HEURISTIC ALGORITHMS FOR THE TRAVELLING SALESMAN PROBLEM

Given a DIRECTED GRAPH $G = (V,A)$ with

- $V = \{1, \ldots, n\}$ vertex set
- $A = \{(i, j) : i \in V, j \in V\}$ arc set (complete digraph)
- c_{ij} = cost associated with arc $(i, j) \in A$ ($c_{ii} = \infty$, $i \in V$)
  (the costs can take any value)

- Find a HAMILTONIAN CIRCUIT (Tour) whose global cost is minimum (Asymmetric Travelling Salesman Problem: ATSP).

Maximization version

- ATSP is $\mathcal{NP}$-Hard in the strong sense.

- If G is complete the "*feasibility problem*" is polynomial ({(1, 2), (2, 3), …, (n-1, n), (n, 1)} is a feasible tour).

  * If G is sparse the "*feasibility problem*" is $\mathcal{NP}$-Hard

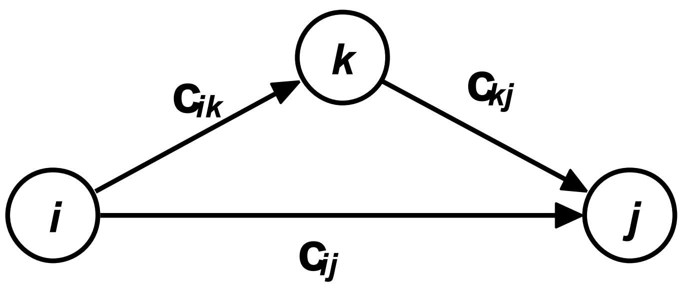
- If G is an undirected graph: Symmetric TSP (STSP)

  ($c_{ij} = c_{ji}$ for each $(i, j) \in A$)

- If G = (V, A) is a sparse graph: $c_{ij} = \infty$ for each $(i, j) \notin A$.

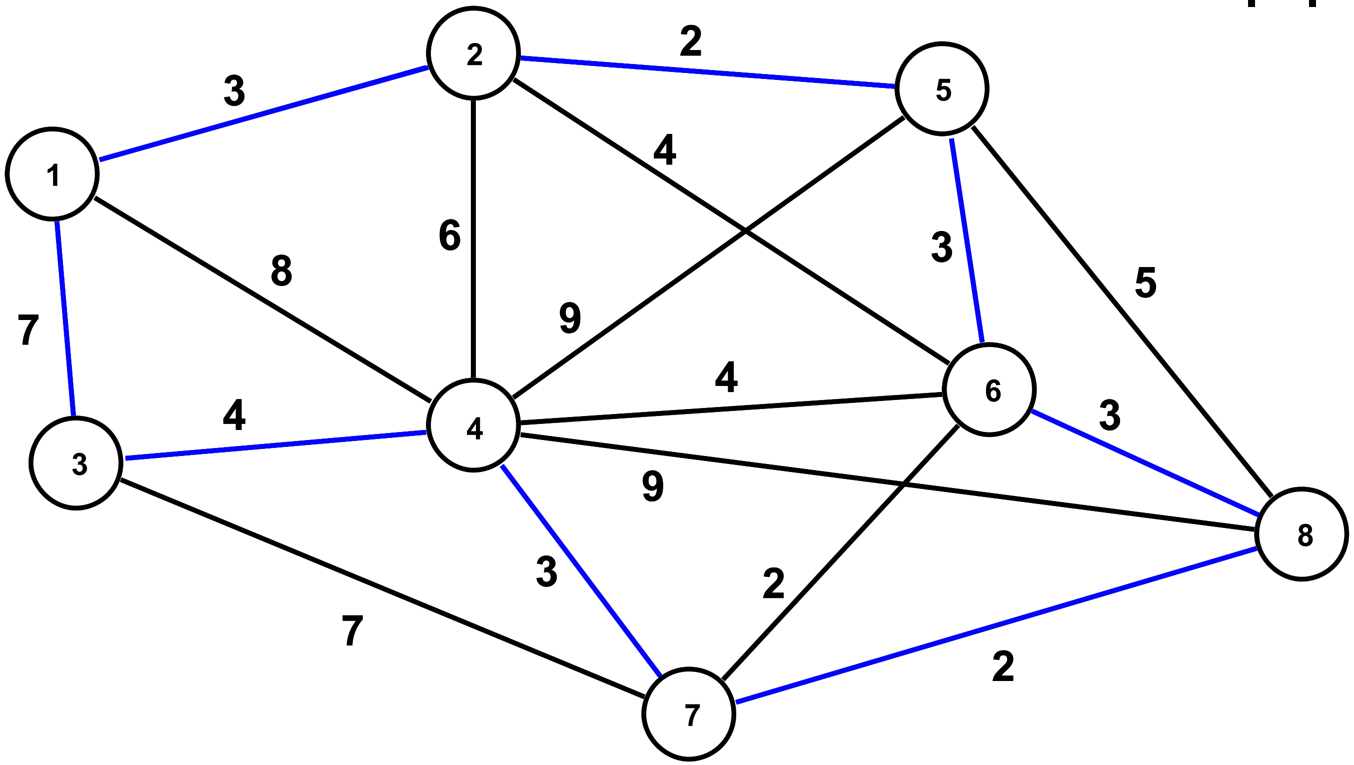
- If the "Triangle Inequality" holds:

  $c_{ij} \leq c_{ik} + c_{kj}$ for each $i, j, k \in V$.



Example 1 (STSP: undirected graph)

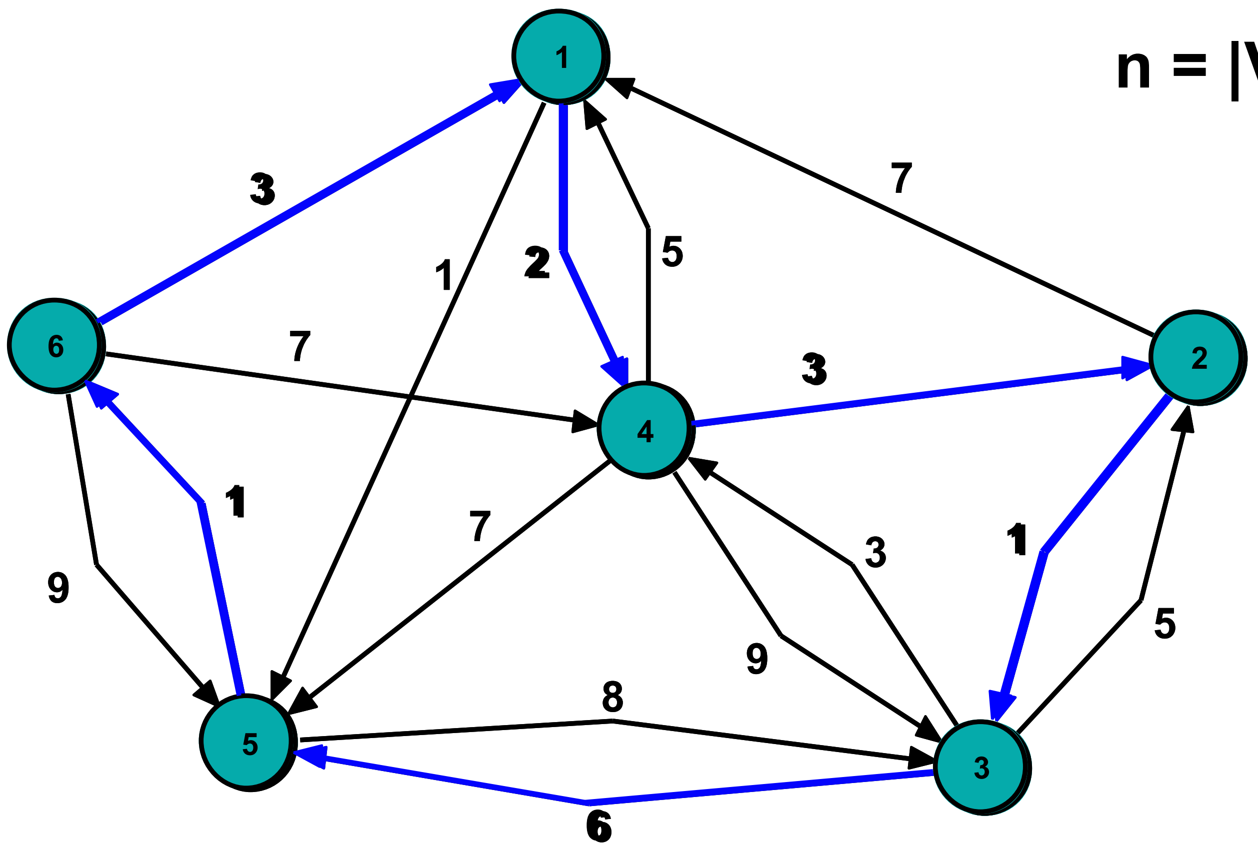
n = |V| = 8



Optimal solution

Optimal solution cost: Opt = 27

Example A (ATSP)



Optimal solution

Optimal solution Cost = 16

Applications

- Vehicle Routing (sequencing the customers in each route in an urban area calls for the optimal solution of the ATSP corresponding to the depot and the customers in the route).

- Scheduling (optimal sequencing of jobs on a machine when the set-up costs depend on the sequence in which the jobs are processed).

- Picking in an Inventory System (sequence of movements of a crane to pick-up a set of items stored on shelves).

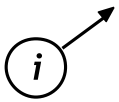
- …

INTEGER LINEAR PROGRAMMING FORMULATION

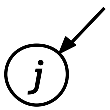
$$x_{ij} = \begin{cases} 1 & \text{if arc } (i, j) \text{ is in the optimal tour} \\ 0 & \text{otherwise} \end{cases} \qquad i \in V,\ j \in V$$

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

s.t.



$$\sum_{j \in V} x_{ij} = 1 \qquad i \in V$$



$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V$$

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subset V,\ |S| \geq 2$$

SUBTOUR ELIMINATION CONSTRAINTS
(impose the connectivity of the solution; $O(2^n)$)

$$x_{ij} \in \{0, 1\} \qquad i \in V,\ j \in V$$

ASSIGNMENT PROBLEM (AP) RELAXATION ($O(n^3)$ time)

$$\min \sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

s.t.

$$\sum_{j \in V} x_{ij} = 1 \qquad i \in V$$

$$\sum_{i \in V} x_{ij} = 1 \qquad j \in V$$

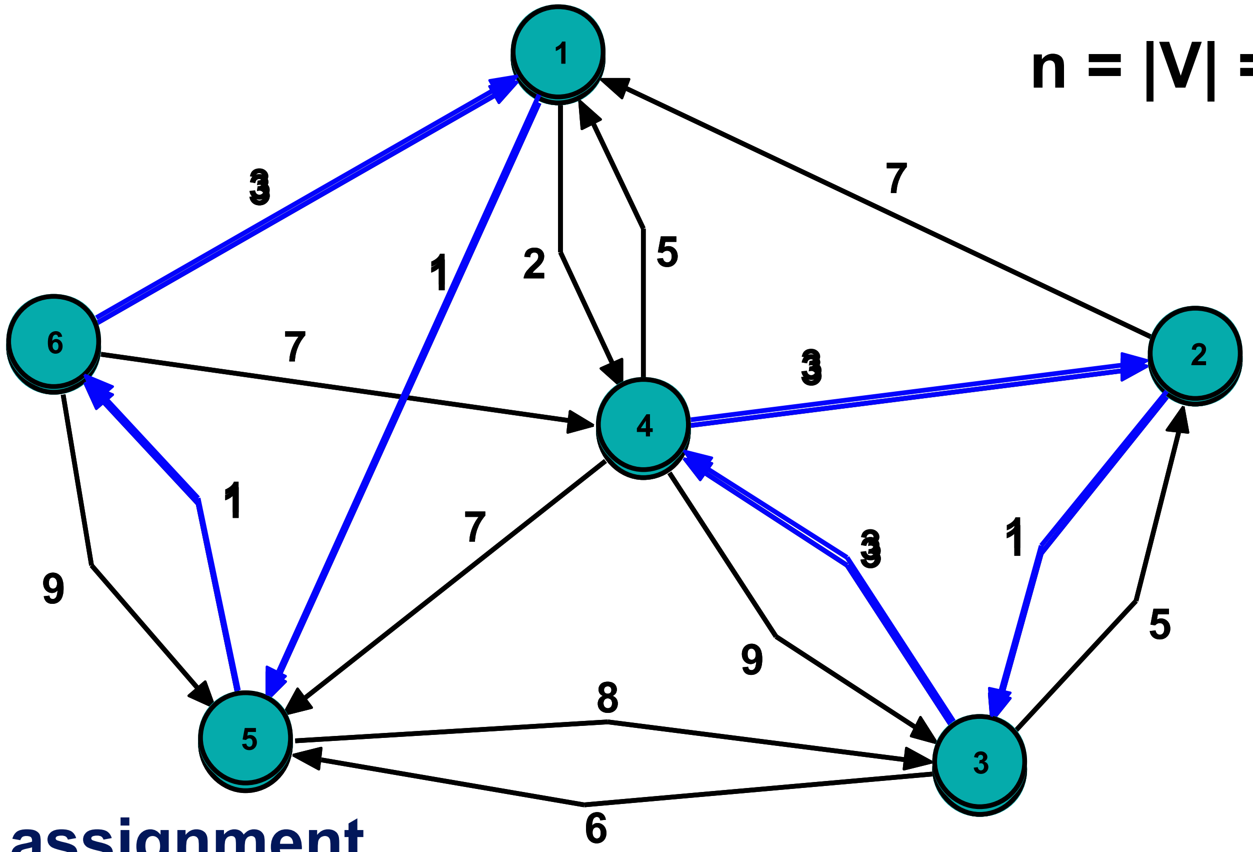
~~$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1 \qquad S \subset V, |S| \geq 2$$~~

~~$$x_{ij} \in \{0, 1\} \qquad i \in V, j \in V$$~~

$$x_{ij} \geq 0 \quad \text{(LP Relaxation)} \quad i \in V, j \in V$$

* The AP solution is given by a family of "*subtours*" (partial circuits)

Example A: AP relaxation of ATSP



Optimal assignment

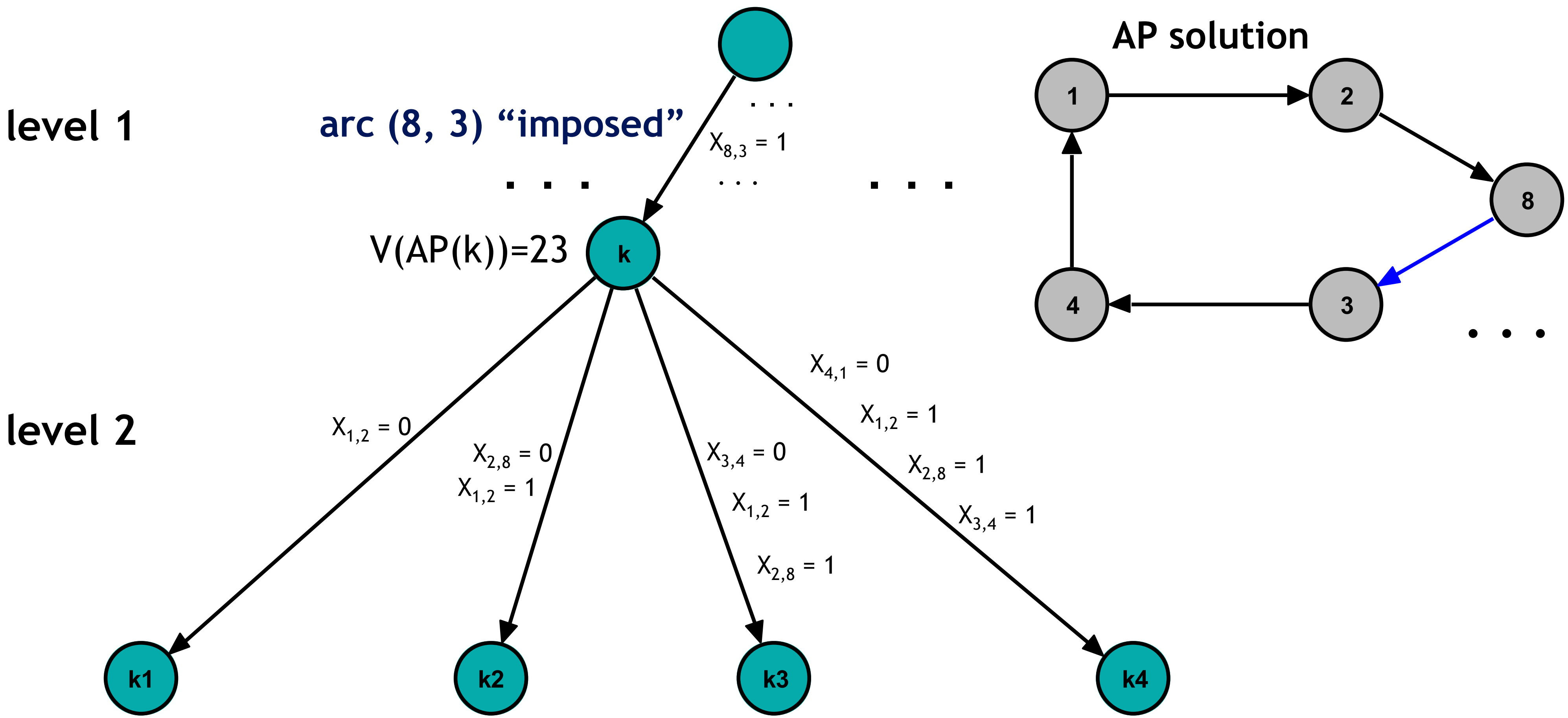
v(AP) = 12 (lower bound)

Optimal solution Cost = 16

BRANCH-AND-BOUND ALGORITHM FOR ATSP

- At each node of the decision tree solve the AP-RELAXATION of the corresponding subproblem.
- If the AP solution contains no subtour (feasible solution), "fathom" the node (possible updating of the best solution so far)
- Otherwise: SUBTOUR-ELIMINATION BRANCHING SCHEME:
  - Select the subtour S with the minimum number h of not imposed arcs.
  - Generate h descendent nodes so as to forbid subtour S for each of them (by "imposing" and "excluding" proper arc subsets).

BRANCHING TREE FOR ATSP



TWO CLASSES OF HEURISTIC ALGORITHMS FOR TSP

1) CONSTRUCTIVE ALGORITHMS

build a Hamiltonian circuit starting from the input data of the original problem (i.e. n, cost matrix c_{ij}).

2) LOCAL SEARCH ALGORITHMS (tour improvement)

starting from an initial feasible Hamiltonian circuit (tour), try to find a tour with a lower cost through a sequence of "moves" corresponding to "arc exchanges" or "vertex exchanges".

CONSTRUCTIVE ALGORITHMS (iterative algorithms)

Main ingredients

a) choice of the "initial partial circuit" (subtour) or of the "initial vertex";

b) choice of the vertex to be inserted, at each iteration, into the current subtour (or into the current "path");

c) choice of the position of the selected vertex in the current solution.

GREEDY ALGORITHM "NEAREST NEIGHBOUR"

Version for STSP

1. Choose any vertex h as "initial vertex" of the current "path".

   Set $i := h$ (last visited vertex),

   $V' := V \setminus \{i\}$ (set of the "unvisited" vertices).

2. Determine the "unvisited" vertex k "nearest" to vertex i
   (k : $c_{ik} = \min \{c_{ij} : j \in V'\}$).

3. Insert vertex k just after vertex i in the current path ($V' := V' \setminus \{k\}$);
   set $i := k$;

   If $V' \neq \emptyset$ (at least one vertex is unvisited) return to STEP 2.

4. Complete the Hamiltonian circuit with arc (i, h);

   STOP.

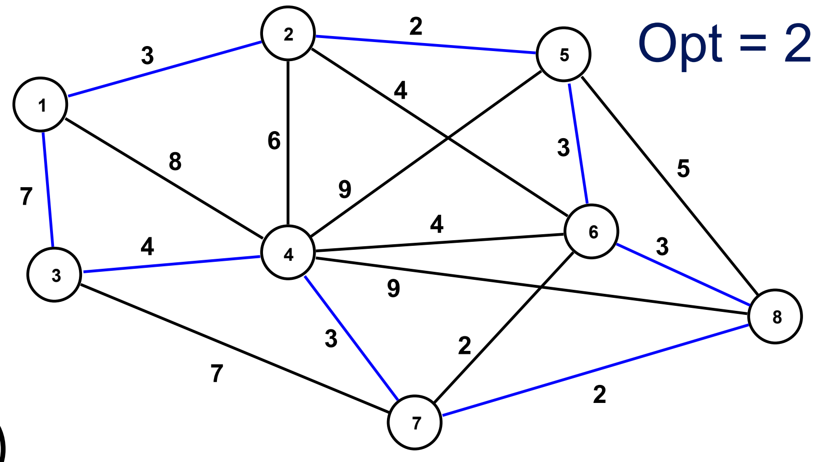
GREEDY ALGORITHM "NEAREST NEIGHBOUR" (2)

- Time complexity: $O(n^2)$.

- Different choices of the "initial vertex" lead to different solutions.

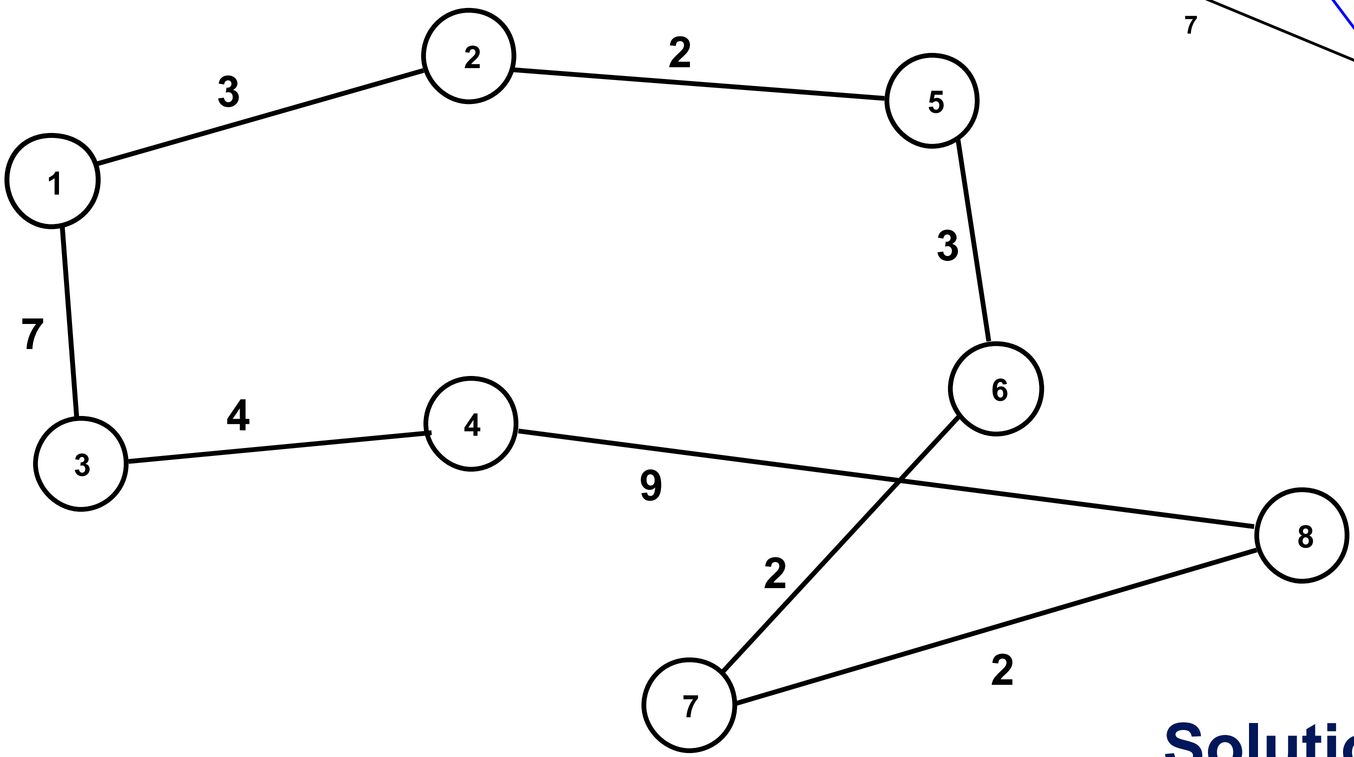
The same algorithm can be used for ATSP

Example 1 (Alg. Nearest Neighbour)



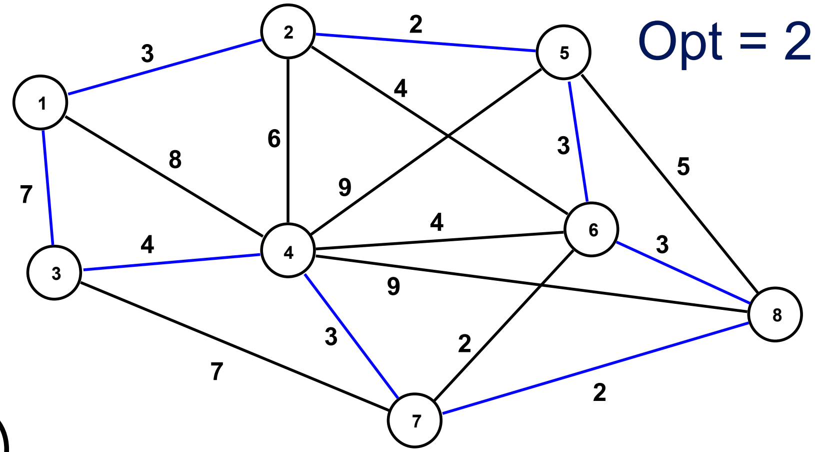
Opt = 27

Initial vertex: 1



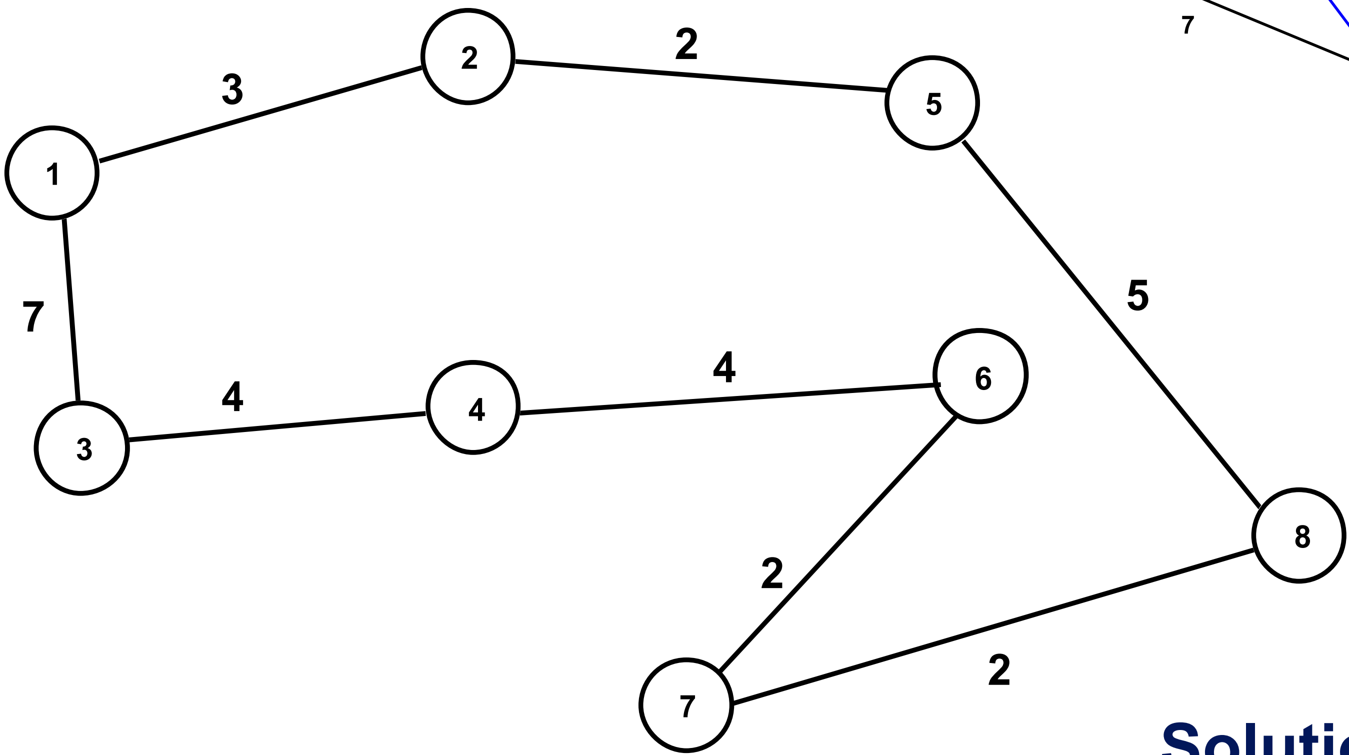
Solution cost: 32

Example 1 (Alg. Nearest Neighbour)



Opt = 27

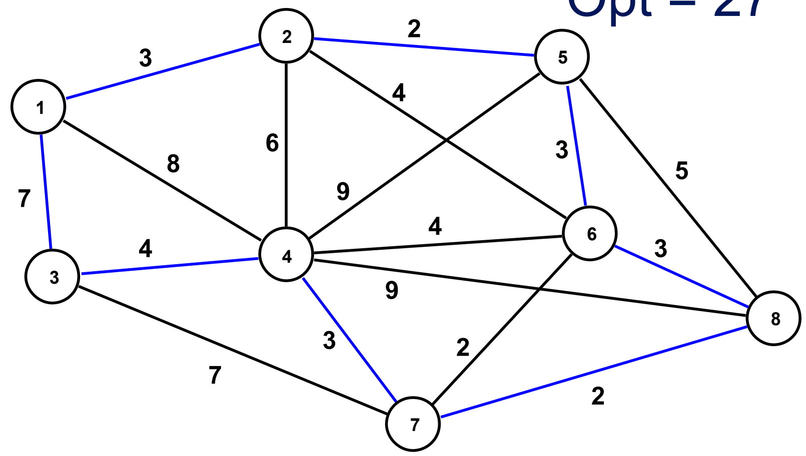
Initial vertex: 6



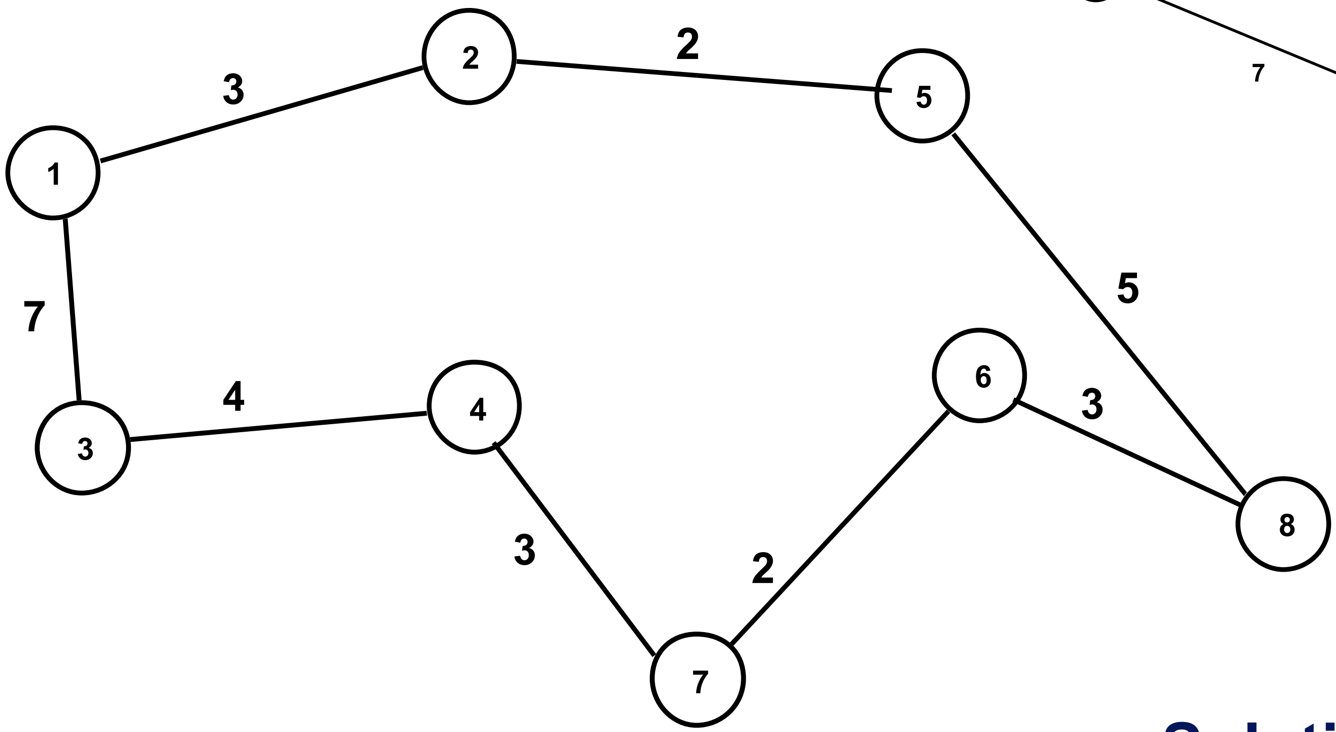
Solution cost: 29

Example 1 (Alg. Nearest Neighbour)

Opt = 27

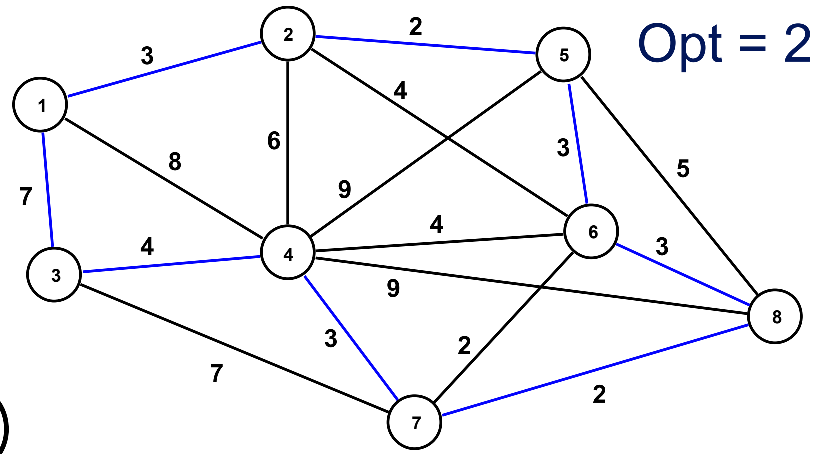


Initial vertex: 7 (second vertex: 6)



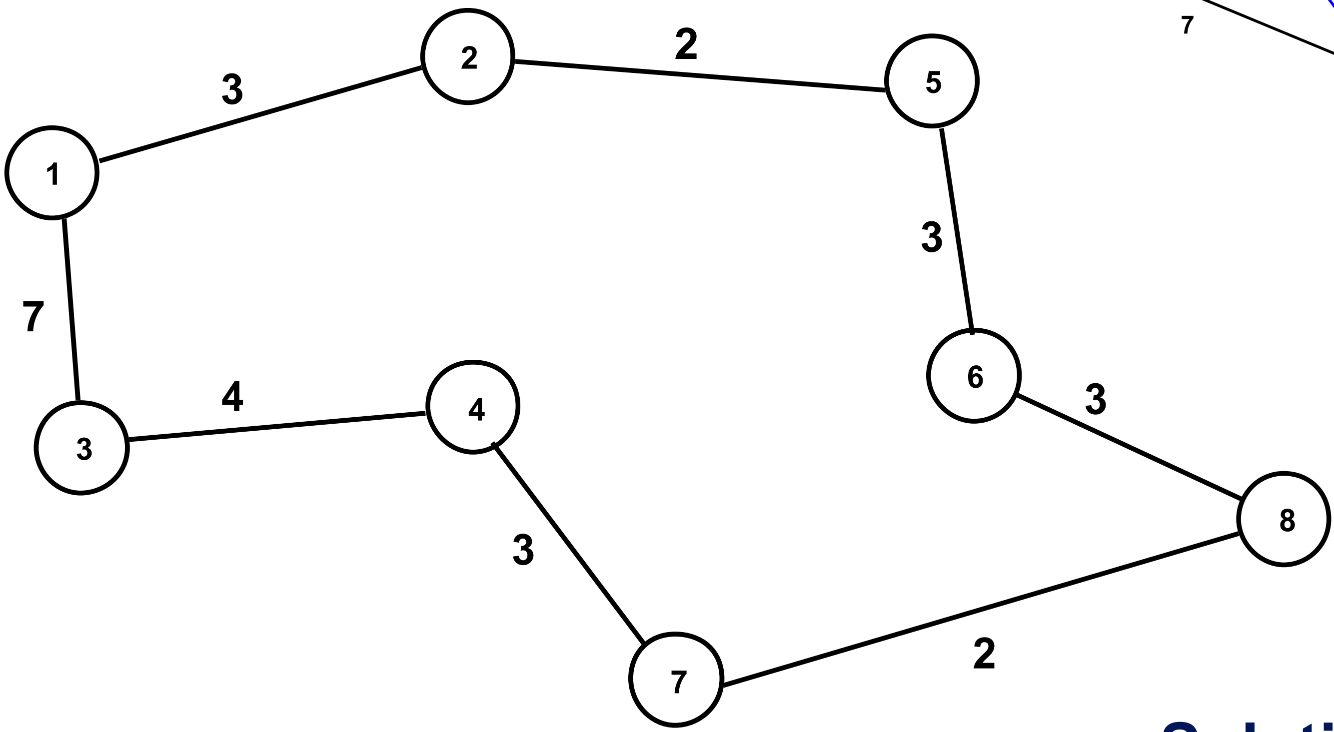
Solution cost: 29

Example 1 **(Alg. Nearest Neighbour)**



Opt = 27

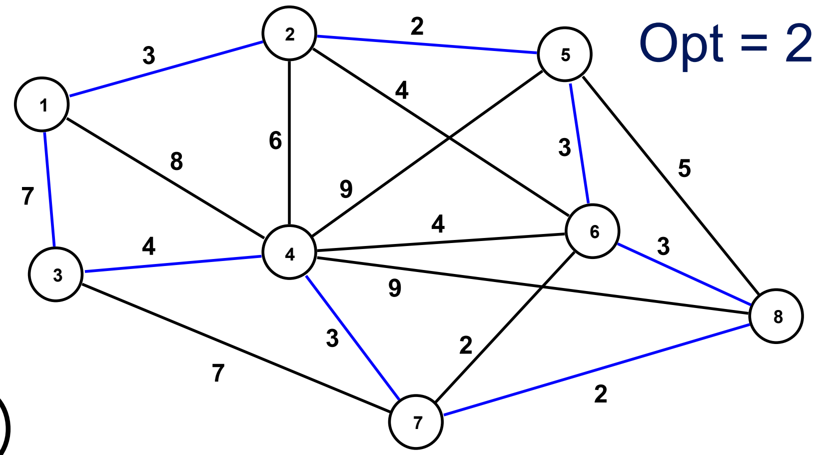
Initial vertex: 7 (second vertex: 8)



Solution cost: 27

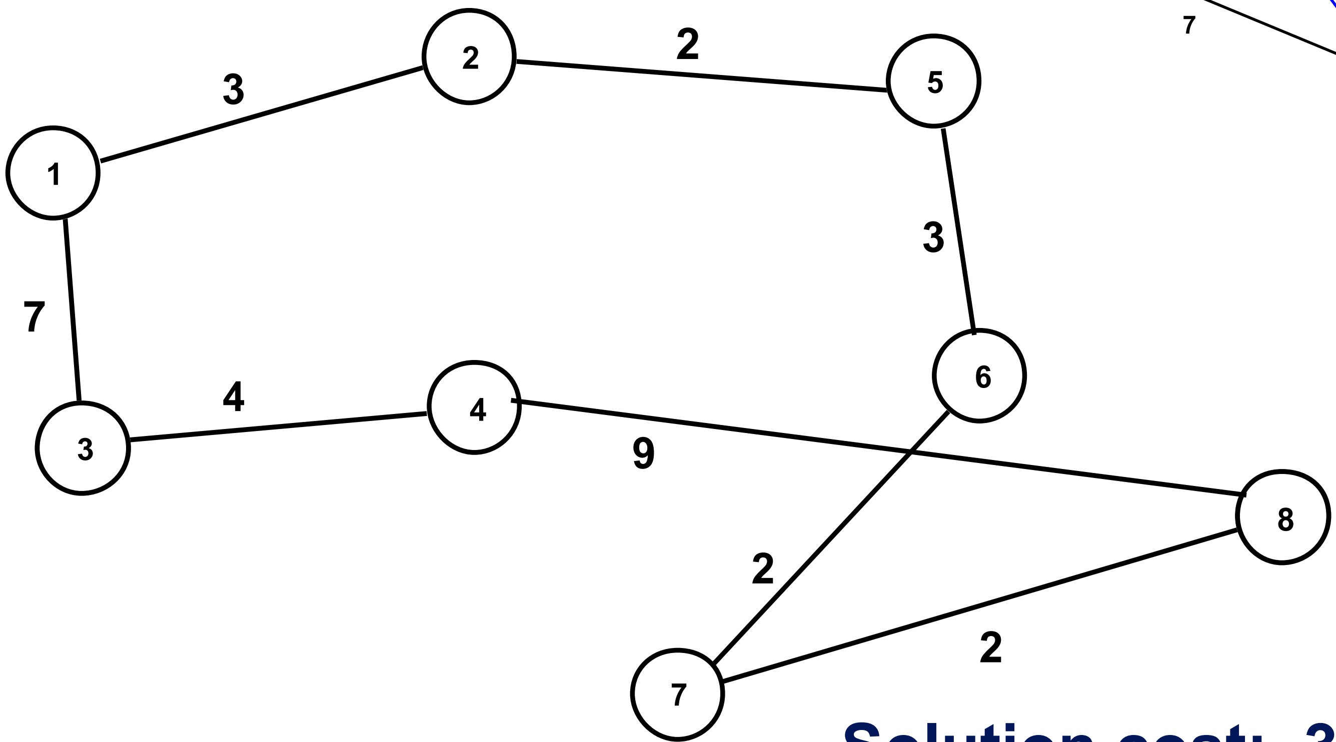
(optimal solution)

Example 1 (Alg. Nearest Neighbour)



Opt = 27

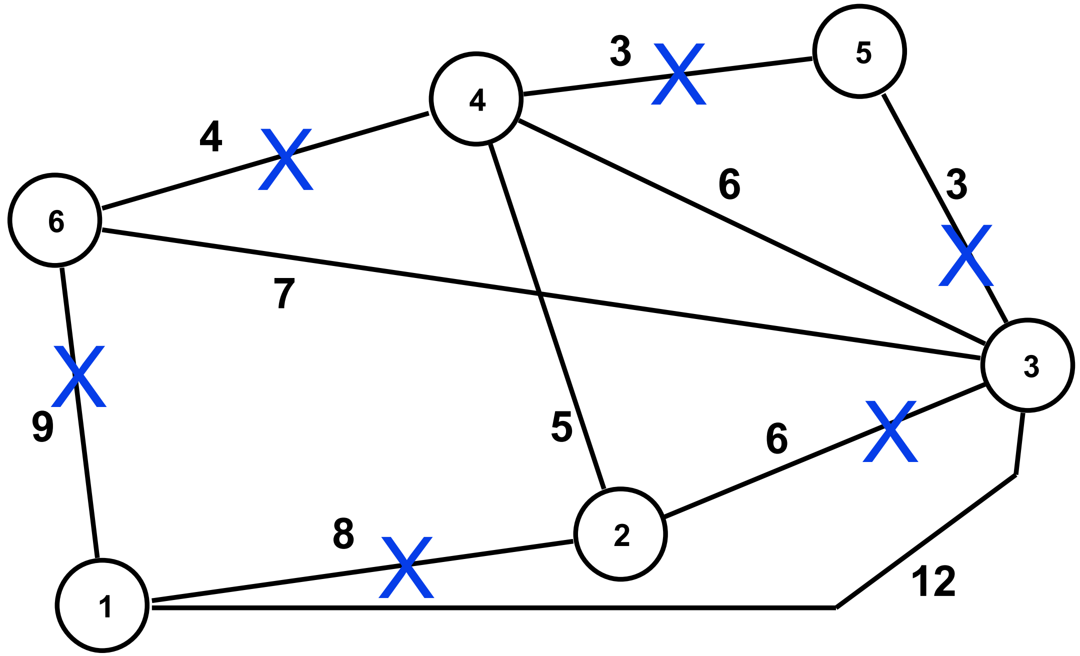
Initial vertex: 8



Solution cost: 32

(same solution as that found with "Initial vertex": 1)

Example 2

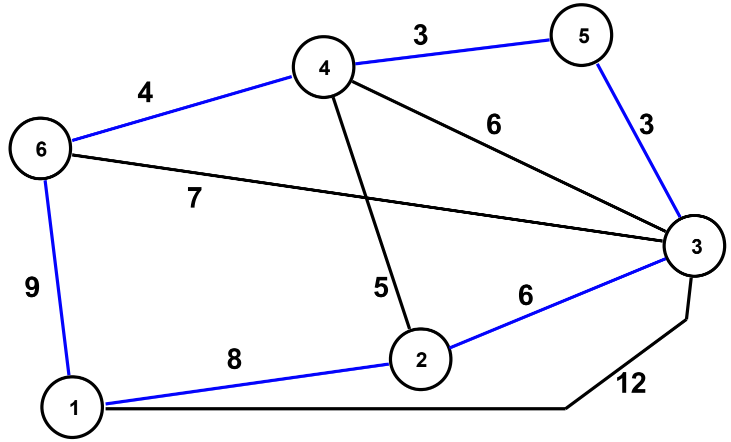


n = 6

Optimal solution X

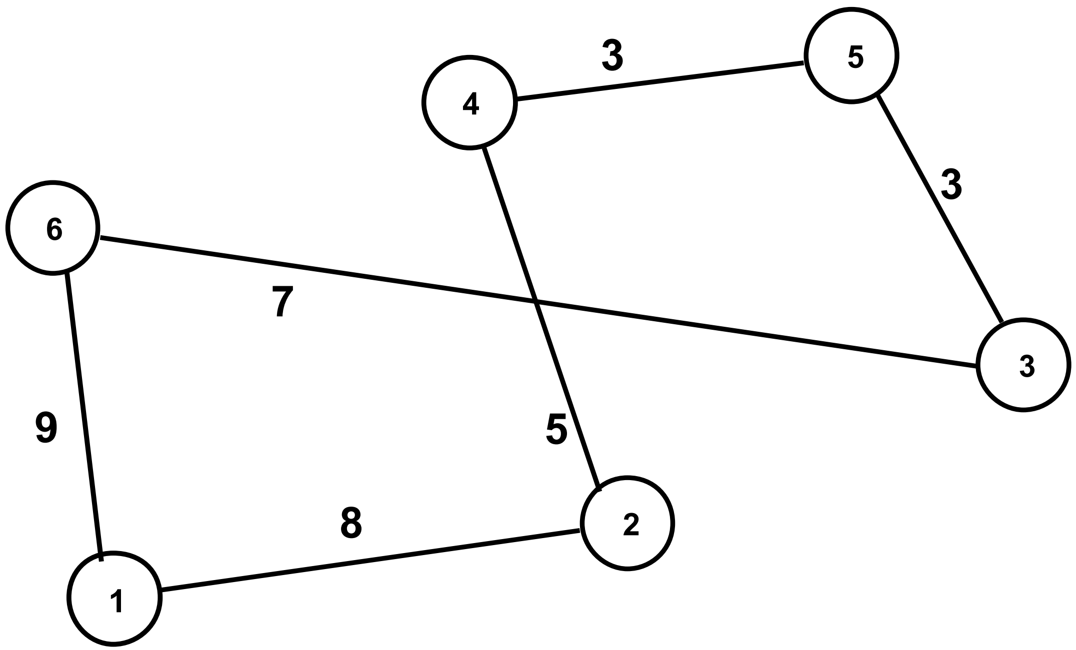
Optimal solution cost: Opt = 33

Example 2 (Alg. Nearest Neighbour)



Opt = 33

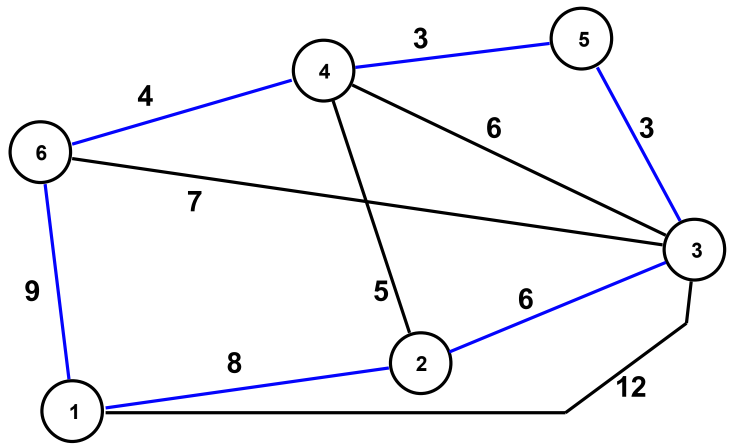
Initial vertex: 1



Solution cost: 35

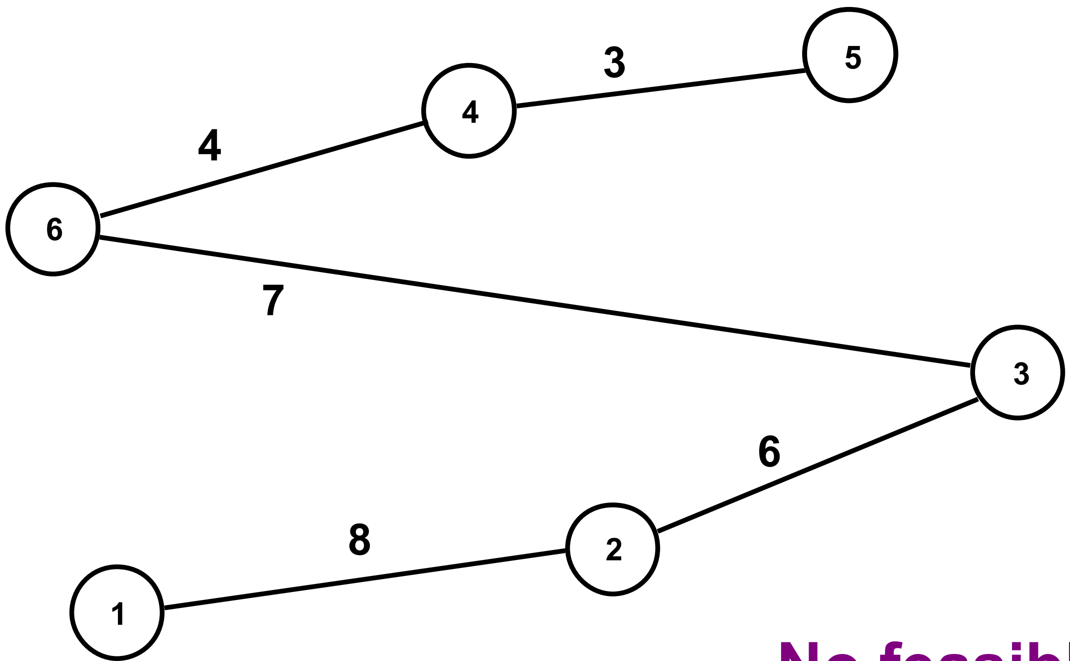
Same solution found with “Initial vertex”: 2

Example 2 (Alg. Nearest Neighbour)



Opt = 33

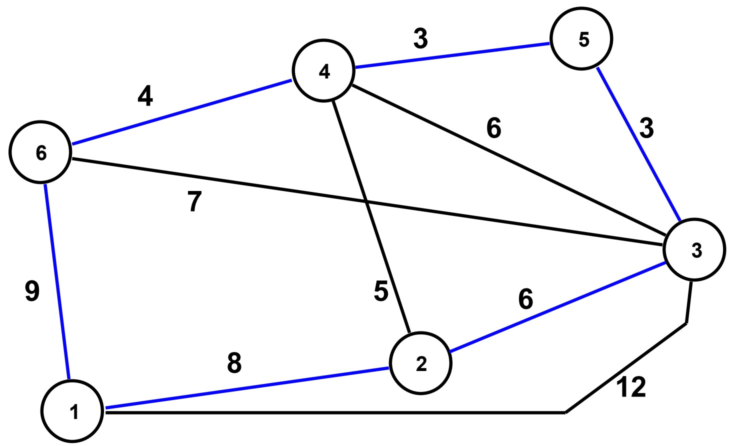
Initial vertex: 5



No feasible solution found

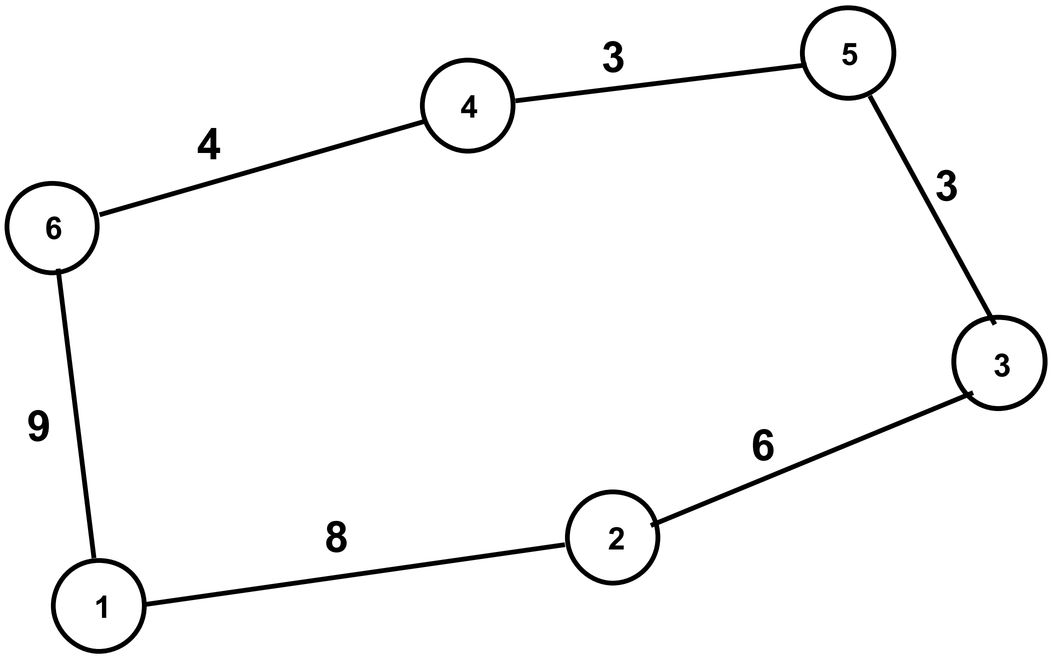
(with alternative choices as well)

Example 2 (Alg. Nearest Neighbour)



Opt = 33

Initial vertex: 3

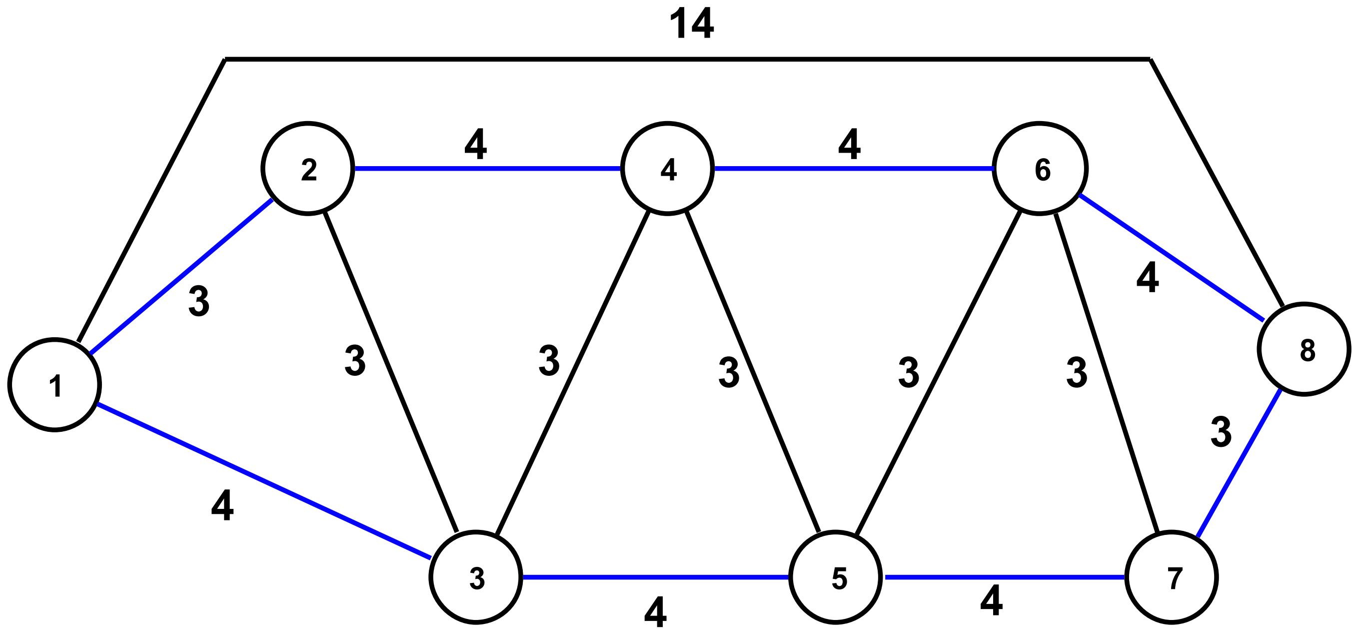


Solution cost: 33

(Optimal solution)

Same solution found with “Initial vertex”: 4, 6

Example B

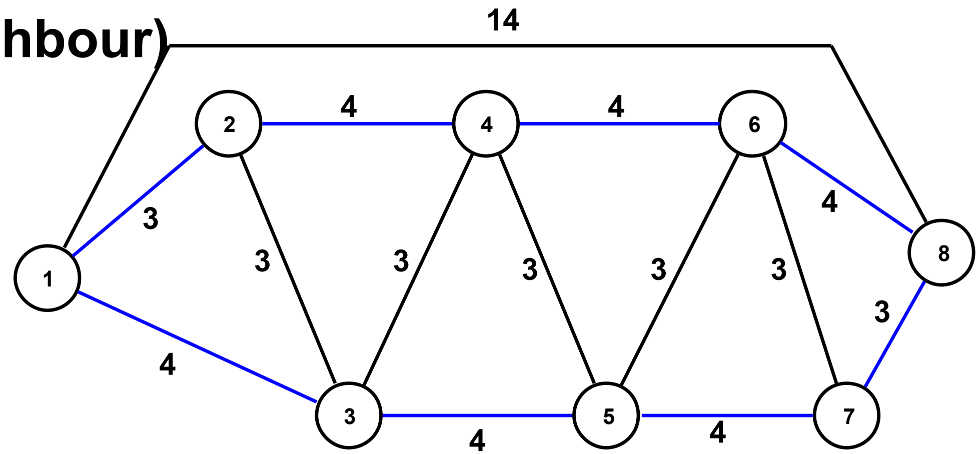


n = 8

Optimal solution

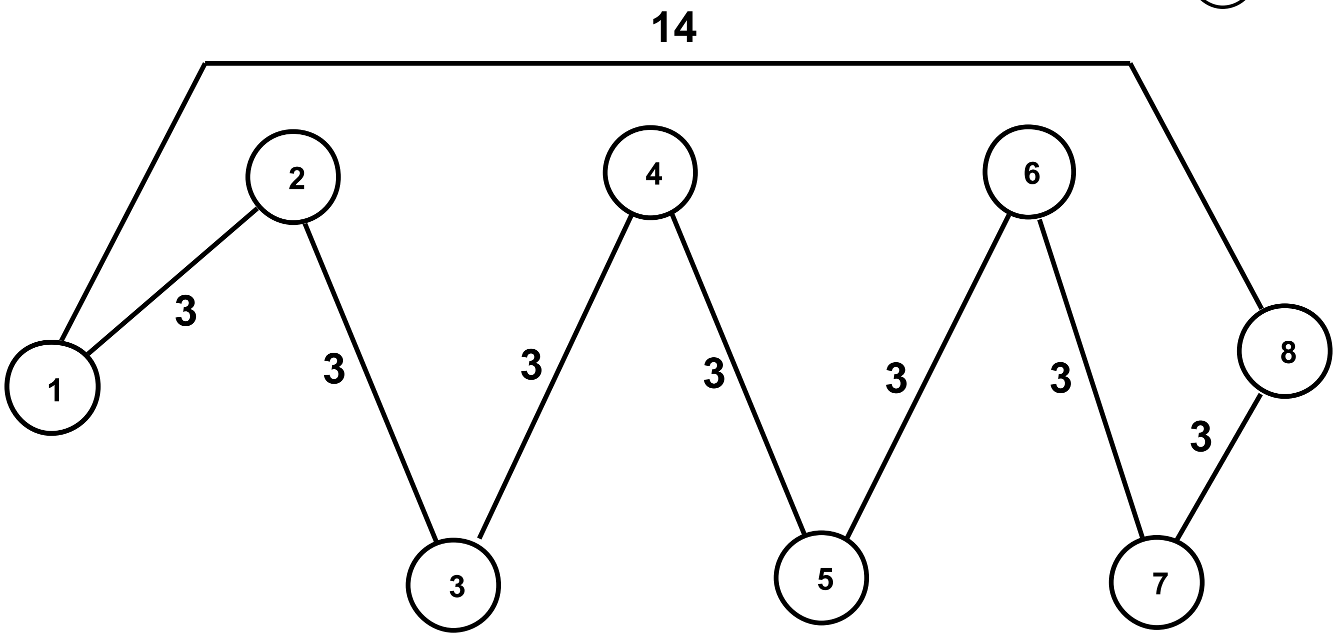
Optimal solution cost: Opt = 30

Example B (Alg. Nearest Neighbour)



Opt = 30

Initial vertex: 1

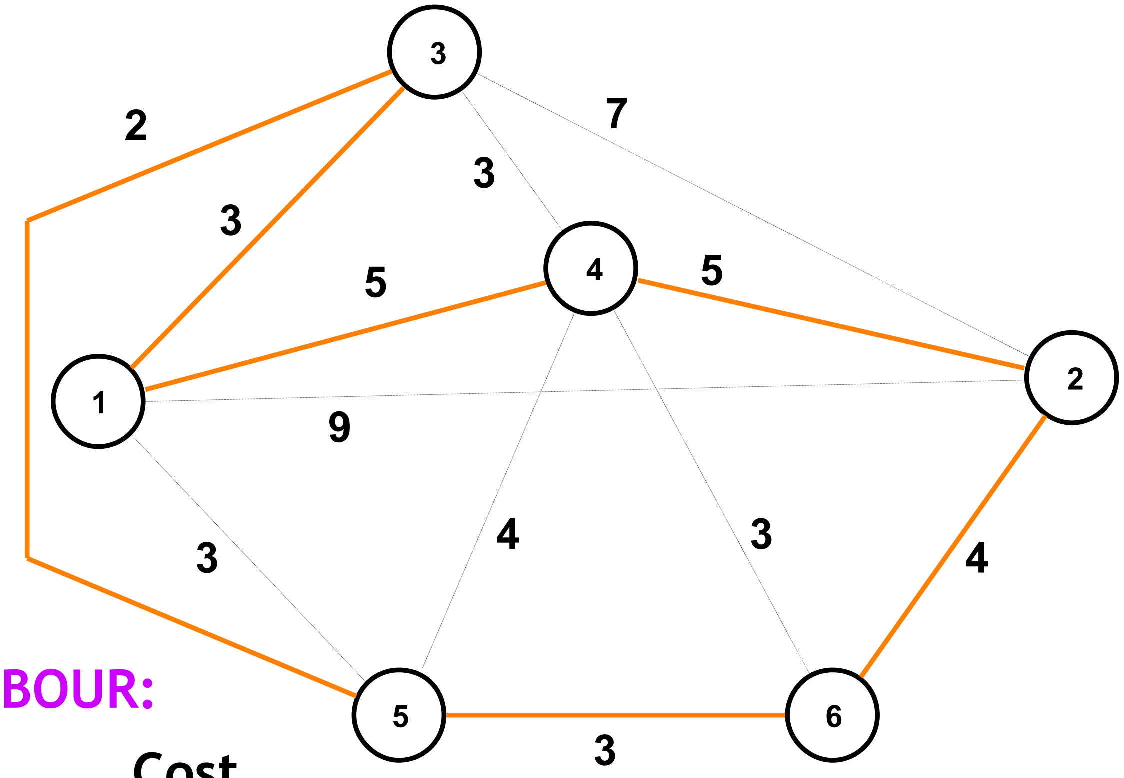


Solution cost: 35

No better solution found with "Initial vertex": 2, 3, 4, 5, 6, 7, 8

Example 3

when ties are present one can construct alternative solutions



n = 6

Optimal solution

Optimal solution cost = 21

ALGORITHM NEAREST NEIGHBOUR:

Initial vertex	Solution	Cost	
1	red	25	(blue 24)
2	blue	24	(orange 22, black 26)
3	purple	24	(red 25, green 29)
4	blue	24	(red 25)
5	blue	24	(grey no feasible sol. found)
6	orange	22	(. . . >= 24)