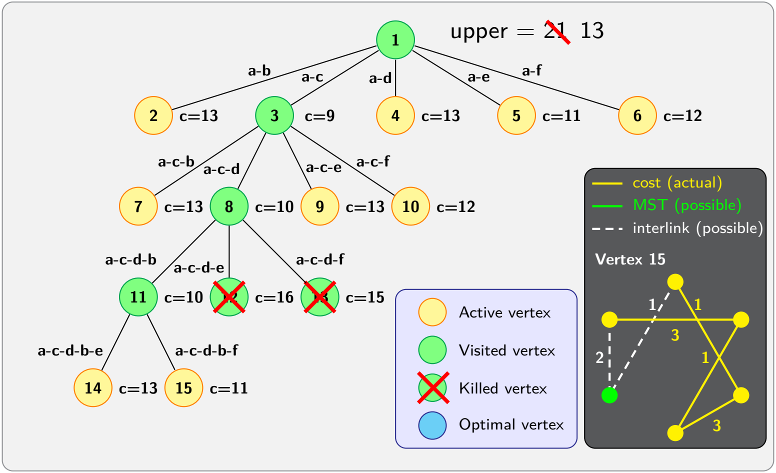
A Third Example: Travelling Salesman Problem (TSP)



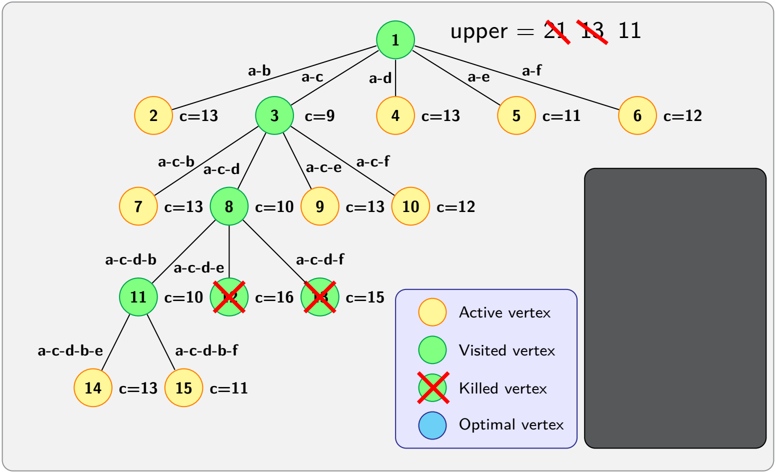
A Third Example: Travelling Salesman Problem (TSP)



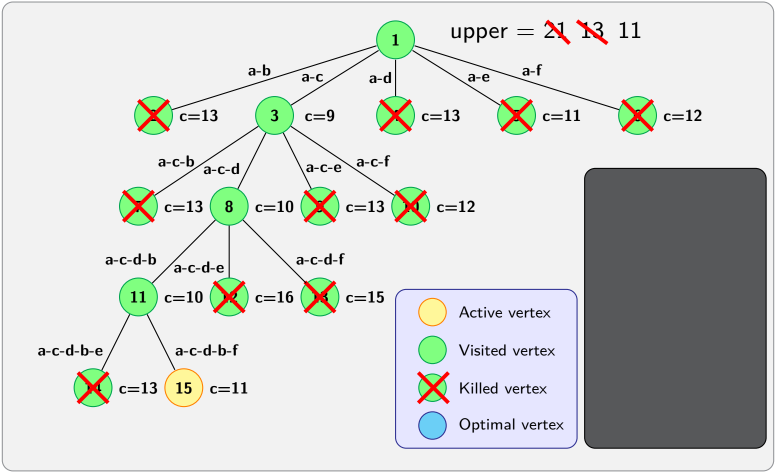
A Third Example: Travelling Salesman Problem (TSP)



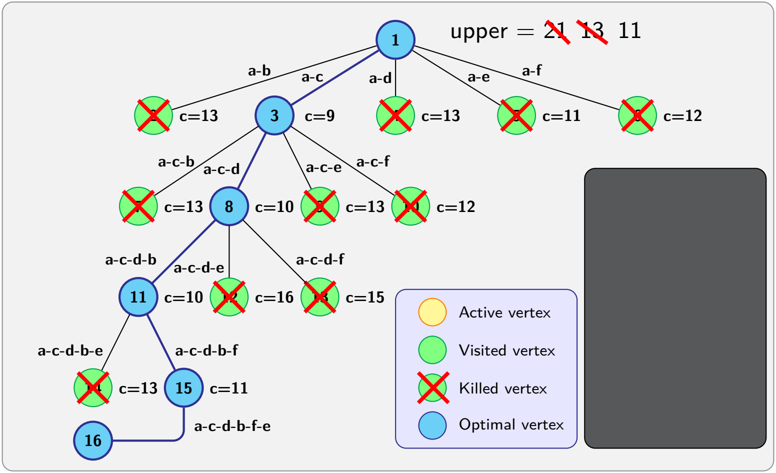
A Third Example: Travelling Salesman Problem (TSP)



A Third Example: Travelling Salesman Problem (TSP)



A Third Example: Travelling Salesman Problem (TSP)



A Third Example: Travelling Salesman Problem (TSP)