Branch & Bound

Branch and Bound

- Used to find optimal solution to many optimization problems, especially in discrete and combinatorial optimization.
- Systematic enumeration of all candidate solutions.
- Discard large subsets of fruitless candidates by using upper and lower estimated bounds of quantity being optimized.

Terminology

- **Live node** is a node that has been generated but whose children have not yet been generated.
- **E-node** is a live node whose children are currently being explored. In other words, an E-node is a node currently being expanded.
- **Dead node** is a generated node that is not to be expanded or explored any further. All children of a dead node have already been expanded.
- **Branch-and-bound** refers to all state space search methods in which all children of an E-node are generated before any other live node can become the E-node.

General method

- Both BFS and DFS generalize to branch-and-bound strategies
  - BFS is an FIFO search in terms of live nodes
    - List of live nodes is a queue
  - DFS is an LIFO search in terms of live nodes
    - List of live nodes is a stack

- We will use bounding functions to avoid generating subtrees that do not contain an answer node.

Backtracking vs Branch & Bound

Backtracking	Branch & Bound
It is used to find all possible solutions available to a problem.	It is used to solve optimization problem.
It traverses the state space tree by **DFS**(Depth First Search) manner.	It may traverse the tree in any manner, **DFS or BFS**.
It realizes that it has made a bad choice & undoes the last choice by backing up.	It realizes that it already has a better optimal solution that the pre-solution leads to so it abandons that pre-solution.
It searches the state space tree until it has found a solution.	It completely searches the state space tree to get optimal solution.
It involves **feasibility function**.	It involves a **bounding function**.

Branch & Bound

0/1 knapsack Problem

0/1 Knapsack Problem

- A thief robbing a store finds n items. The *i* th item is worth *i* dollars and weighs w_i pounds, where *i* and w_i are integers.
- The thief wants to take as valuable a load as possible, but he can carry at most W pounds in his knapsack, for some integer W.
- For each item, the thief must either take it or leave it behind. That is why it is called 0/1 knapsack.
- Which items should he take?
- In the ***fractional knapsack problem***, the setup is the same, but the thief can take fractions of items.

0/1 knapsack Example

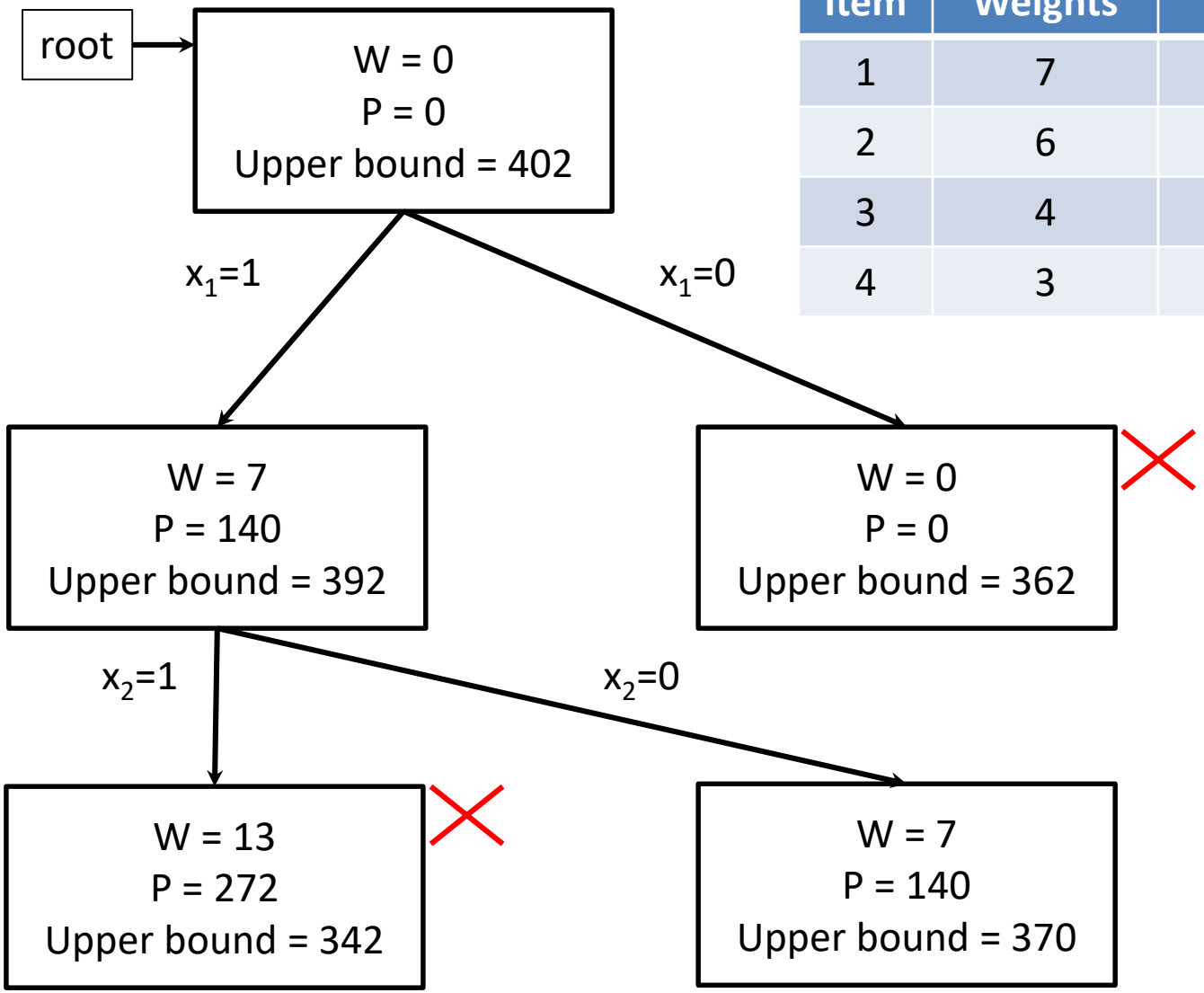
- The items' weights and values are:

Item	Weights	Value
1	7	140
2	6	132
3	4	140
4	3	90

- The knapsack capacity is 15.

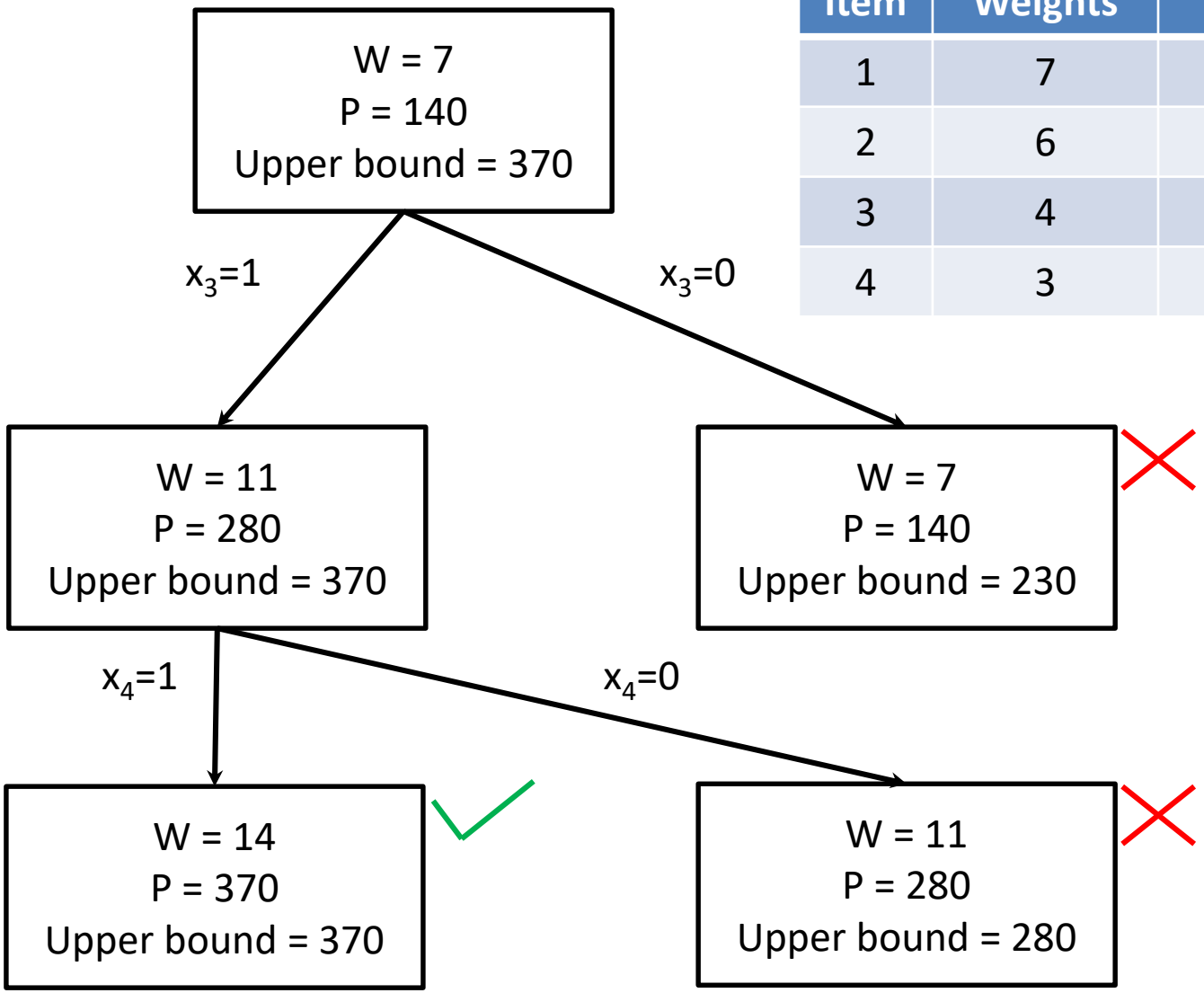
0/1 knapsack Example

Item	Weights	Value	Per Unit
1	7	140	20
2	6	132	22
3	4	140	35
4	3	90	30



0/1 knapsack Example

Item	Weights	Value	Per Unit
1	7	140	20
2	6	132	22
3	4	140	35
4	3	90	30



Optimal Solution:

Weight = 14

Profit = 370

Branch & Bound

15 Puzzle Problem

15 Puzzle Problem

- 15 numbered tiles on a square frame with a capacity for 16 tiles.
- Given an initial state, transform it to the goal state through a series of legal moves.

1	3	4	15
2		5	12
7	6	11	14
8	9	10	13

Initial State

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Goal State

- A state is reachable from the initial state iff there is a sequence of legal moves from initial state to this state.

15 Puzzle Problem

- Search the state space for the goal state and use the path from initial state to goal state as the answer.
- Number of possible arrangements for tiles: $16! \approx 20.9 \text{ X } 10^{12}$

- Depth first search takes us away from the goal rather than closer
  - The search of state space tree is blind; taking leftmost path from the root regardless of initial state
  - An answer node may never be found
- Breadth-first search will always find a goal state nearest to the root.

15 Puzzle Problem

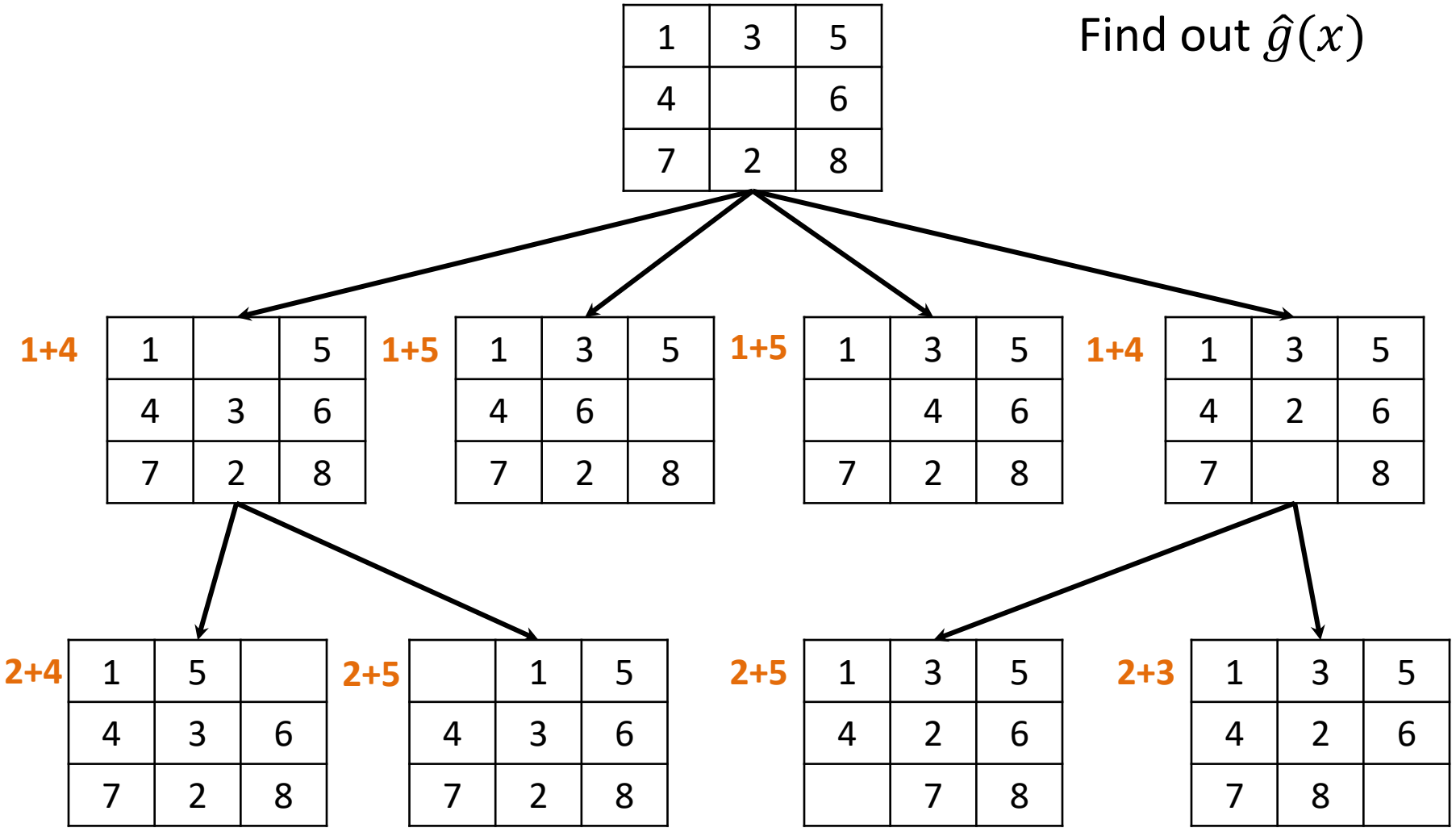
Intelligent Solution:

- Associate a cost c(x) with each node x of state space tree
  - c(x) is the length of path from the root to a nearest goal node (if any) in the subtree with root x

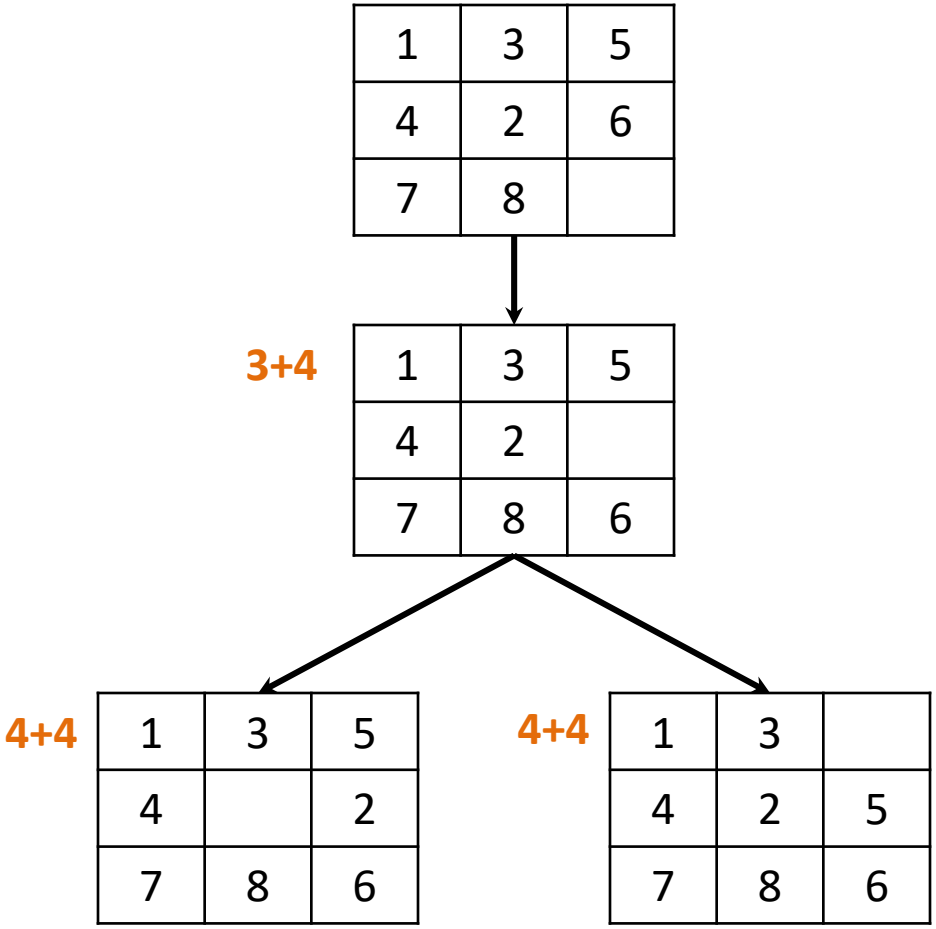
- $\hat{c}(x)$ = *f(x)* + $\hat{g}(x)$ where *f(x)* is the length of the path from root to x and $\hat{g}(x)$ is an estimate of the length of a shortest path from x to a goal node in the subtree with root x.

- One possible choice for $\hat{g}(x)$ is the number of nonblank tiles not in their goal position.

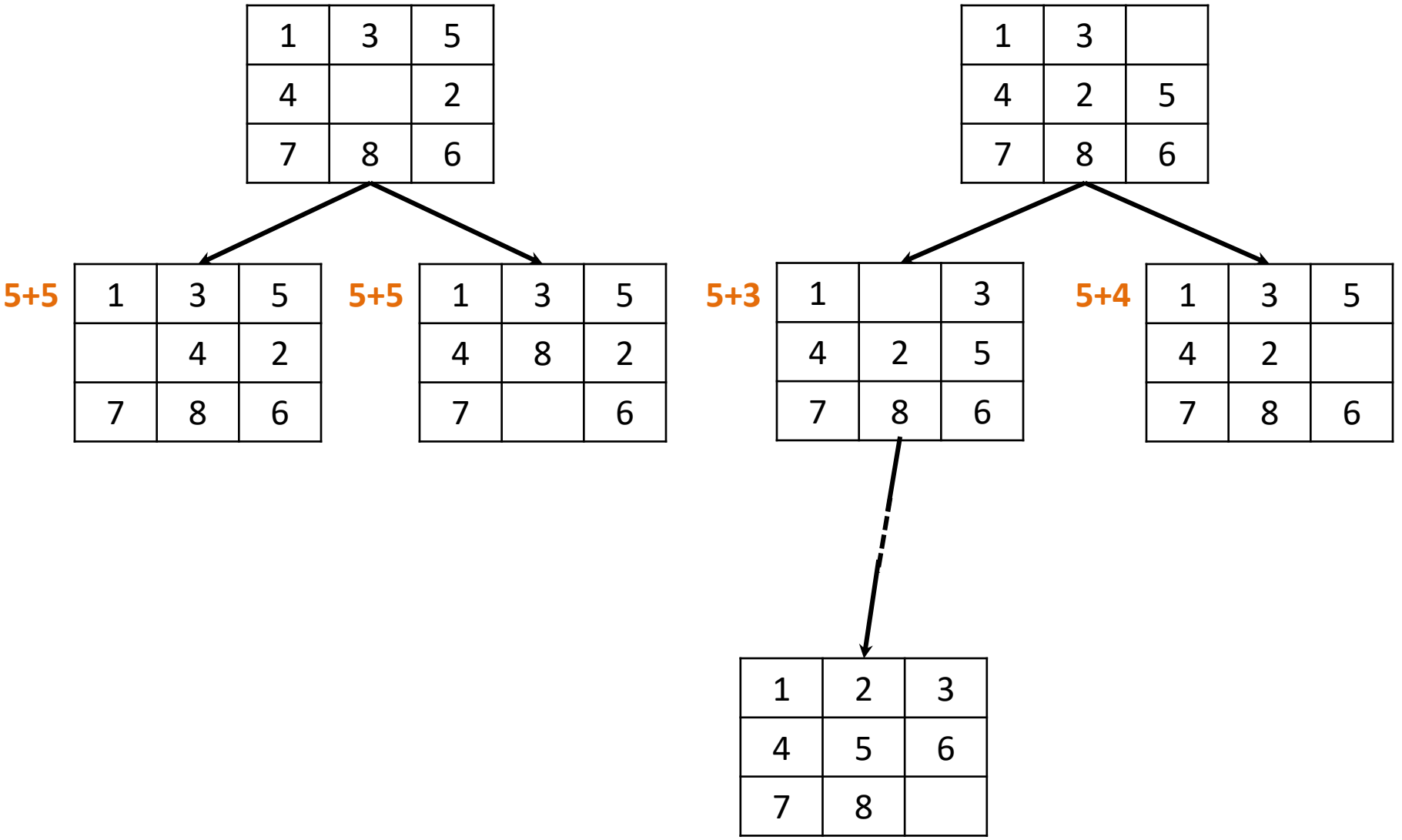
8 Puzzle Problem Example



8 Puzzle Problem Example



8 Puzzle Problem Example



Branch & Bound

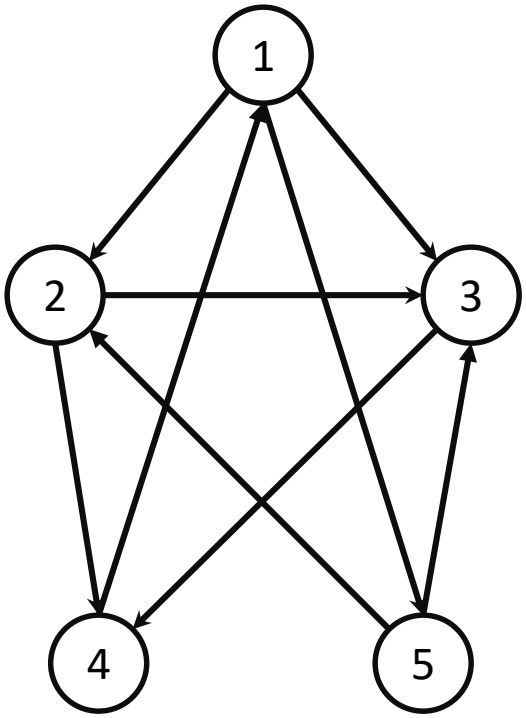
Traveling Salesman Problem

Traveling Salesman Problem

- Let us look at a situation that there are 5 cities, Which are represented as nodes
- There is a Person at node: 1
- This person has to reach each nodes one and only once and come back to original (starting)position.
- This process has to occur with minimum cost or minimum distance travelled.
- Note that starting point can start with any node.
- For Example:

  1-5-2-3-4-1

  2-3-4-1-5-2

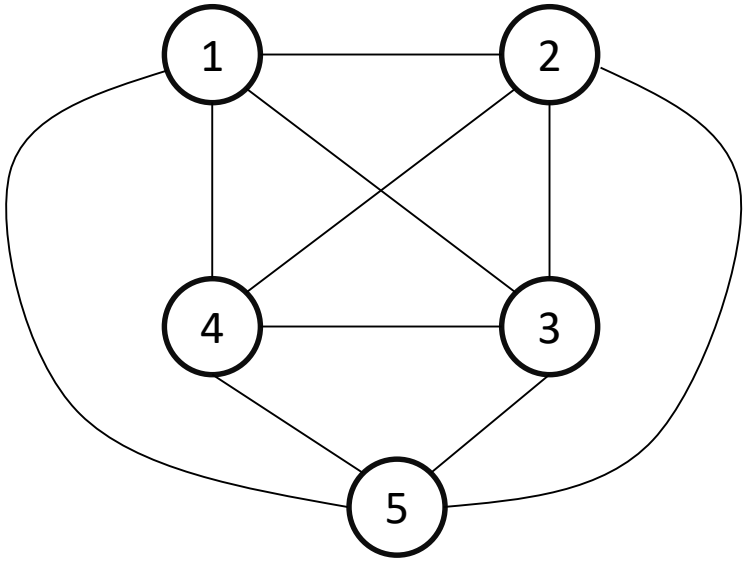


Traveling Salesman Problem

- If there are ‘n’ nodes there are (n-1)! Feasible solutions
- From these (n-1)! Feasible solutions we have to find **optimal solution**.
- This can be related to graph theory.
- Travelling Salesman Problem is nothing but finding out **LEAST COST HAMILTONIAN CIRCUIT**

Traveling Salesman Problem Example

- Let the adjacency matrix for the given graph be:



	1	2	3	4	5
1	-	20	30	10	11
2	15	-	16	4	2
3	3	5	-	2	4
4	19	6	18	-	3
5	16	4	7	16	-

Traveling Salesman Problem Example

- Find out Row Minima and then reduce the matrix
- Next find the Column Minima
- Total minimum distance will be summation of them.

	1	2	3	4	5
1	-	20	30	10	11
2	15	-	16	4	2
3	3	5	-	2	4
4	19	6	18	-	3
5	16	4	7	16	-

	1	2	3	4	5
1	-	10	20	0	1
2	13	-	14	2	0
3	1	3	-	0	2
4	16	3	15	-	0
5	12	0	3	12	-

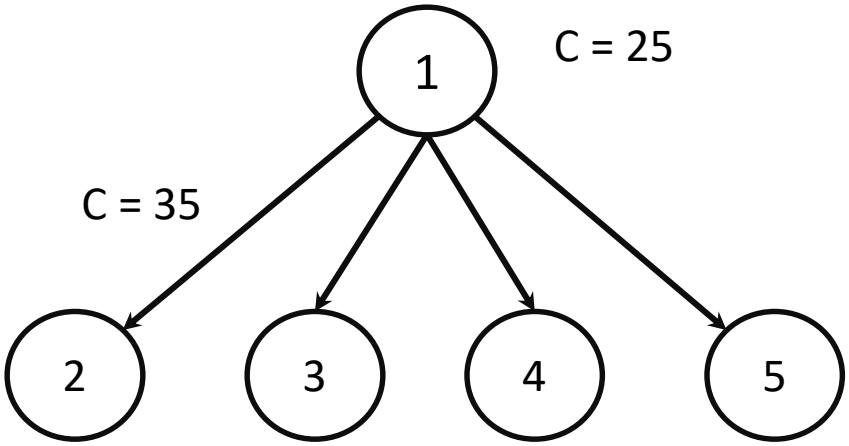
Traveling Salesman Problem Example

- Here the total minimum distance = (10+2+2+3+4)+(1+3)
- So the lower bound is 25.
- That is a person can surely travel the cities with the distance greater than or equal to 25.

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

Reduced matrix and reduced cost = 25

Traveling Salesman Problem Example



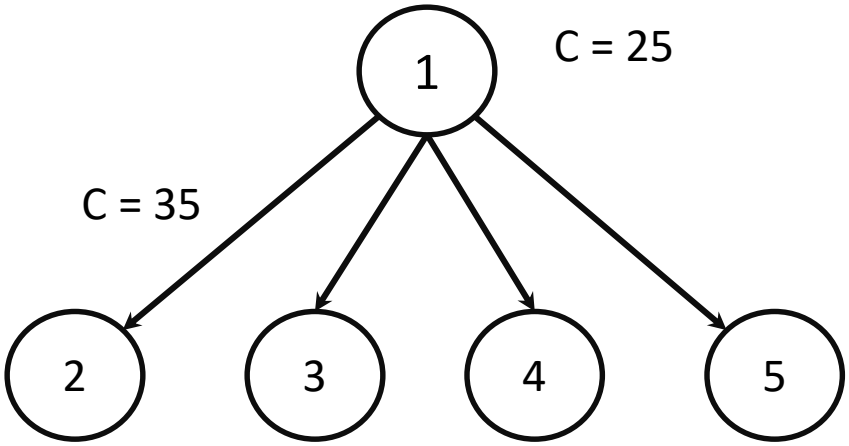
	1	2	3	4	5
1	-	-	-	-	-
2	-	-	11	2	0
3	0	-	-	0	2
4	15	-	12	-	0
5	11	-	0	12	-

$$C(1, 2) + \gamma + \hat{\gamma}$$

$$= 10 + 25 + 0 = 35$$

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

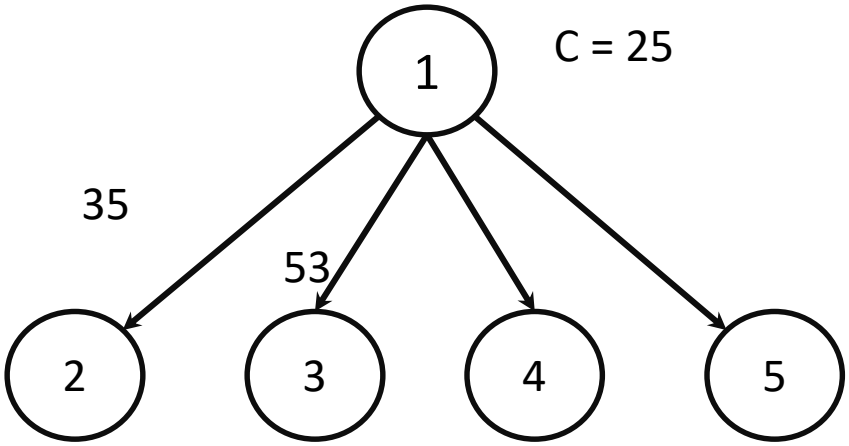
Traveling Salesman Problem Example



	1	2	3	4	5
1	-	-	-	-	-
2	12	-	-	2	0
3	-	3	-	0	2
4	15	3	-	-	0
5	11	0	-	12	-

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

Traveling Salesman Problem Example



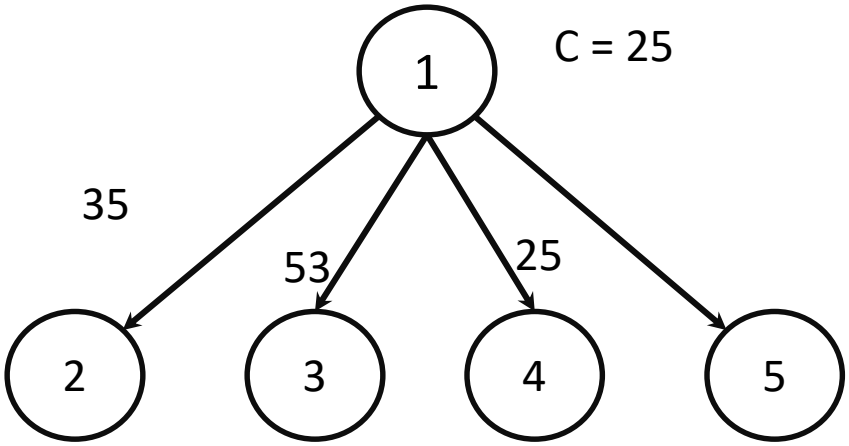
	1	2	3	4	5
1	-	-	-	-	-
2	1	-	-	2	0
3	-	3	-	0	2
4	4	3	-	-	0
5	0	0	-	12	-

$$C(1, 3) + \gamma + \hat{\gamma}$$

$$= 17 + 25 + 11 = 53$$

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

Traveling Salesman Problem Example



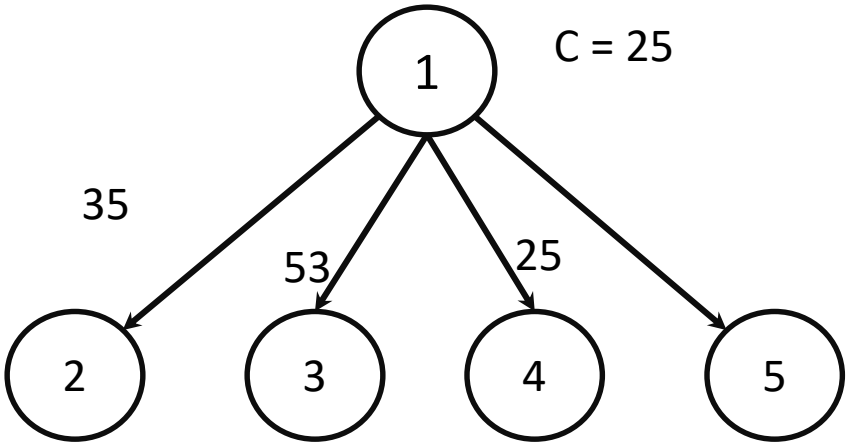
	1	2	3	4	5
1	-	-	-	-	-
2	12	-	11	-	0
3	0	3	-	-	2
4	15	3	12	-	0
5	11	0	0	-	-

$$C(1, 4) + \gamma + \hat{\gamma}$$

$$= 0 + 25 + 0 = 25$$

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

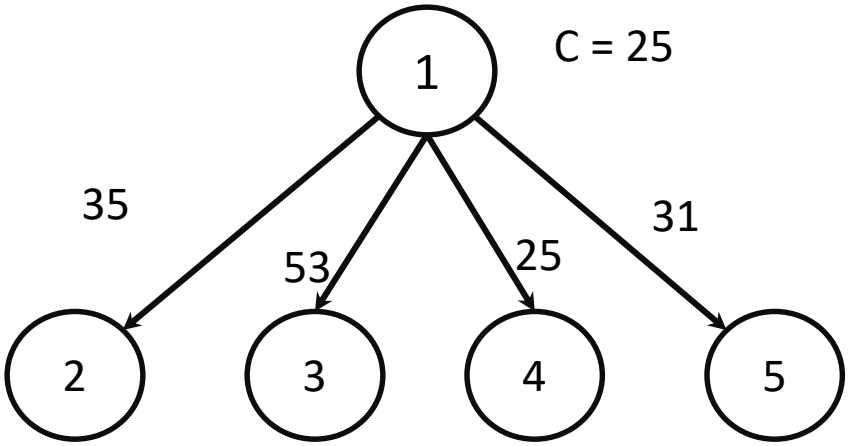
Traveling Salesman Problem Example



	1	2	3	4	5
1	-	-	-	-	-
2	12	-	11	2	-
3	0	3	-	0	-
4	15	3	12	-	-
5	-	0	0	12	-

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

Traveling Salesman Problem Example



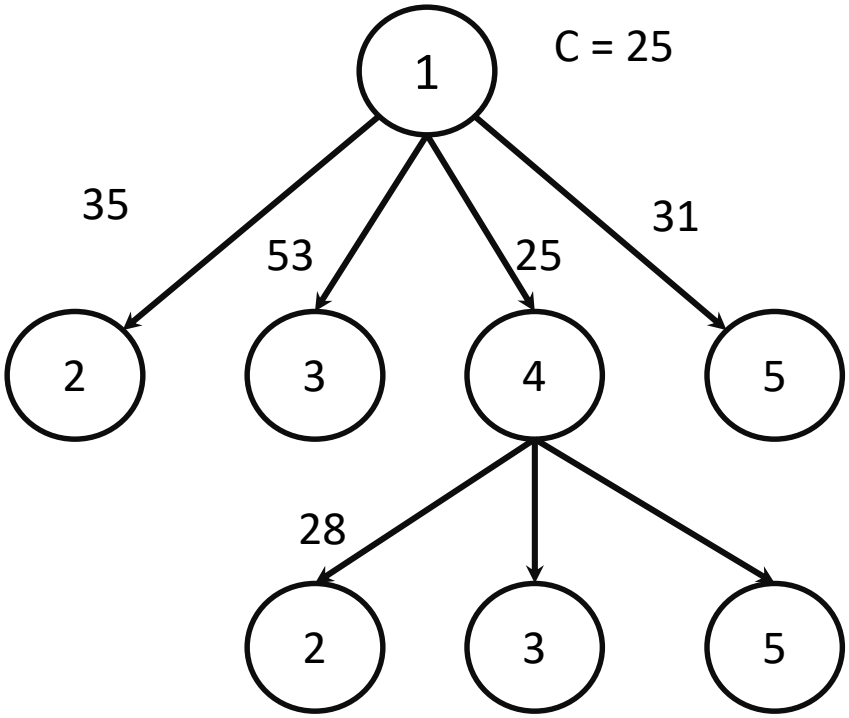
	1	2	3	4	5
1	-	-	-	-	-
2	10	-	9	0	-
3	0	3	-	0	-
4	12	0	9	-	-
5	-	0	0	12	-

$$C(1, 5) + \gamma + \hat{\gamma}$$

$$= 1 + 25 + 5 = 31$$

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

Traveling Salesman Problem Example



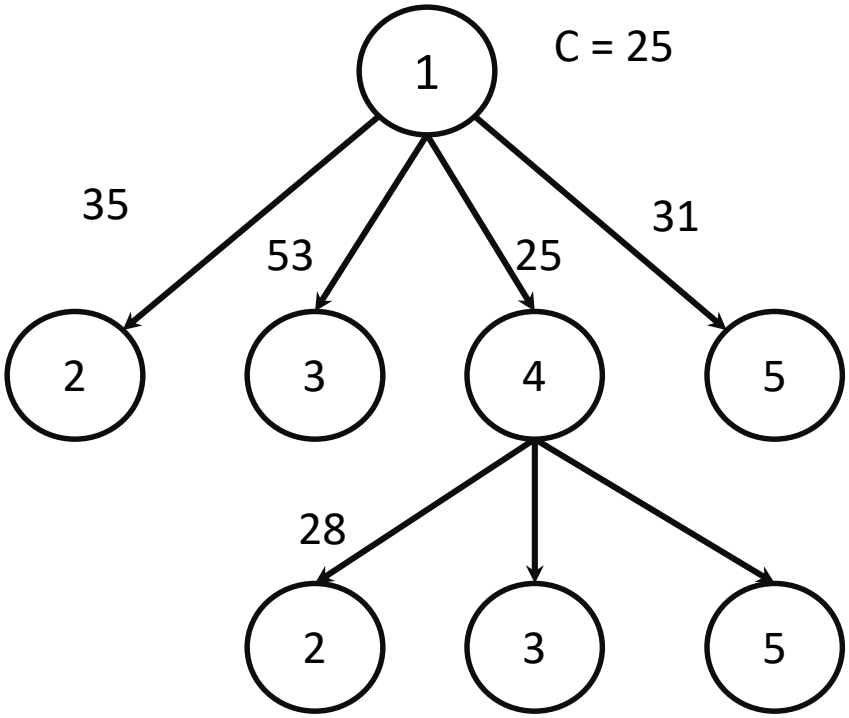
	1	2	3	4	5
1	-	-	-	-	-
2	-	-	11	-	0
3	0	-	-	-	2
4	-	-	-	-	-
5	11	-	0	-	-

$$C(4, 2) + \gamma + \hat{\gamma}$$

$$= 3 + 25 + 0 = 28$$

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

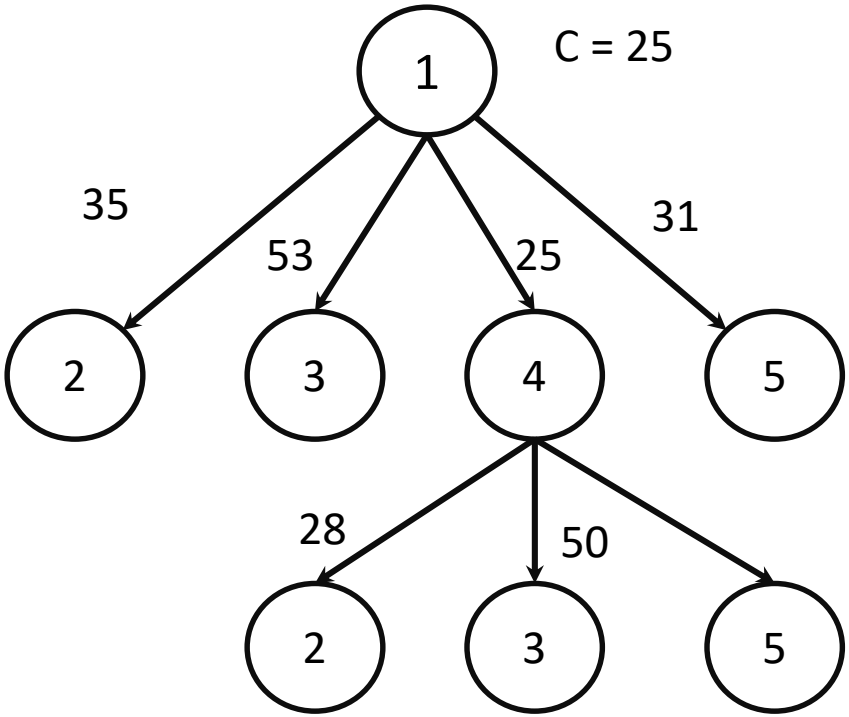
Traveling Salesman Problem Example



	1	2	3	4	5
1	-	-	-	-	-
2	12	-	-	-	0
3	-	3	-	-	2
4	-	-	-	-	-
5	11	0	-	-	-

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

Traveling Salesman Problem Example



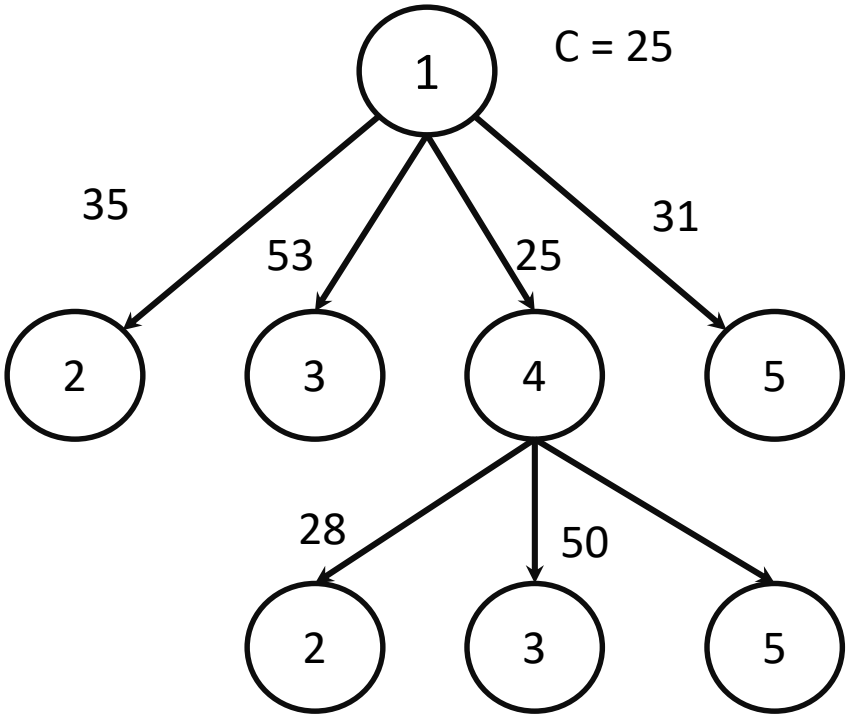
	1	2	3	4	5
1	-	-	-	-	-
2	1	-	-	-	0
3	-	1	-	-	0
4	-	-	-	-	-
5	0	0	-	-	-

$$C(4, 3) + \gamma + \hat{\gamma}$$

$$= 12 + 25 + 13 = 50$$

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

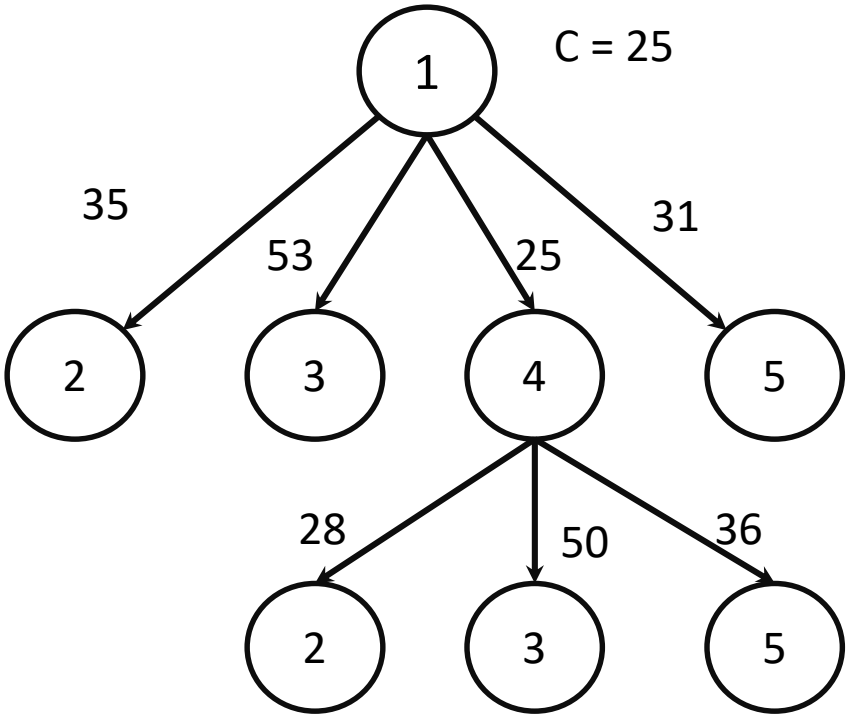
Traveling Salesman Problem Example



	1	2	3	4	5
1	-	-	-	-	-
2	12	-	11	-	-
3	0	3	-	-	-
4	-	-	-	-	-
5	-	0	0	-	-

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

Traveling Salesman Problem Example



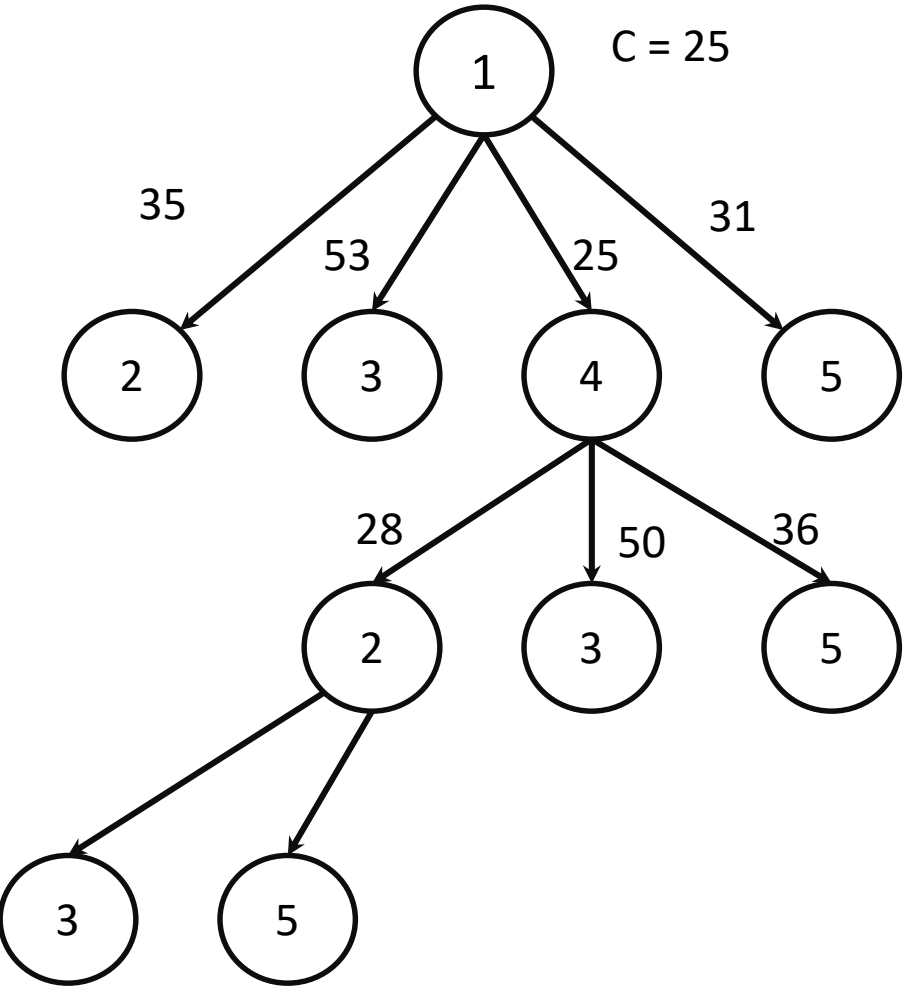
	1	2	3	4	5
1	-	-	-	-	-
2	1	-	0	-	-
3	0	3	-	-	-
4	-	-	-	-	-
5	-	0	0	-	-

$C(4, 5) + \gamma + \hat{\gamma}$

$= 0 + 25 + 11 = 36$

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

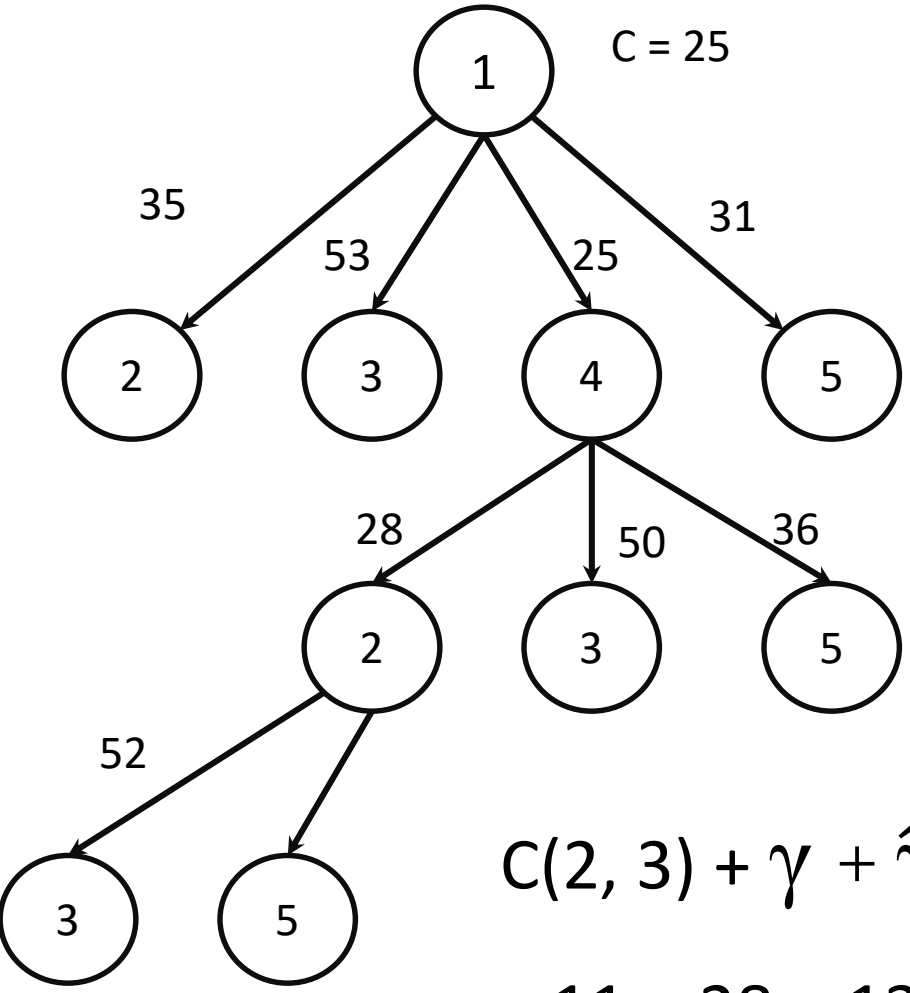
Traveling Salesman Problem Example



	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	2
4	-	-	-	-	-
5	11	-	-	-	-

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

Traveling Salesman Problem Example



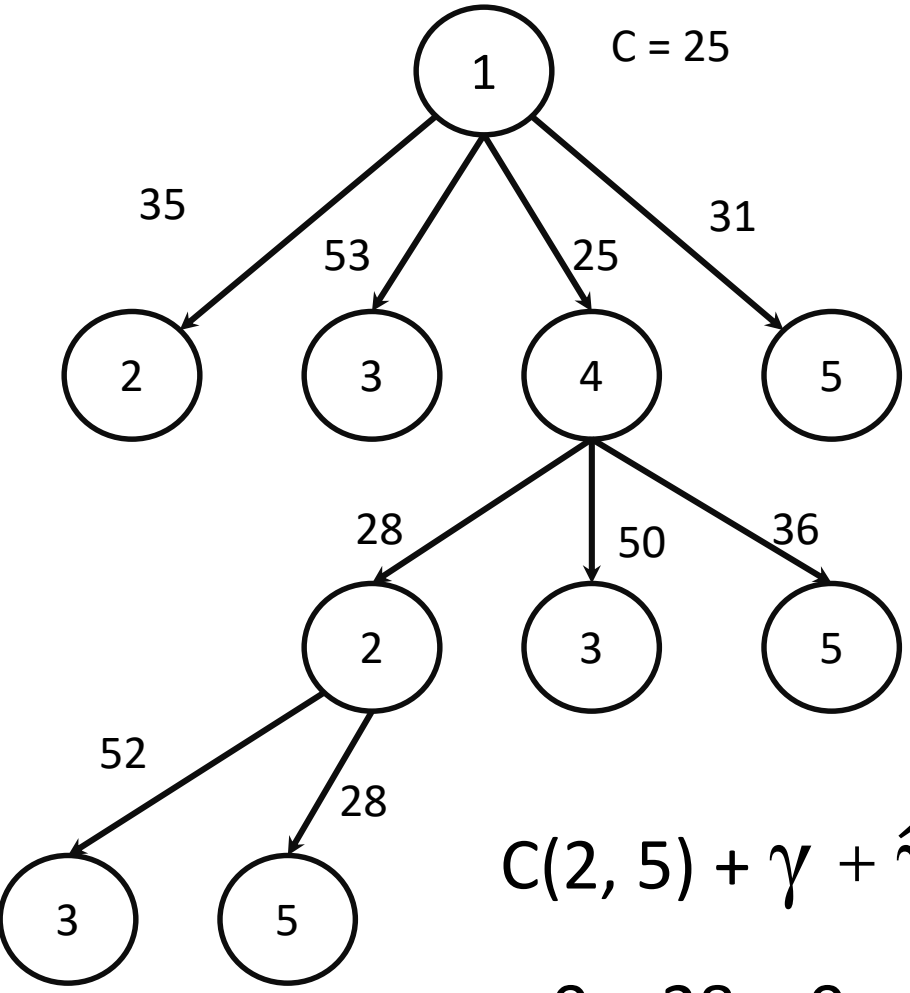
	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	-	-	-	-	0
4	-	-	-	-	-
5	0	-	-	-	-

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

$$C(2, 3) + \gamma + \hat{\gamma}$$

$$= 11 + 28 + 13 = 52$$

Traveling Salesman Problem Example



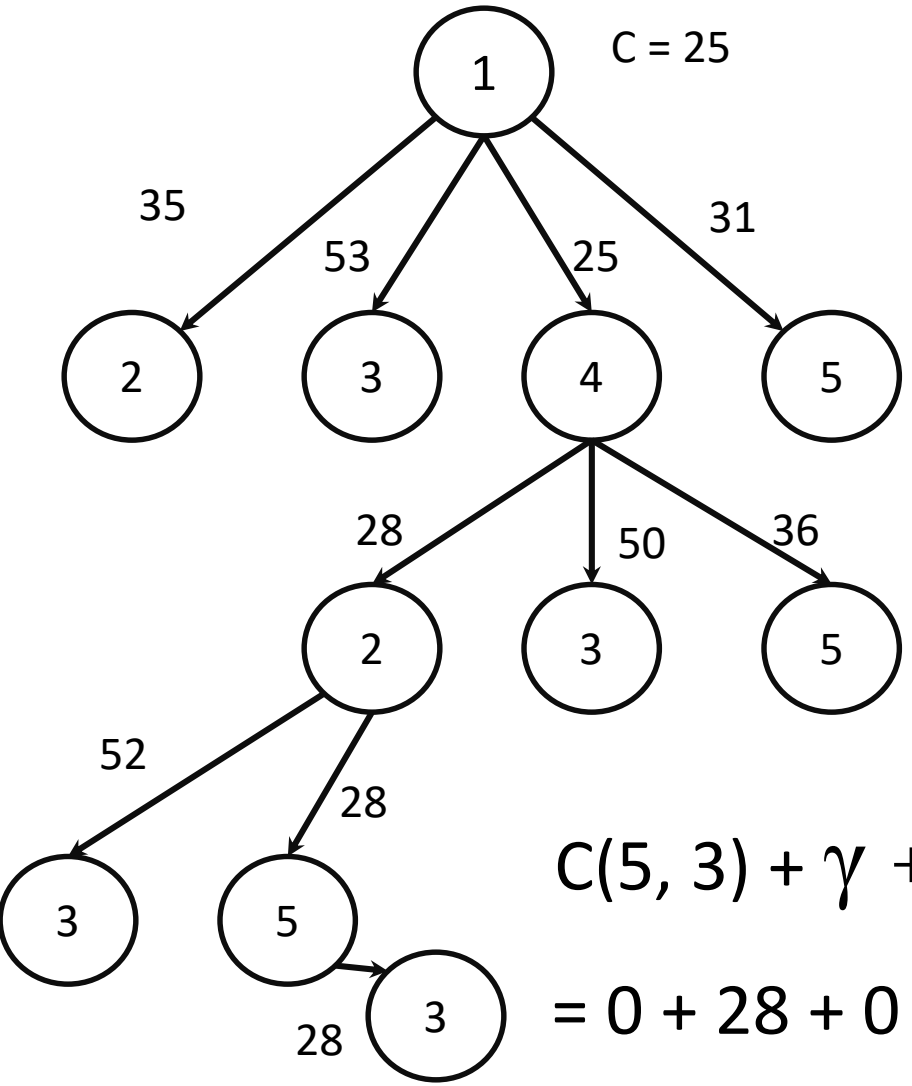
	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	0	-	-	-	-
4	-	-	-	-	-
5	-	-	0	-	-

$$C(2, 5) + \gamma + \hat{\gamma}$$

$$= 0 + 28 + 0 = 28$$

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

Traveling Salesman Problem Example



	1	2	3	4	5
1	-	-	-	-	-
2	-	-	-	-	-
3	0	-	-	-	-
4	-	-	-	-	-
5	-	-	0	-	-

	1	2	3	4	5
1	-	10	17	0	1
2	12	-	11	2	0
3	0	3	-	0	2
4	15	3	12	-	0
5	11	0	0	12	-

$$C(5, 3) + \gamma + \hat{\gamma}$$

$$= 0 + 28 + 0 = 28$$