Lecture #23
State Space Search (1)

Algorithm

JBNU

Jinhong Jung

In This Lecture

❑ State Space Search

- Concept of state space tree

- Techniques for state space search
  - Backtracking
  - Bounded branch

Outline

- ❑ Overview
- ❑ State Space Tree
- ❑ Backtracking
- ❑ Bounded Branch

Algorithmic Strategies

- Algorithmic strategies commonly used for designing algorithms to solve a problem
  - ✅ Brute-force method
    - Enumerate all possible cases of the problem
  - ✅ Divide and conquer
    - Divide the problem into sub-problems & solve them recursively
  - ✅ Dynamic programing
    - Memo the optimal solutions of sub-problems and re-use them
  - ✅ Greedy algorithm
    - Always make the choice that seems to be the best at that moment
  - State space (combinatorial) search ⇐ Today's topic

Motivation

- Suppose we need to obtain an optimal solution of a problem
  - What if it is hard to find optimal sub-structure?
  - What if there is no overlapping sub-problems?
  - What if it does not have a greedy choice property?
- If there are no other options, how can do solve it?
  - A brute-force method solves it definitely
- Q. But it’s too slow! Is there an efficient way?
  - A. State space search with bounded branch!

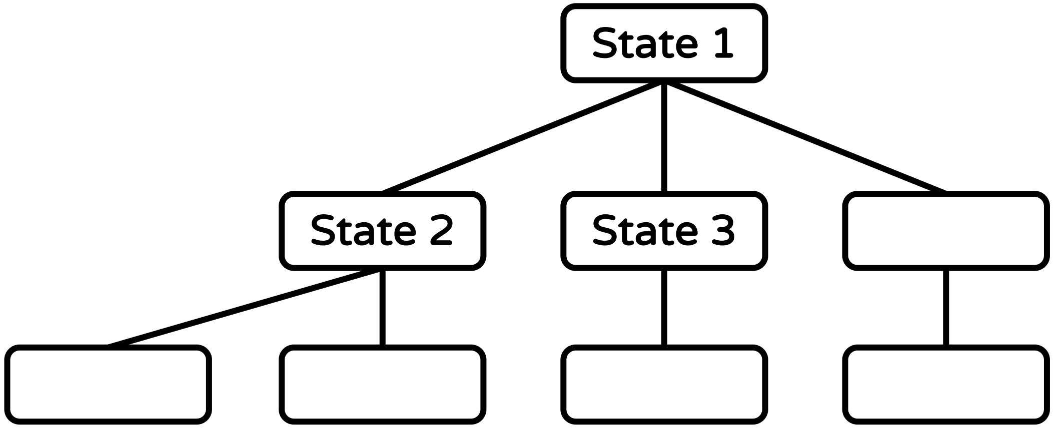
Outline

- Overview
- State Space Tree
- Backtracking
- Bounded Branch

State Space Tree

❑ State space tree of an algorithm for a problem

- **State:** an intermediate state in a process solving a problem
- **State space tree:** a tree constructed from all of the possible states as nodes
  - Connected via state transitions from initial state (root) to terminal state (leaf)

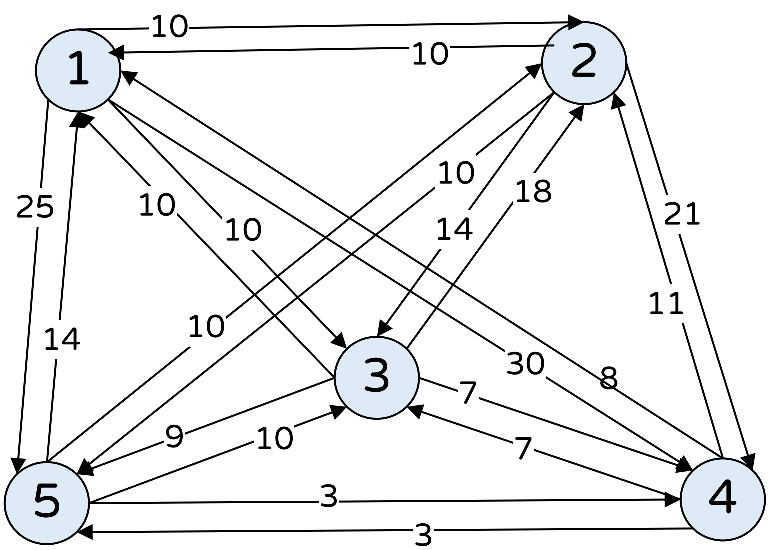


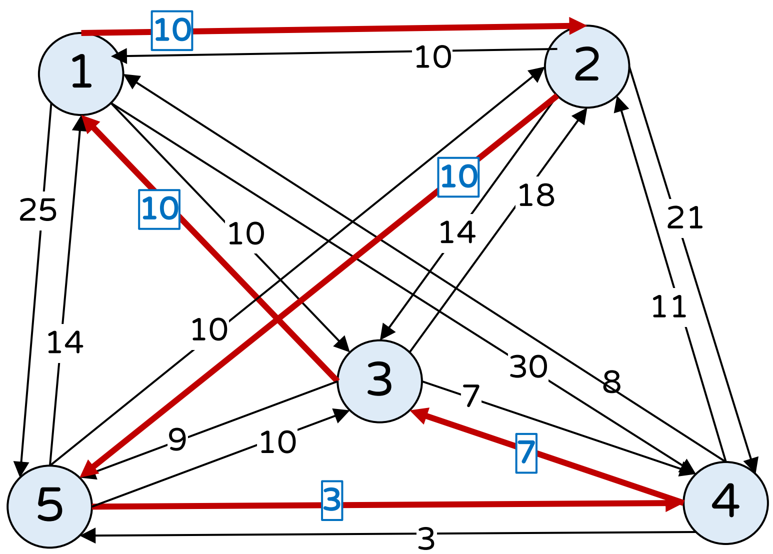
Imagine you enumerate all possible cases

Example: SST of TSP

❑ Asymmetric TSP

- Find the shortest Hamiltonian cycle in a directed graph?
  - Hamiltonian cycle is a directed cycle visiting all nodes exactly once
- Also, it’s NP-Hard, but we use this problem to describe SST and bounded branch.

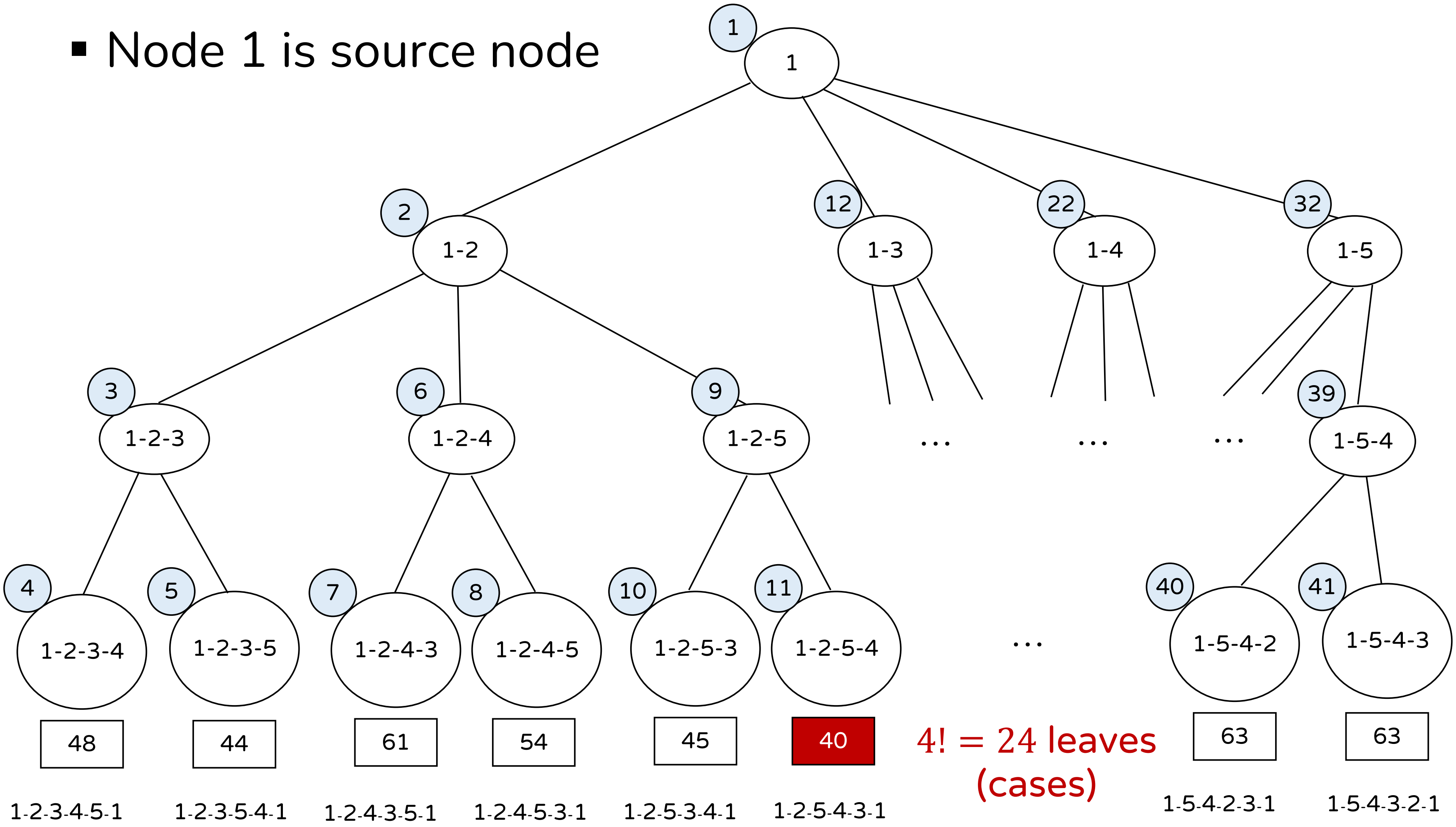




Example: SST of TSP

❑ SST of Brute-force Search for TSP

- Node 1 is source node



Outline

- Overview
- State Space Tree
- **Backtracking**
- Bounded Branch

Backtracking

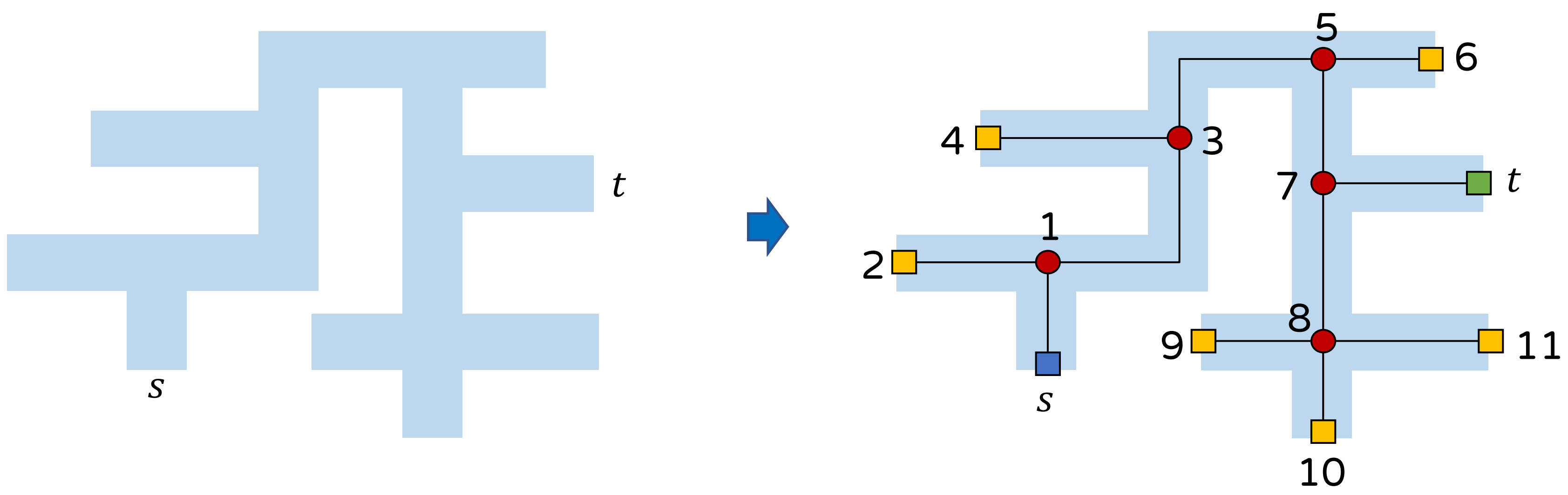
- **Solving a problem is equal to**
  - Searching for an answer in its state space tree ⇒ **state space search**

- **Backtracking is a technique for state space search**
  - A backtracking algorithm performs DFS (depth first search) in the state space tree (SST)
    - DFS goes as deeply as possible, and **backtracks** when it reaches a dead-end
    - Don't need to explicitly construct an SST because DFS's (or backtracking's) process corresponds to the state space search

Example Of Backtracking (1)

❑ Maze problem

- Find a path from a starting point s to a target point t in a maze ⇒ Do DFS from s
  - Branch (○) and dead-end (▢)



Example Of Backtracking (2)

❑ Backtracking algorithm for Maze problem

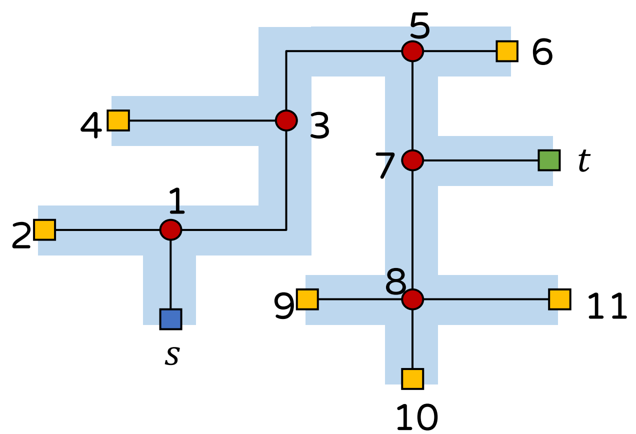
- Do DFS from s in the maze

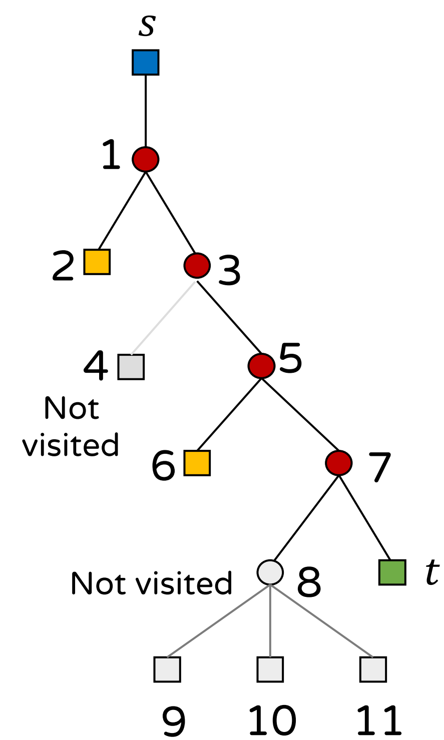
```
def maze(u):
    visited[u] ← true

    if u == t:
        return true

    for each v in N(u):
        if visited[v] == false:
            prev[v] ← u
            maze(v)
```

Backtracking algorithm
for Maze problem





State space tree
incurred by DFS

Backtracking v.s. Brute-force

- **Backtracking can be more efficient than brute-force**
  - Brute-force method enumerates all possible cases
    - e.g., There are k^n cases for graph coloring problem
    - However, many of them are not valid ⇒ don't need to see all of them
  - Backtracking tries to search for all possible cases
    - But **unpromising states could be pruned** during DFS!
      - If the backtracking algorithm has pruning strategies
    - The searching cost from pruned states is saved
- **Let's check a pruning technique called bounded branch**

Outline

- ❑ Overview
- ❑ State Space Tree
- ❑ Backtracking
- ❑ **Bounded Branch**

Bounded Branch

❑ Let's consider TSP problem again

- There are $(n-1)!$ cases in a graph of node n
- However, we don't need to see all of them
  - If the length of a partial path is greater than the best solution we've found so far, the partial path is very unlikely to be a better solution

❑ Main idea of bounded branch (branch & bound)

- Let's prune unpromising states during backtracking to make the **branches** of SST **bounded**
  - Need to search for all possible cases (backtracking)
  - Need to measure the unpromising-ness of a state (pruning)

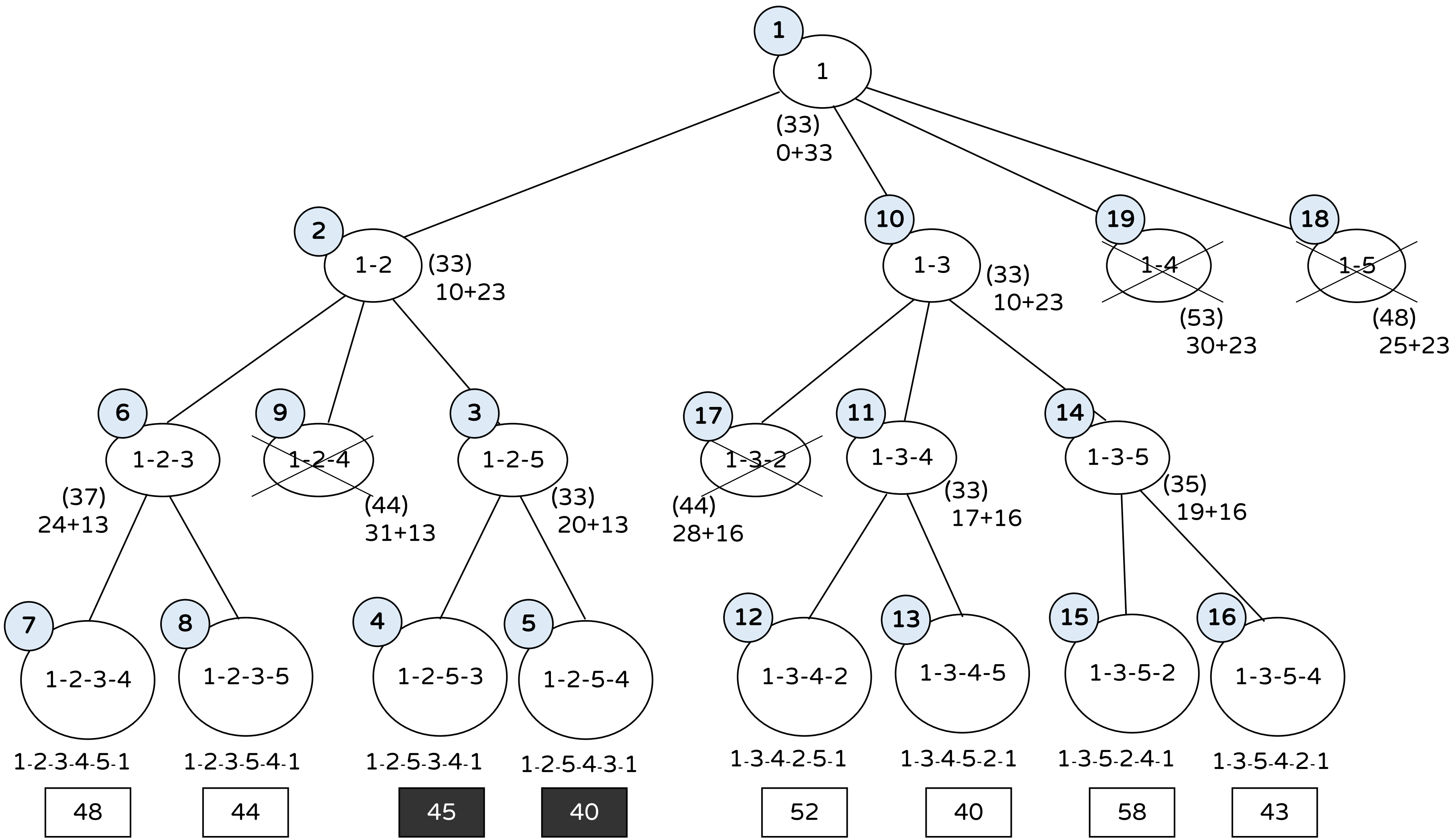
Pruning Technique

❑ How to judge whether a state is pruned?

- In general, we judge it based on the best solution found so for (:= base solution)
- The better the quality of the base solution, the more branches are pruned
  - If a good solution is initially found, the bounded branch is likely to quickly end
  - Otherwise, it could check all possible cases for a worst case
- A practical method for setting base solution
  - Use a good approximate algorithm & set its output to the base solution
  - Then, start the bounded branch algorithm with the base solution

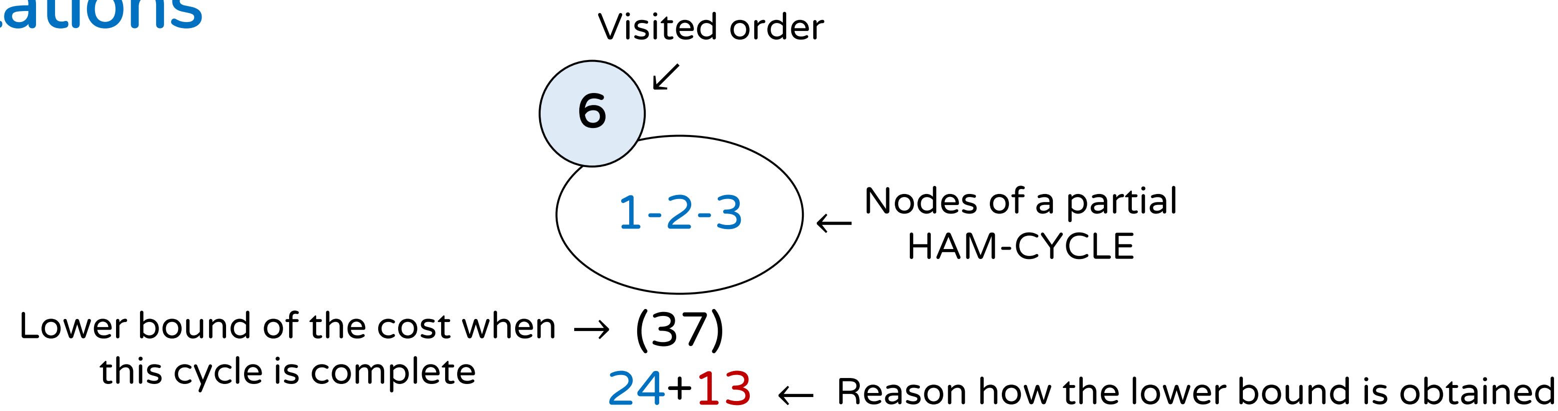
Example Of Bounded Branch

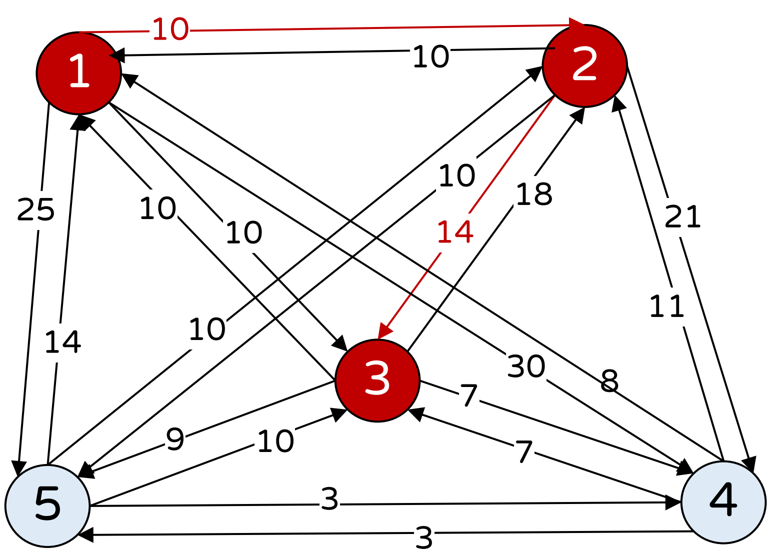
❑ SST of a bounded branch algorithm for TSP



Example Of Bounded Branch

❑ Notations





HAM-CYCLE: 1-2-3-(4,5)-1

- 24: 10 + 14 from 1-2-3
- 13: 7 + 3 + 3
  - Node 3: 7
  - Node 4: 3
  - Node 5: 3

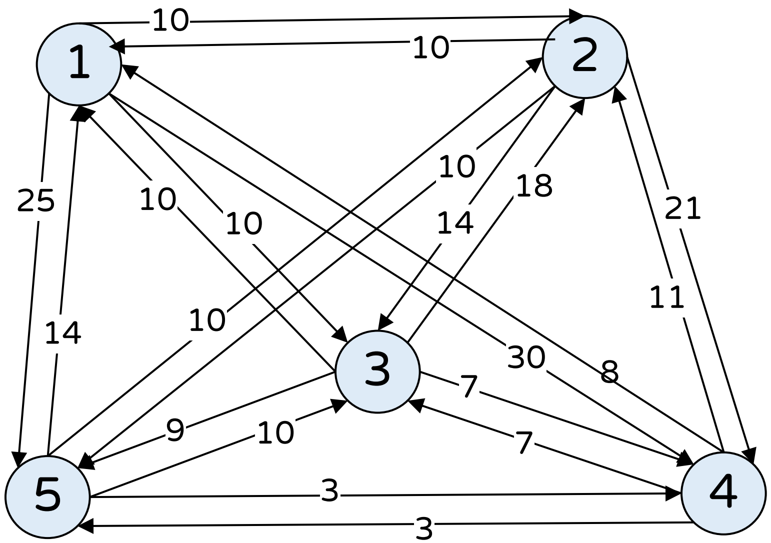
Select the minimum weight of out-going edges for each node

At least 13 cost is needed to compte the cycle

Example Of Bounded Branch

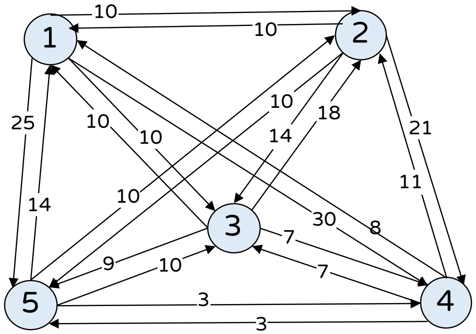
❑ Initial phase

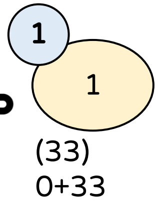
- Find the lower bound of HAM-CYCLE selecting the minimum weight of out-going edges for each node
  - Lower bound: 33
    - Node 1: 10
    - Node 2: 10
    - Node 3: 7
    - Node 4: 3
    - Node 5: 3



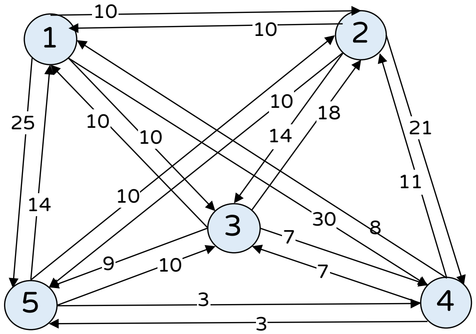
- Why?
  - A Hamiltonian cycle has n edges building a path of n nodes
    - In the path, a node has an out-going edge
  - Thus, an optimal solution cannot be less than the lower bound

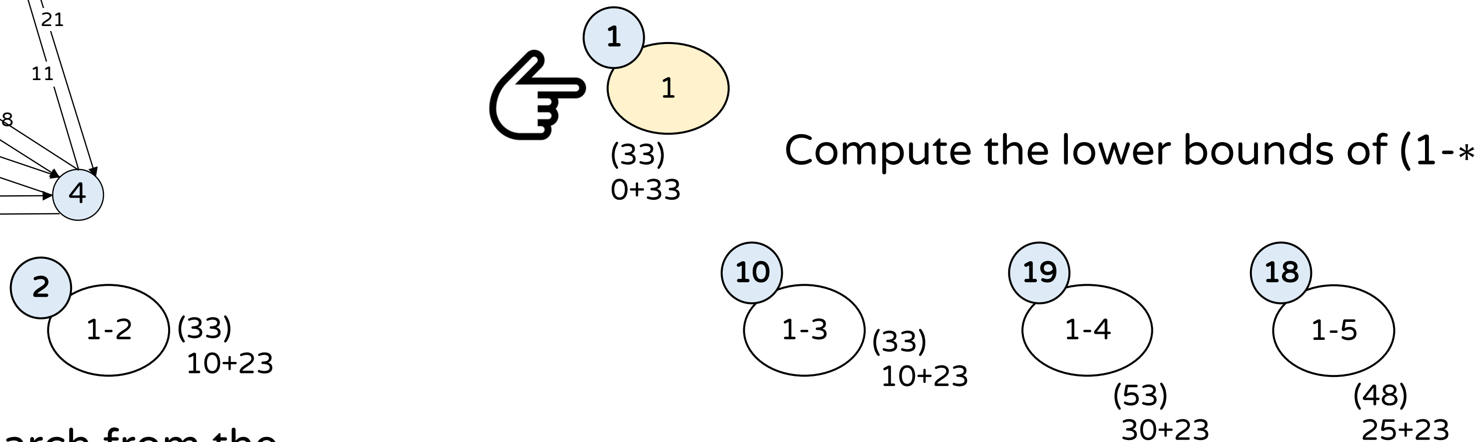
Example Of Bounded Branch





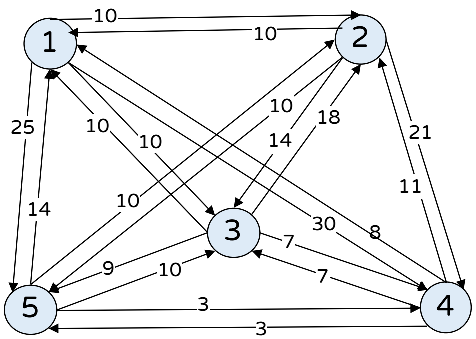
Example Of Bounded Branch

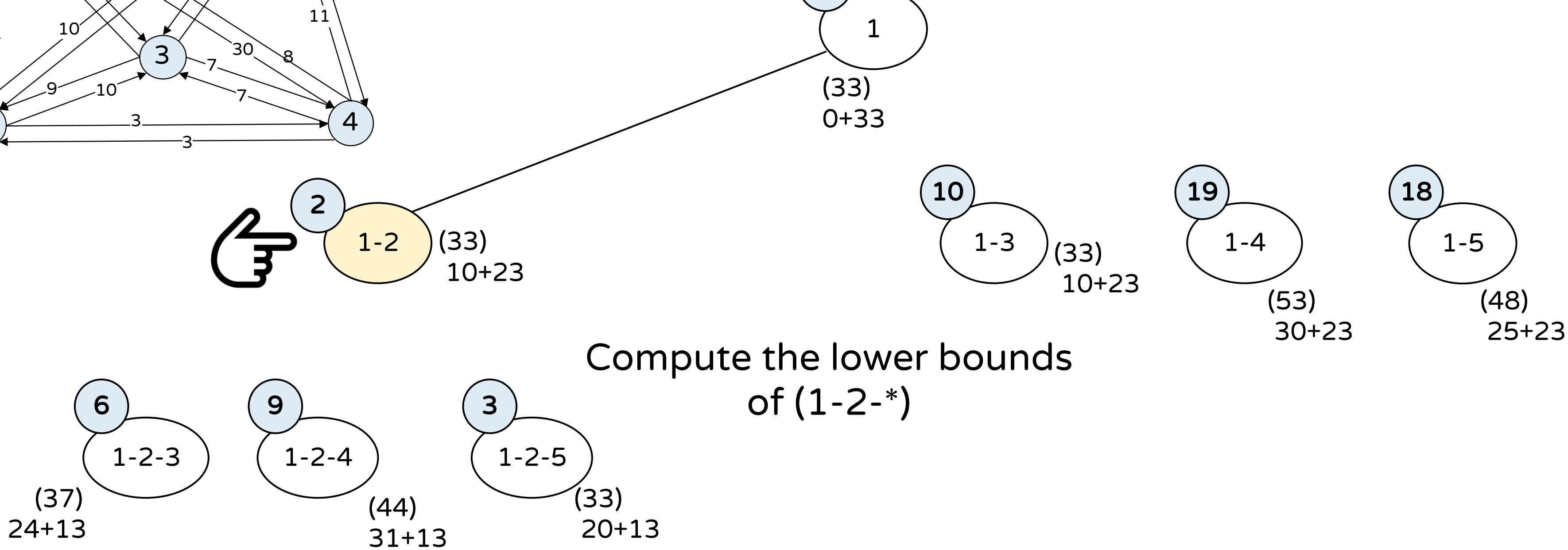




Search from the
lowest lower bound!

Example Of Bounded Branch

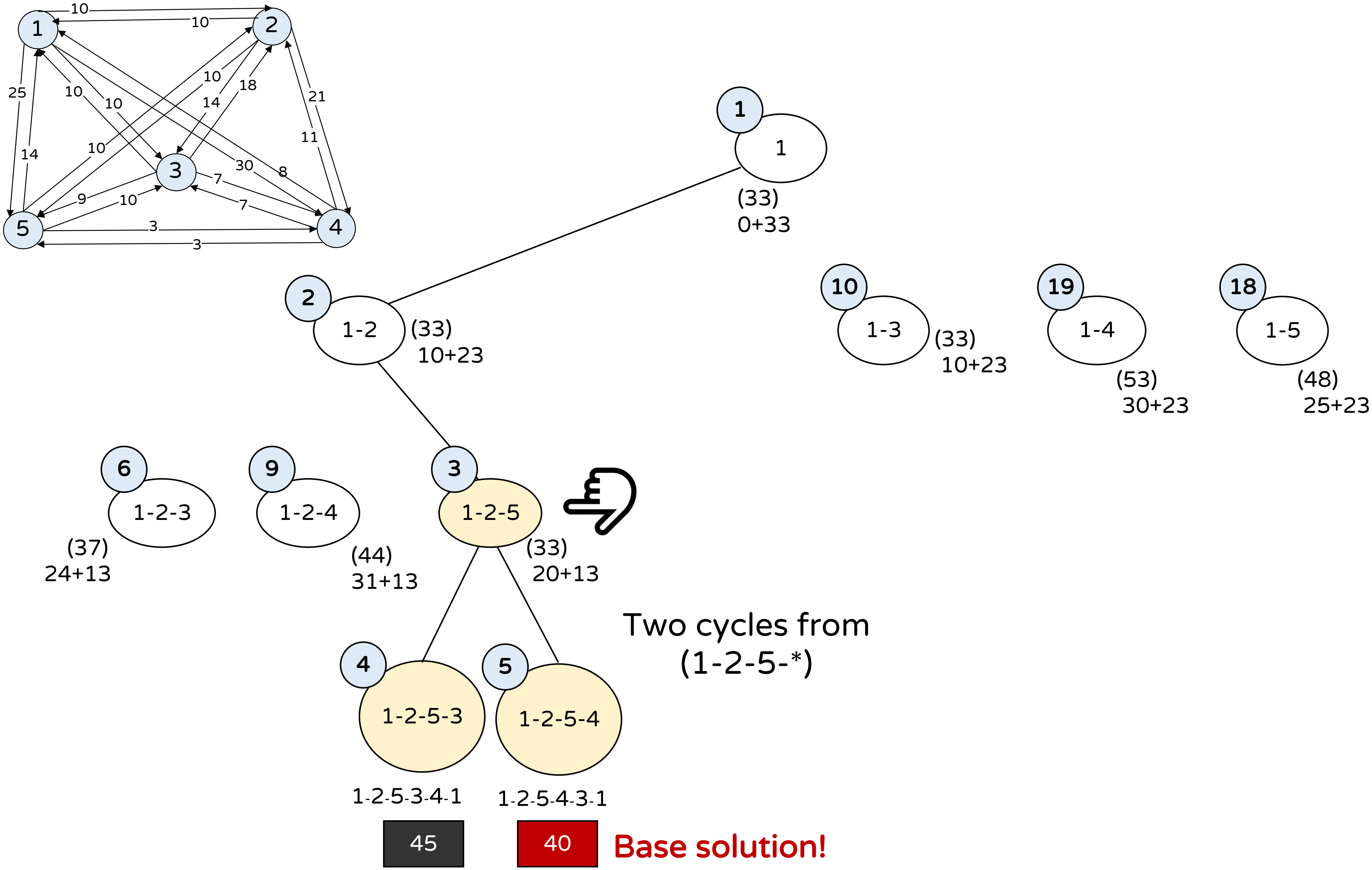




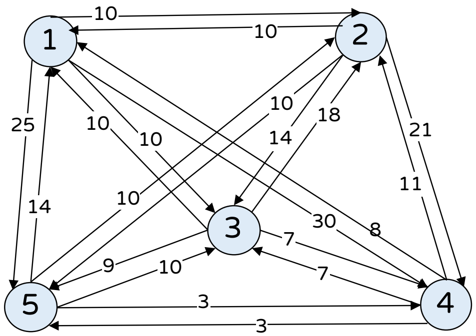
Compute the lower bounds
of (1-2-*)

Search from the
lowest lower bound!

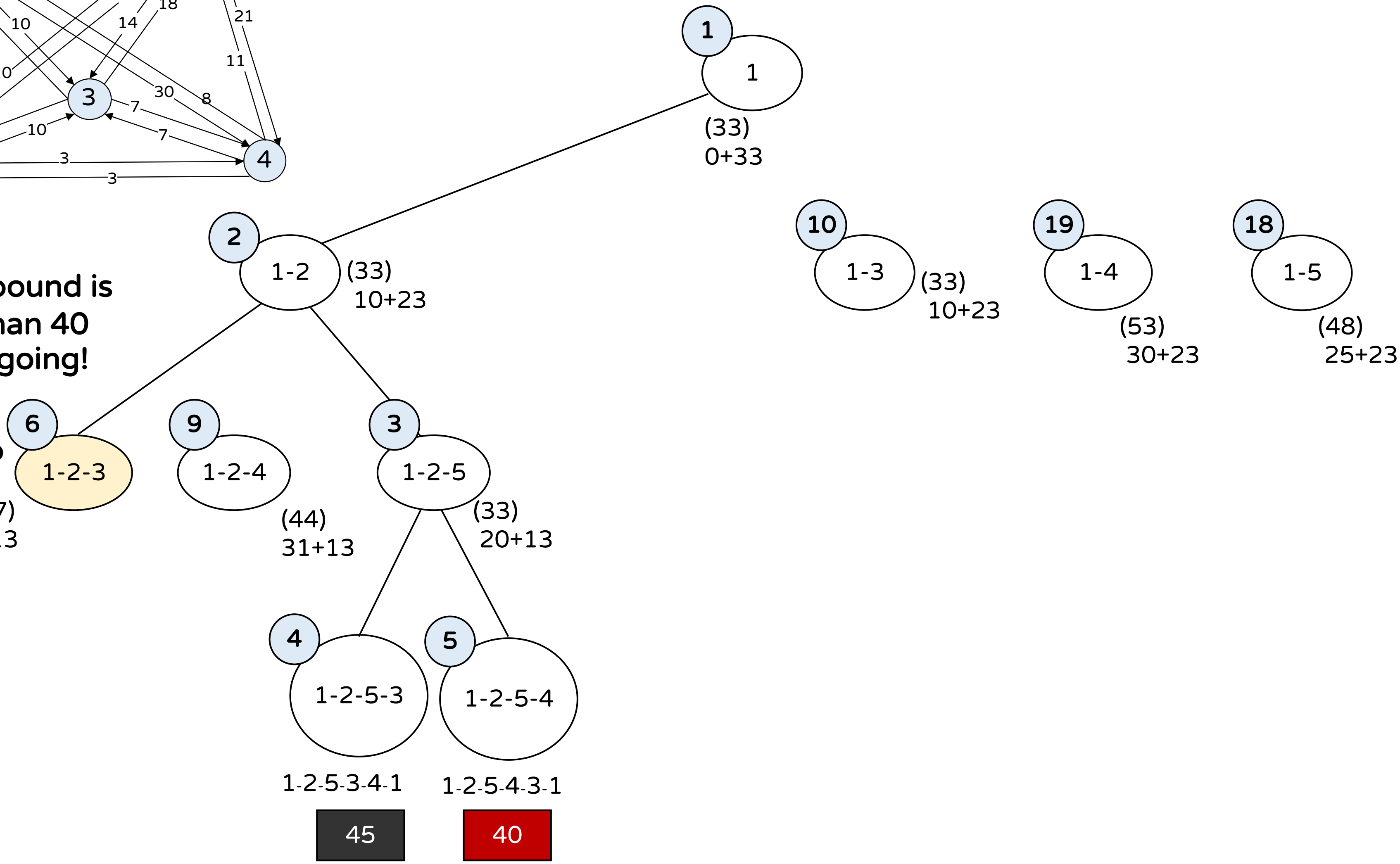
Example Of Bounded Branch



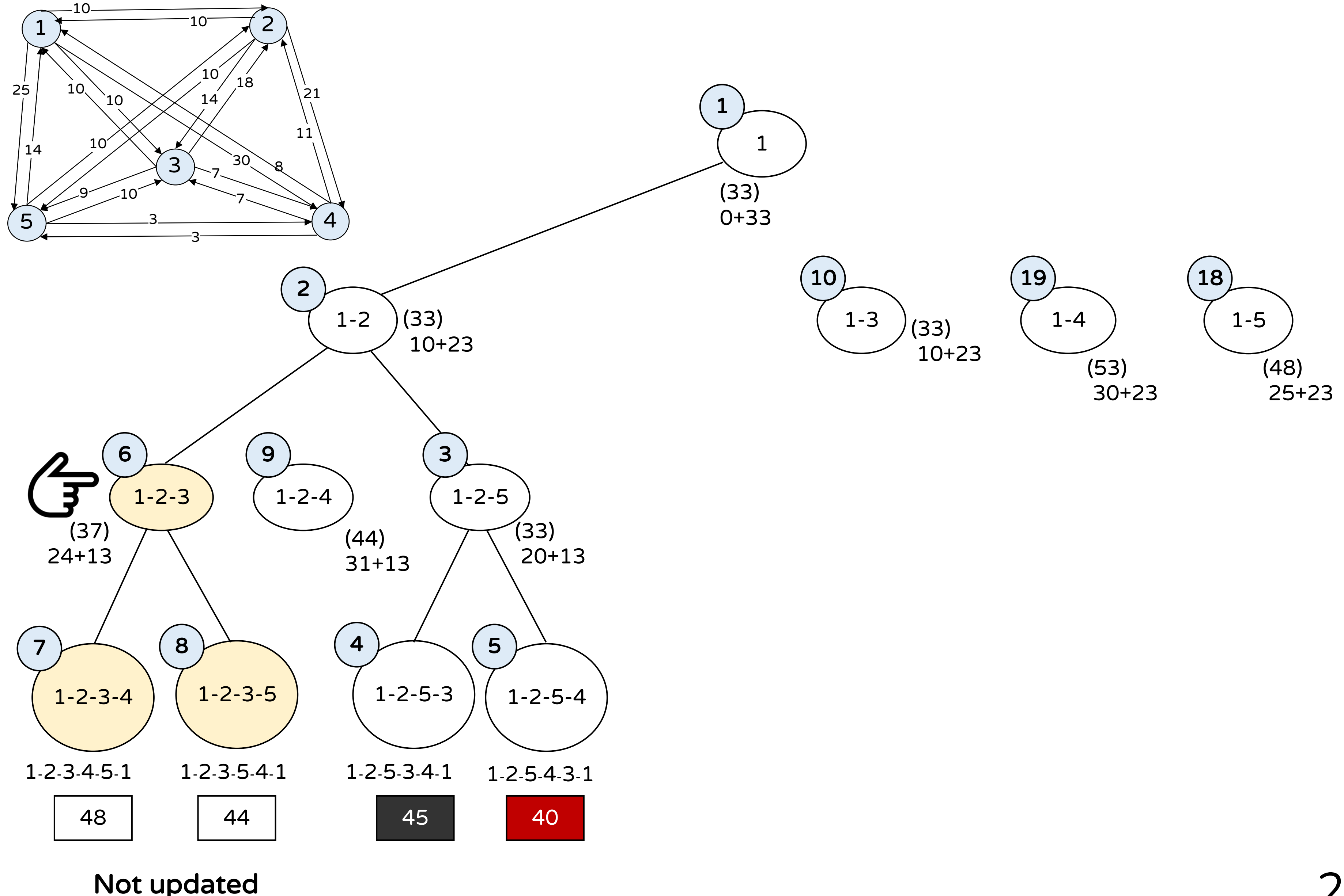
Example Of Bounded Branch



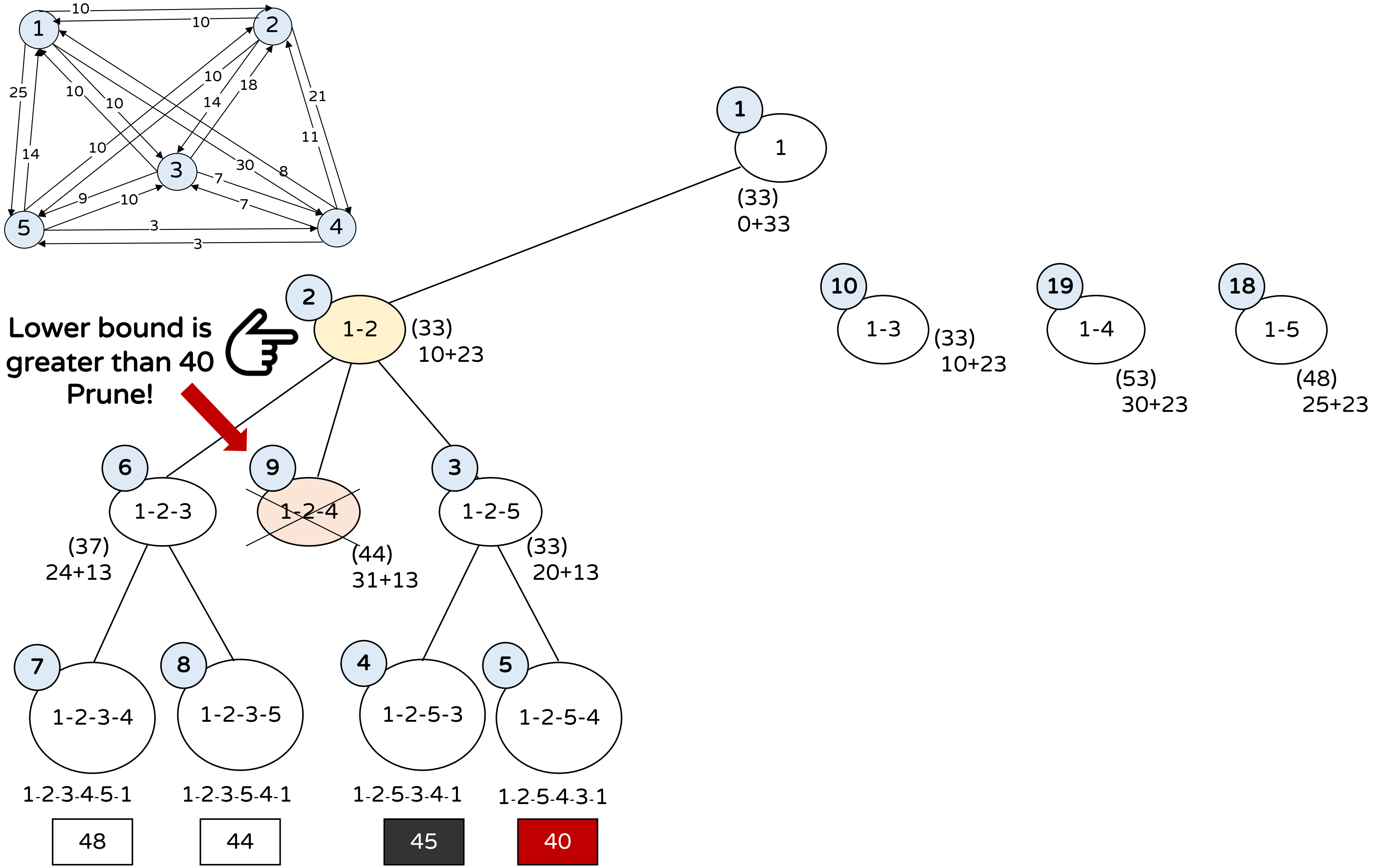
Lower bound is less than 40
Keep going!



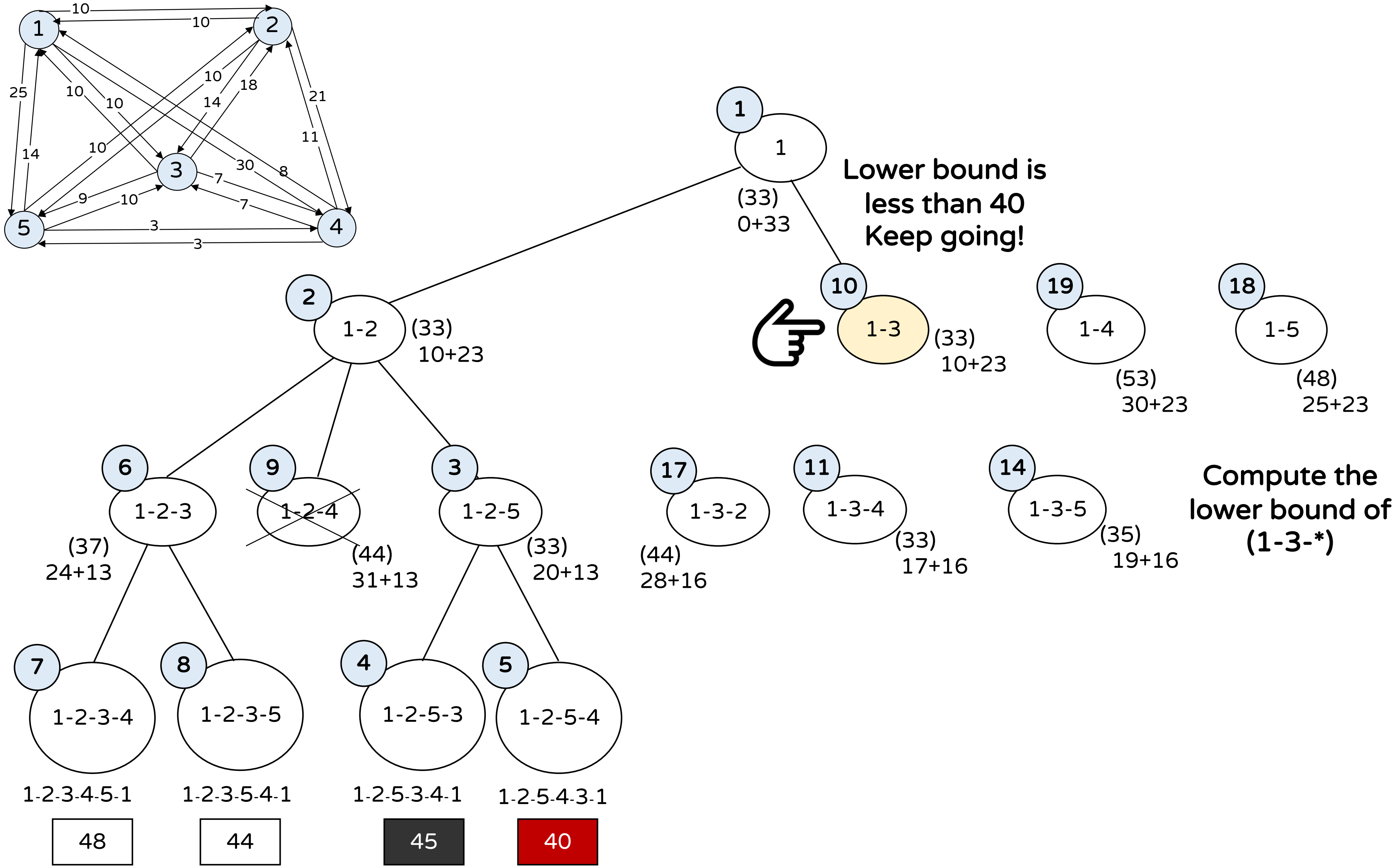
Example Of Bounded Branch



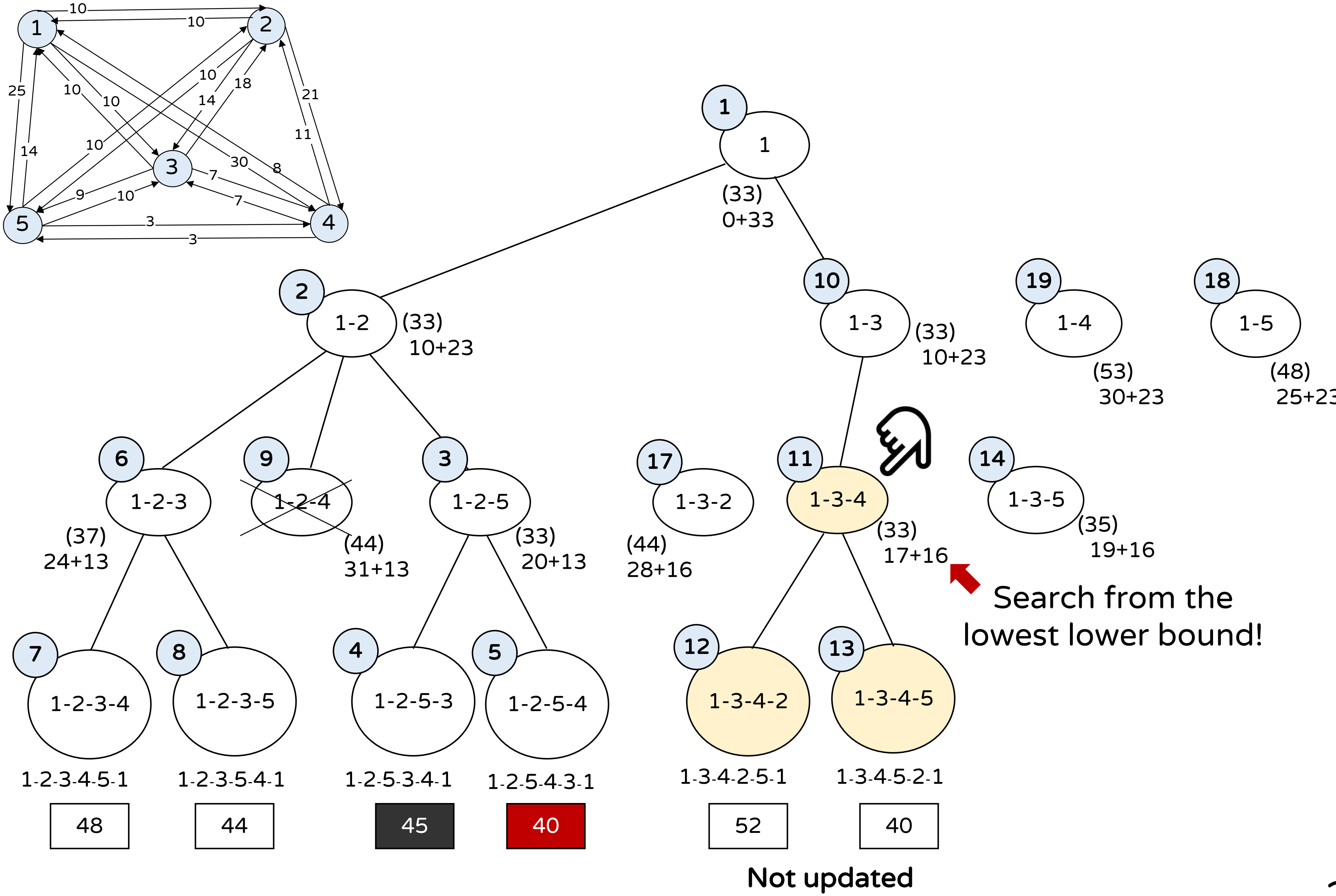
Example Of Bounded Branch



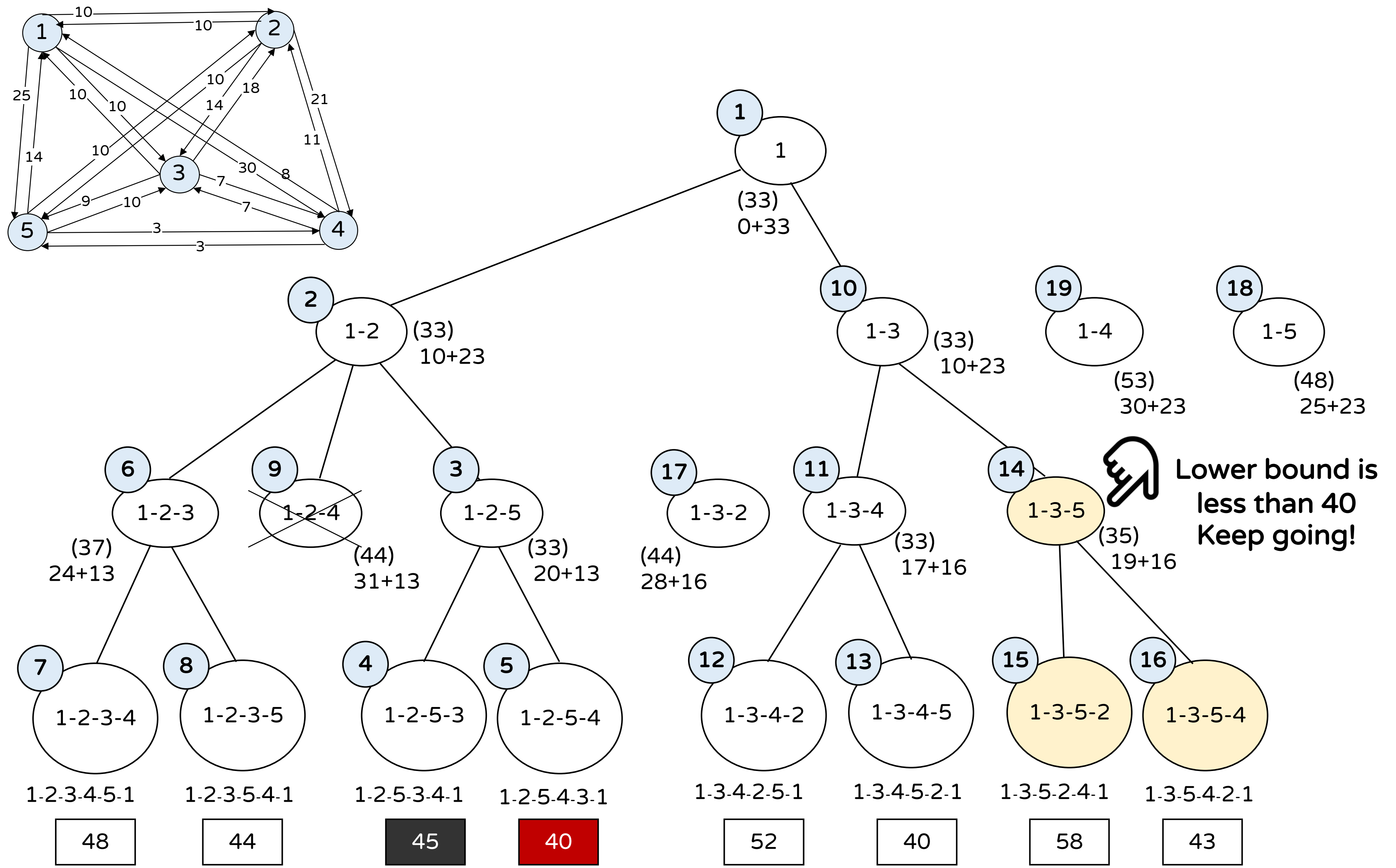
Example Of Bounded Branch



Example Of Bounded Branch

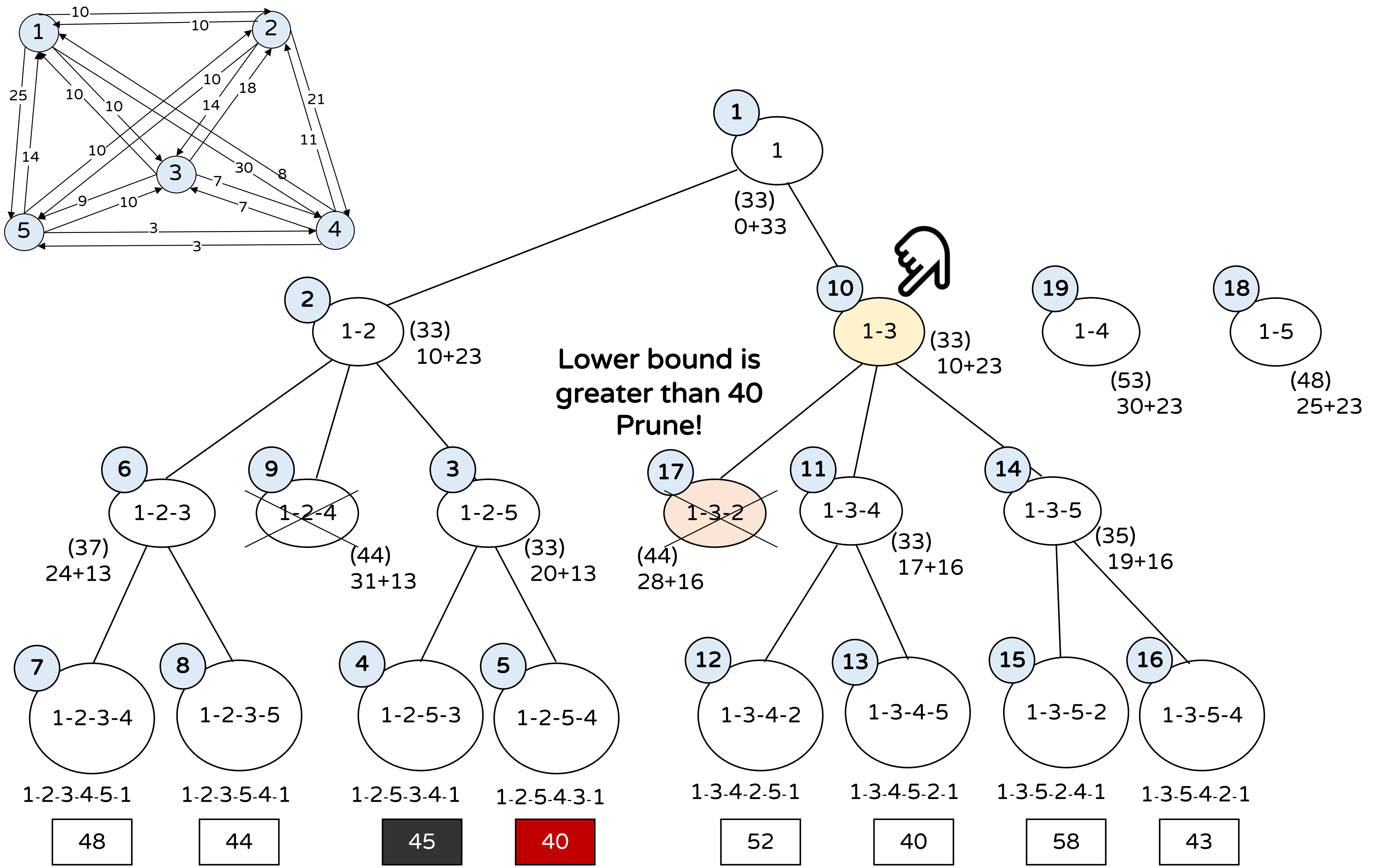


Example Of Bounded Branch

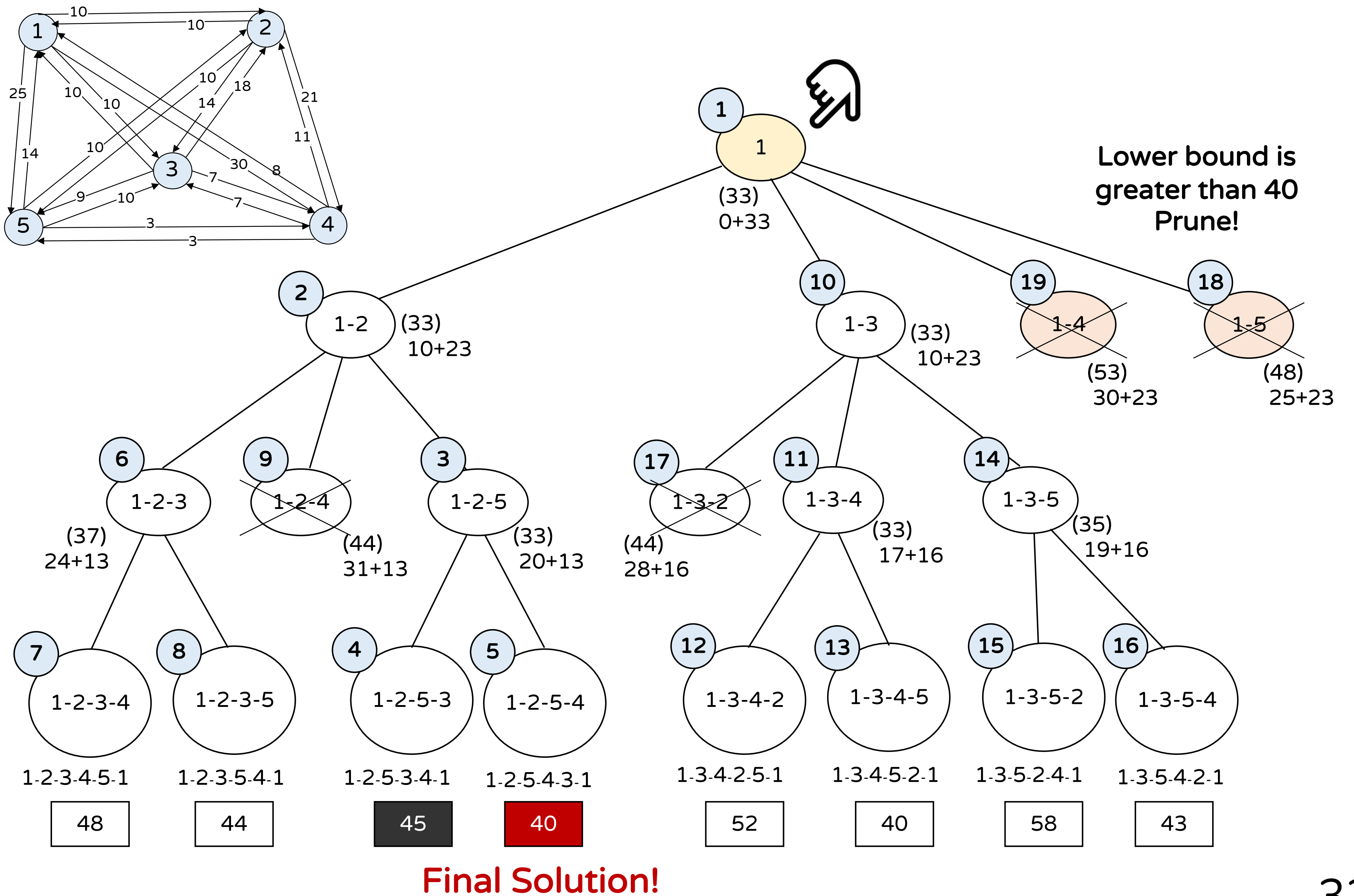


Not updated

Example Of Bounded Branch



Example Of Bounded Branch



Discussion

❑ Bounded branch v.s. brute-force

- Brute-force algorithm for TSP considers 24 cases
- Bounded branch algorithm considers 8 cases

❑ On base solutions

- In general, the quality of an initial base solution is poor
- The base solution monotonically decreases (or improves) during backtracking
  - Pruning is highly likely to occur at the latter part of backtracking
- Approximate solution is a good option for base solution

❑ Better way for measuring the lower bound

- Increase the probability of pruning by quickly finding a good base solution (it's specific to problems – research)

What You Need To Know

❑ State space search of an algorithm

- Searching for an answer in SST depicted by the algorithm

❑ Backtracking

- Performs DFS in the state space tree

❑ Bounded branch (backtracking + pruning)

- Prune unpromising states during backtracking to make the **branches** of SST **bounded**

❑ Pruning technique

- Prune a state based on a base solution found so far
- Problem specific

In Next Lecture

❑ A* Search Algorithm

- Searching technique on SST with bounded branch

Thank You