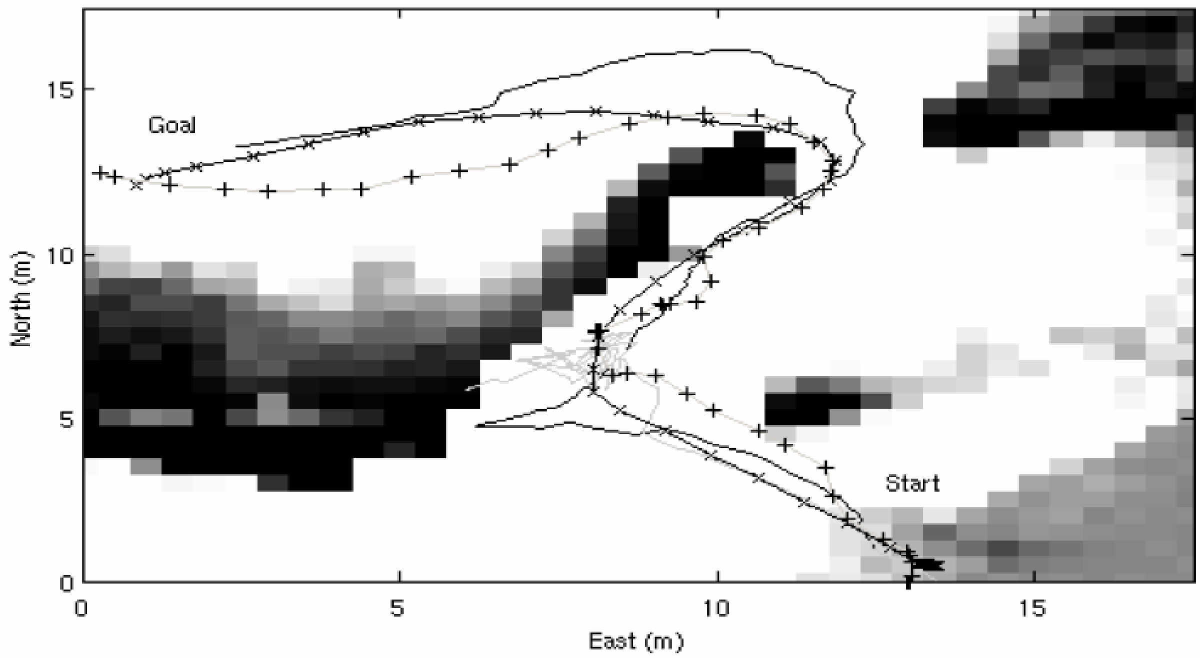
Artificial Intelligence: Graph Based Search Techniques

Presented by Michael Otte



These slides were prepared with help from:

"Artificial Intelligence, A Modern Approach"

by Stuart Russell and Peter Norvig

Graph Based Search

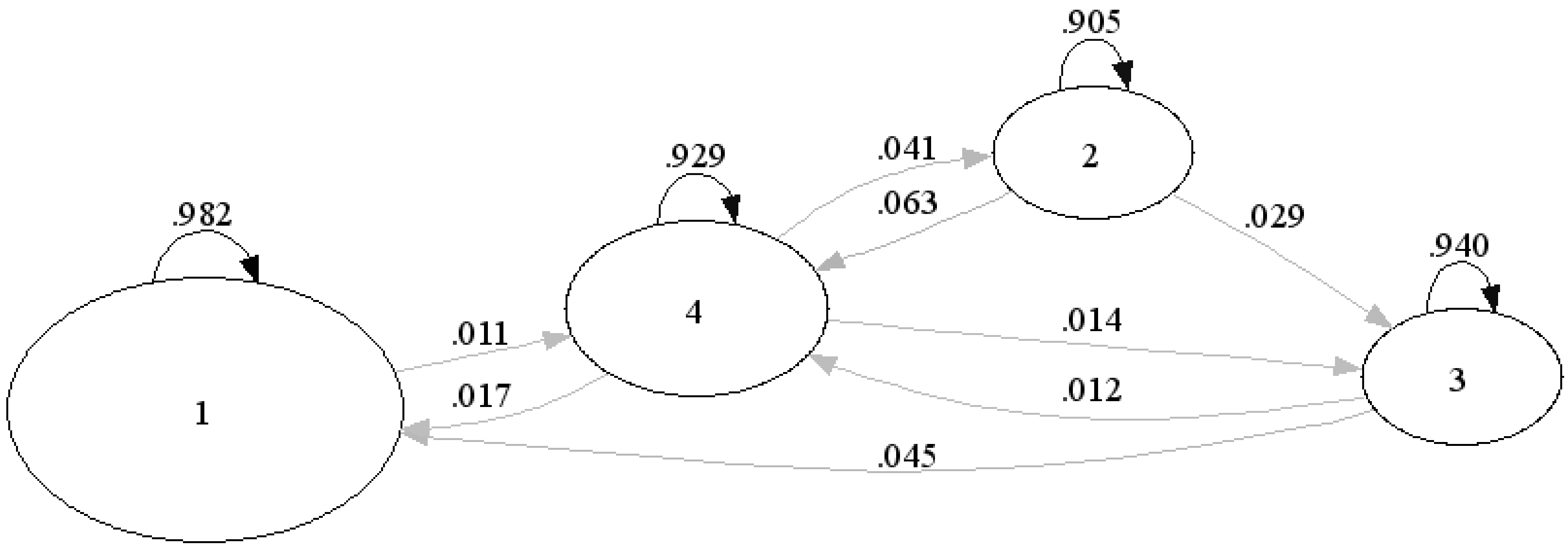
- What I am going to talk about in the next three classes:
  - Uninformed (blind) Search
  - Informed (heuristic based) Search
  - Current applications to artificial intelligence
    - Robotics
    - Computer Games
  - Part 3 of your project
    - Implement a search algorithm (A*) that will be used for navigation on the LAGR robot.

Graph Based Search

- What is a graph?
  - Nodes (states)
    - Initial State
    - Goal State
  - Arcs (the links between the states)
    - Directed
    - Undirected
  - Cost
  - Other stuff:
    - Successor Function
    - Goal Test

Graph Based Search

- A simple Graph:



- I am going to do more interesting example on the board (with corresponding solution tree).

Graph Based Search: Uninformed Search

- Solving a graph can be thought of as searching through a tree until a goal state is found.
- Any traditional tree search methods can be used (but some may work better than others)
  - Depth-First Search
  - Breadth-First Search
  - Depth-Limited Search
  - Iterative Deepening Depth-First Search
  - Bidirectional Search
  - Dijkstra's Algorithm

Uninformed Search: Depth-First Search

- Depth: The number of nodes along the path from the initial state.
- Depth-first search always expands the deepest node in the current *fringe* of the search tree.
  - Fringe: the set of nodes that the search has found
- It is possible to implement depth first search by storing the fringe in a stack.
- Remember to mark a node as visited when it is expanded.

Uninformed Search: Depth-First Search

- Depth first Requires O(*bm*) memory.
  - *b*: the (maximum) branching factor
  - *m*: the maximum depth of the tree
- A variant called *backtracking search* reduces memory requirements to O(*m*) by letting each node remember which of its children have been explored.
- Problem: what happens if the graph contains an infinite number of nodes?
- Depth-first search is not complete or optimal

Uninformed Search: Breadth-First Search

- Breadth-first search expands nodes in the order of their depth.
  - That is, every node of depth *n* is expanded before any node of depth *n*+1 is expanded.
  - This can be implemented in a first-in-first-out queue
- Memory and time: $O(b^{d+1})$
  - b: branching factor
  - d: depth of goal
- Breadth-first search is complete, it is optimal if all steps have the same cost.

Uninformed Search: Depth-limited and Iterative Deepening

- Depth-limited search is (usually depth-first) search with a constraint on the length of a path.
  - Choose the constraint carefully or risk failure
- Iterative Deepening expands this idea:
  - Perform depth-limited search with depth 1,2,3...
  - Memory (depth-first based): O(bd)
  - Time: O(b^d)
    - This is faster than breadth-first search! Why?
    - Because most nodes exist on the bottom level of the tree and breadth-first search inserts an extra layer of nodes into the fringe.

Uninformed Search: Dijkstra's Shortest Path Algorithm

- Used when arcs have arbitrary non-negative costs.
- Basic idea: expand the shortest path first and store the distance required to reach each node at that node, updating as necessary.
  - runtime (if implemented carefully with a Fibonacci heap) is $O(E + V \log V)$
    - E: total edges in the graph
    - V: total vertices in the graph
- Optimal and Complete

Graph Based Search: Informed Search

- Sometimes we can use additional information about the world to help guide our search.
- The idea is that we can impart a little bit of 'common sense' about what is good or bad to the system using a *heuristic function.*
- Algorithms:
  - Hill-climbing search
  - Genetic Algorithms
  - Greedy best-first search
  - A* Search (You will have to implement this one)

Informed Search (Local Search) Hill-Climbing Search

- Suppose that we are told to find the best possible state in a space with respect to a particular quantity.
  - But, the space is huge and we can only make a few measurements a time.
  - Picture a mouse that is trying to find the top of a roller-coaster track.
- Example on board...

Informed Search (Local Search) Hill-Climbing Search

- Start at a random location and go up
  - If space is convex, then this will always find a global solution, if not it will find a local solution.
  - Steepest ascent: choose the highest neighbour
  - Stochastic Hill Climbing: choose randomly from uphill moves (this makes more sense in > 2D)
  - Random Restart: try a bunch of different things.
    - Now we have a pack of mice instead of just one.
  - Local Beam Search: try a bunch of different things, and allow them to communicate.
    - If one of the mice finds a good state, it tells its friends about it.
  - Simulated Annealing: temperature and probability
    - So much for the mouse metaphor...

Informed Search (Local Search) Genetic Algorithms

- With some probability:
  - Independently successful features of individuals are randomly combined to form a new individual.
  - A new individual is created by mutating an old individual.
  - A cool example:

    http://www.rennard.org/alife/english/gavgb.html

- Back to global path finding...

Graph Based Search: Uninformed Search Revisited

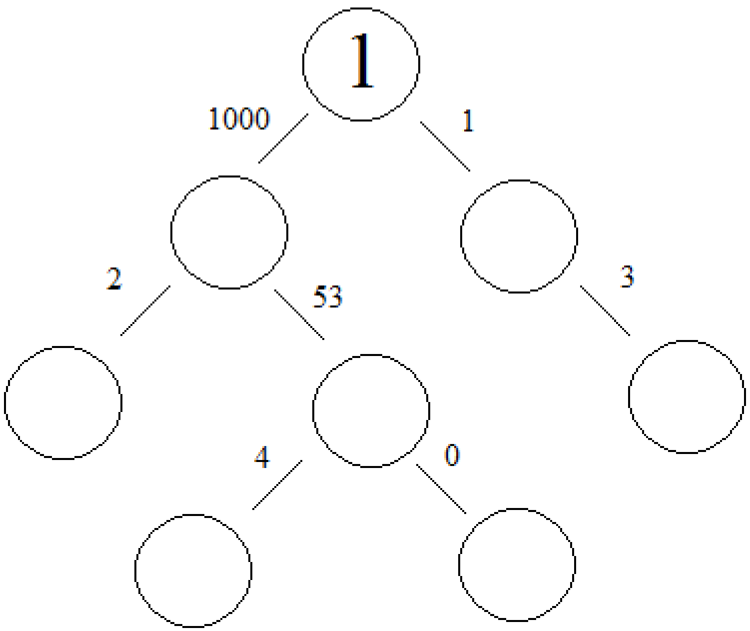
- On the Quiz from last time:
  - Depth First Search: 75%
  - Breadth First Search: 75%
  - Dijkstra's: 25%

- ... I think that I'll work through a few more examples of these things before we move on.

Uninformed Search Revisited: Depth First Search

- Another way of thinking about it:
  - Recursively expand Nodes based on the order that you discover them in.
- That means
  - Do not return back to the parent of a node before all of the children of the current node have been explored.
- Memory Requirement Confusion from last time:
  - O(db) if we store the unexpanded nodes
    - d=depth
    - b=branching factor
  - O(d) if we let each node remember which children have been expanded.

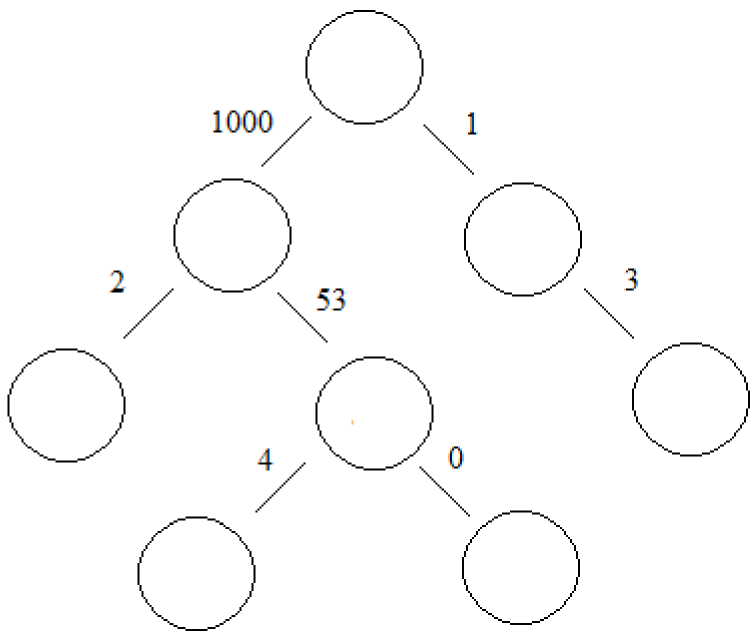
Uninformed Search Revisited: Depth First Search



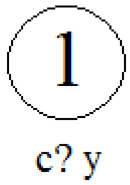
Suppose our successor function breaks ties by expanding the left most node

Uninformed Search Revisited: Depth First Search

The whole graph

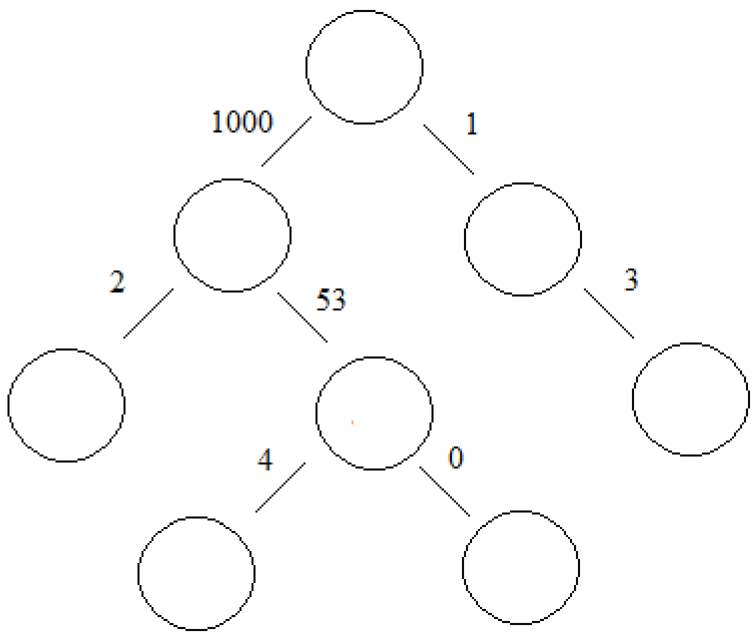


What we know

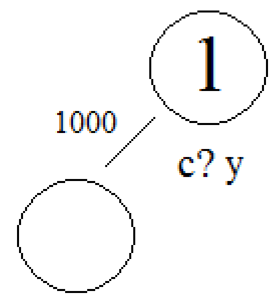


Uninformed Search Revisited: Depth First Search

The whole graph

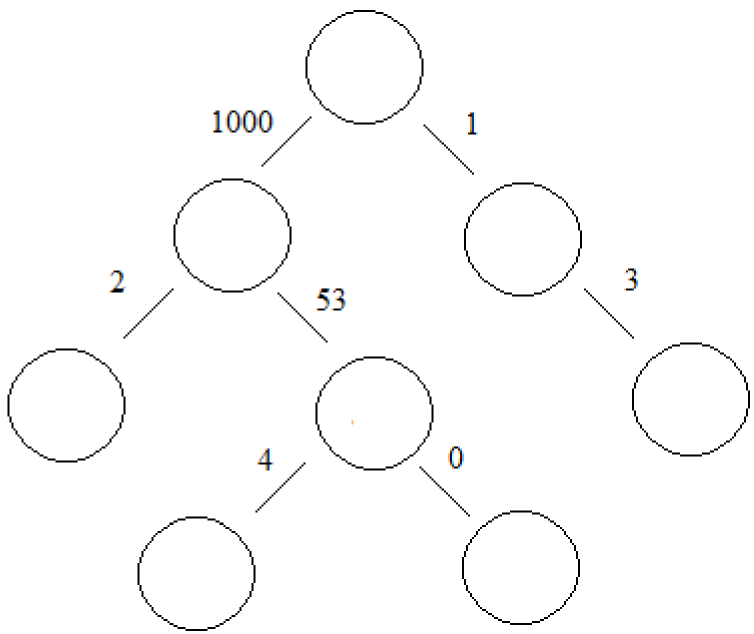


What we know

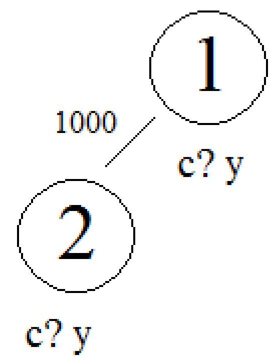


Uninformed Search Revisited: Depth First Search

The whole graph

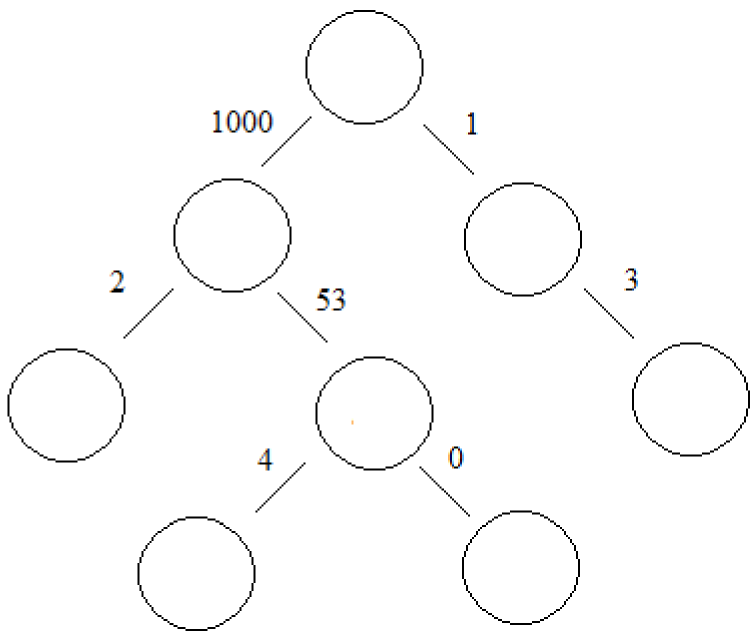


What we know

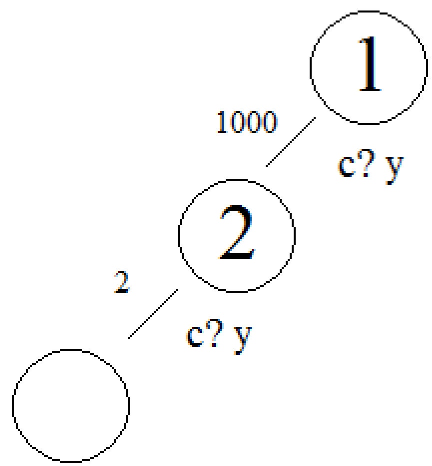


Uninformed Search Revisited: Depth First Search

The whole graph

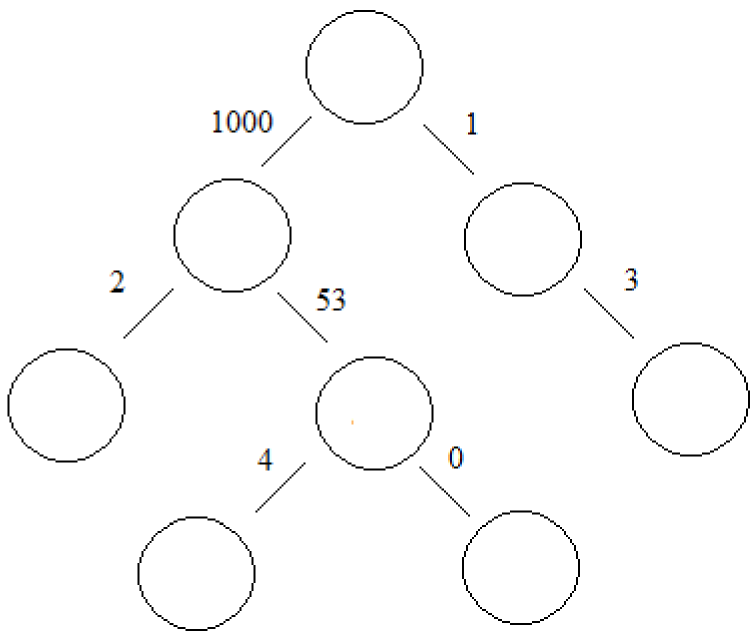


What we know

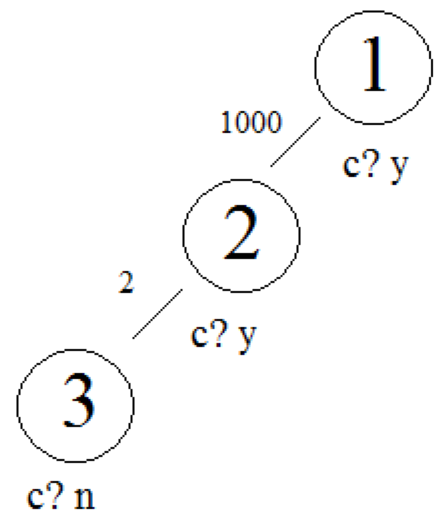


Uninformed Search Revisited: Depth First Search

The whole graph

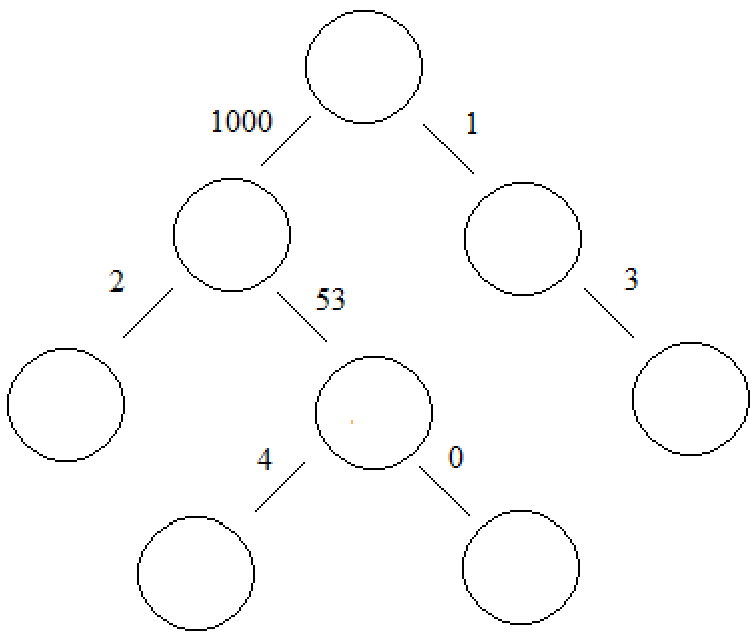


What we know

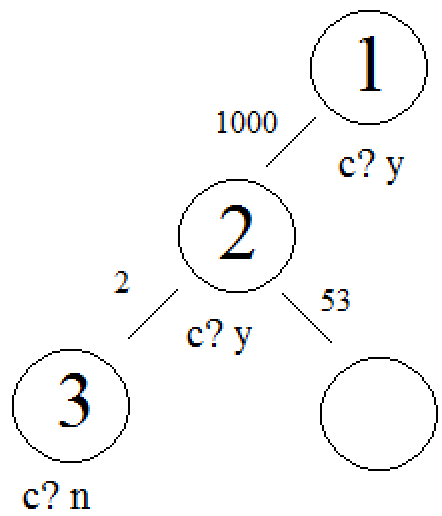


Uninformed Search Revisited: Depth First Search

The whole graph

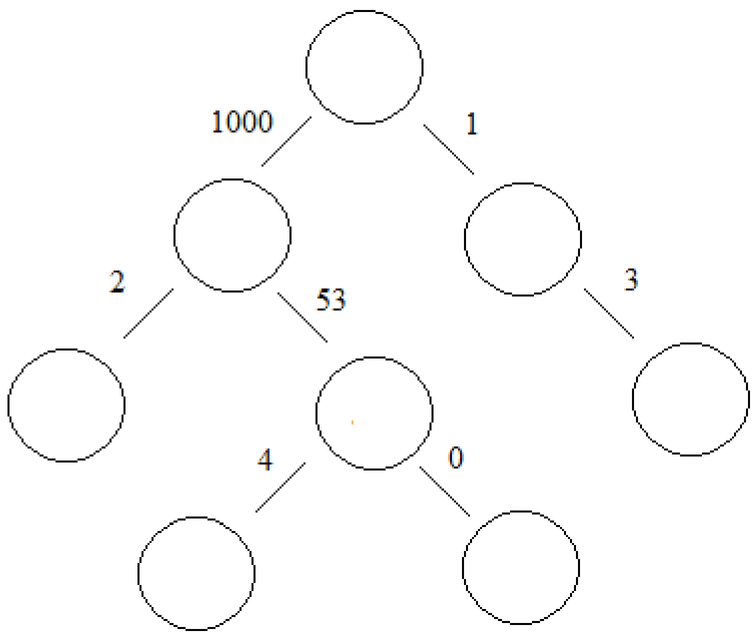


What we know

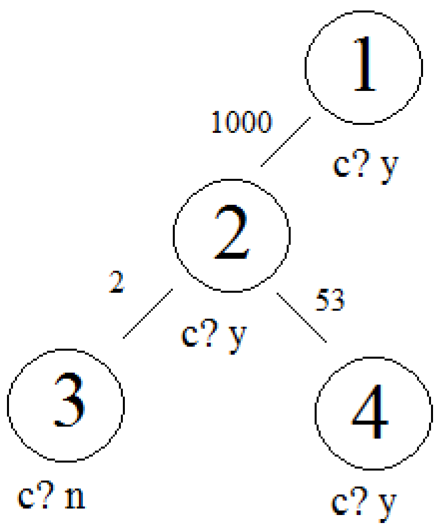


Uninformed Search Revisited: Depth First Search

The whole graph

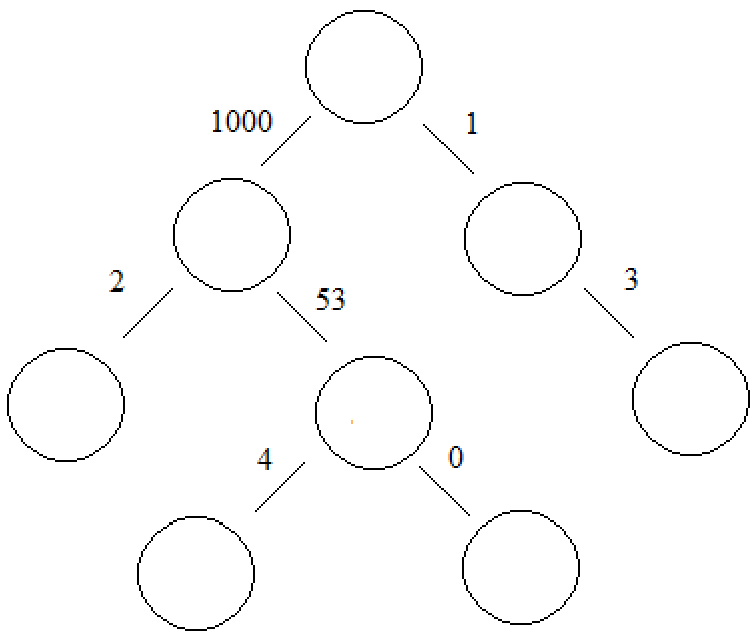


What we know

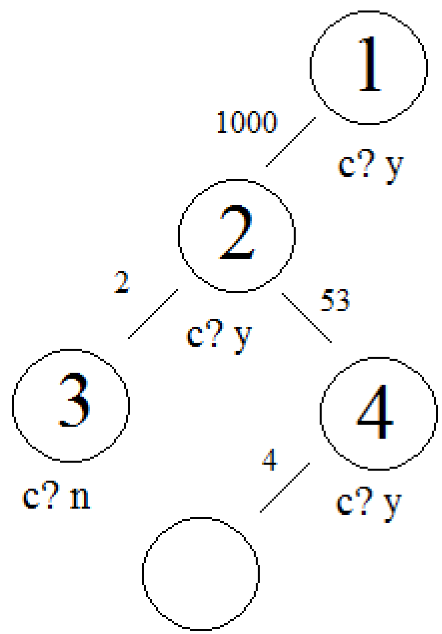


Uninformed Search Revisited: Depth First Search

The whole graph

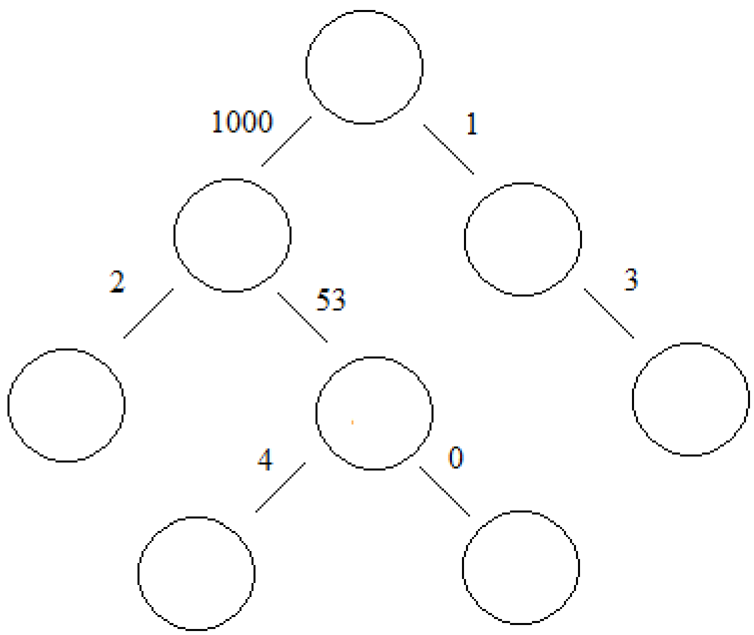


What we know

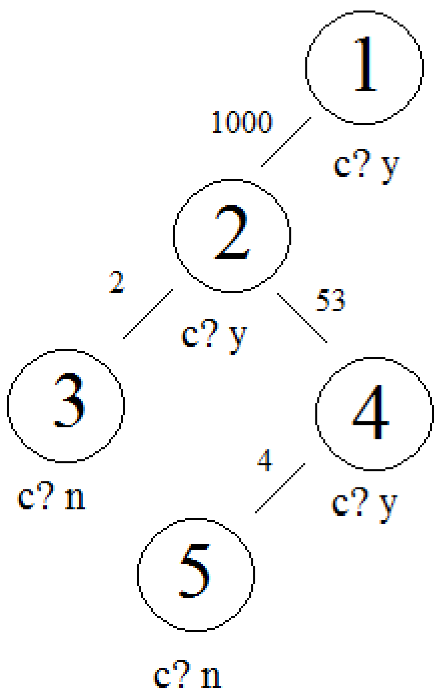


Uninformed Search Revisited: Depth First Search

The whole graph

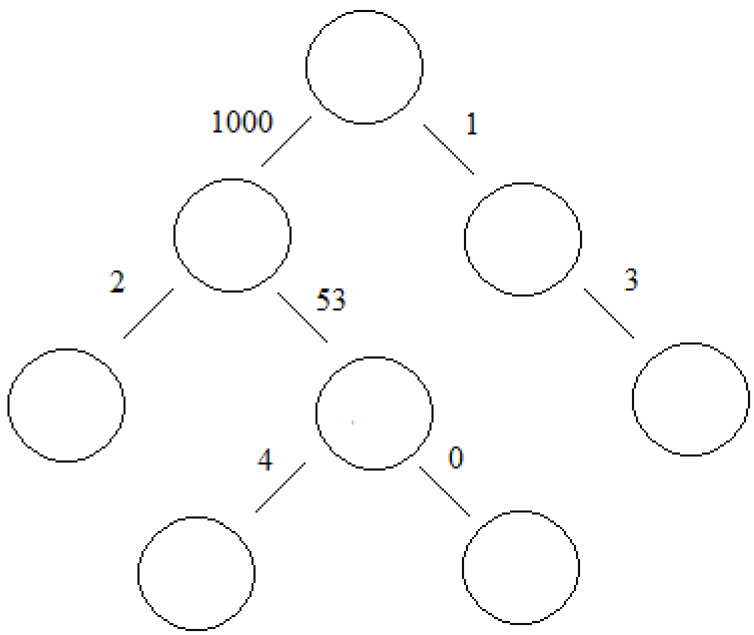


What we know

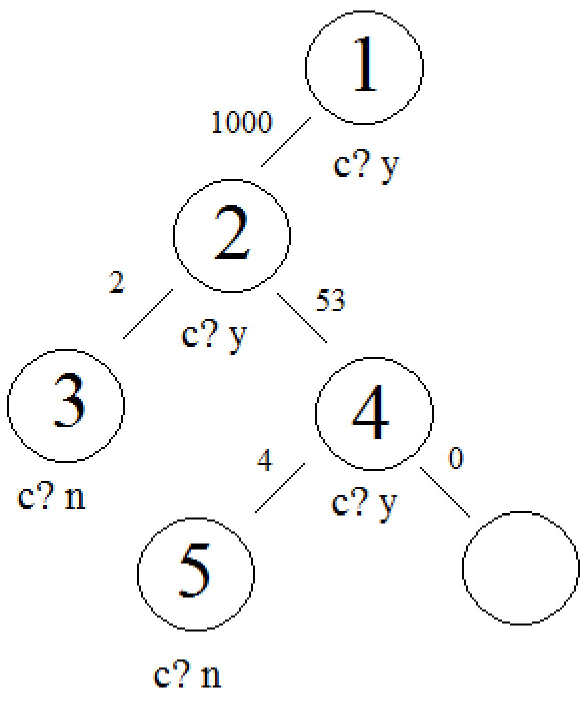


Uninformed Search Revisited: Depth First Search

The whole graph

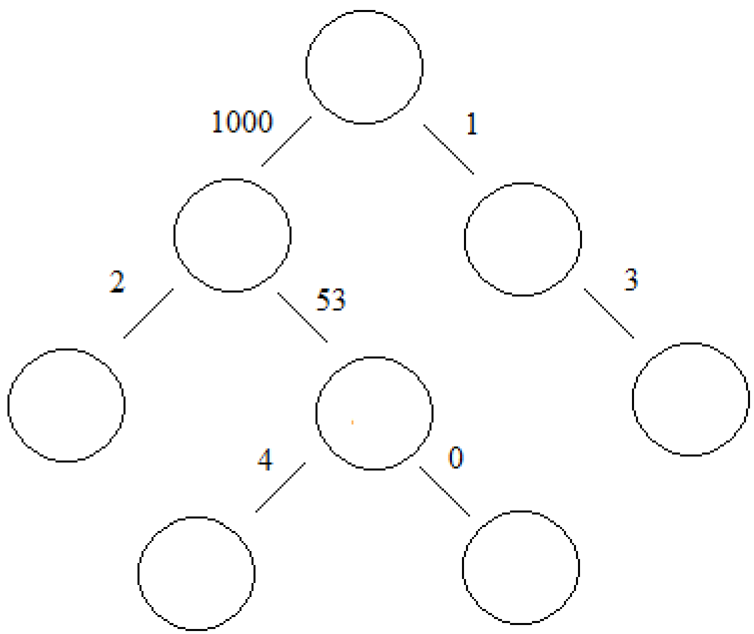


What we know

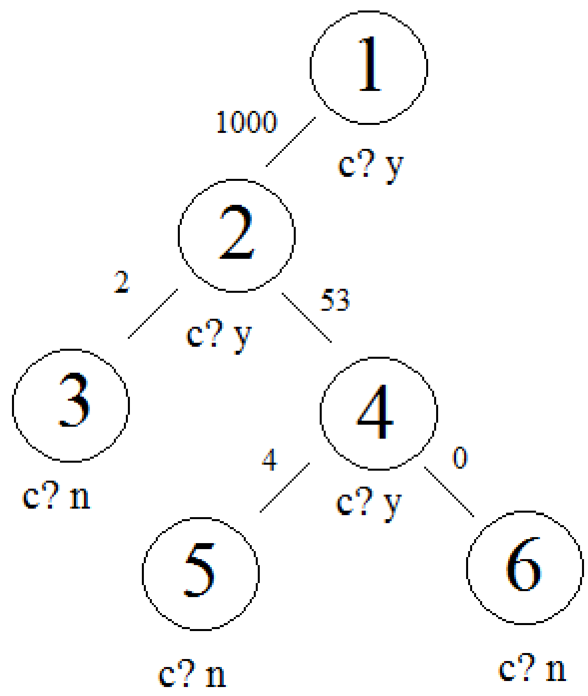


Uninformed Search Revisited: Depth First Search

The whole graph

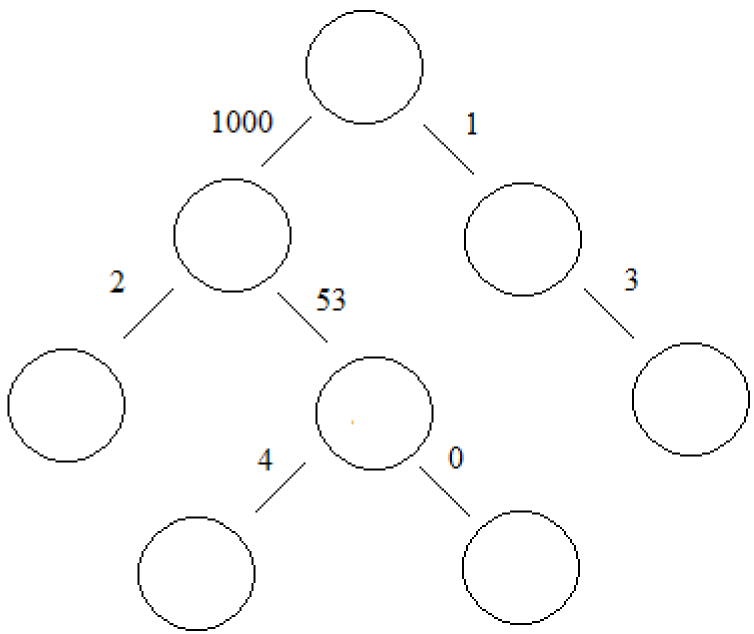


What we know

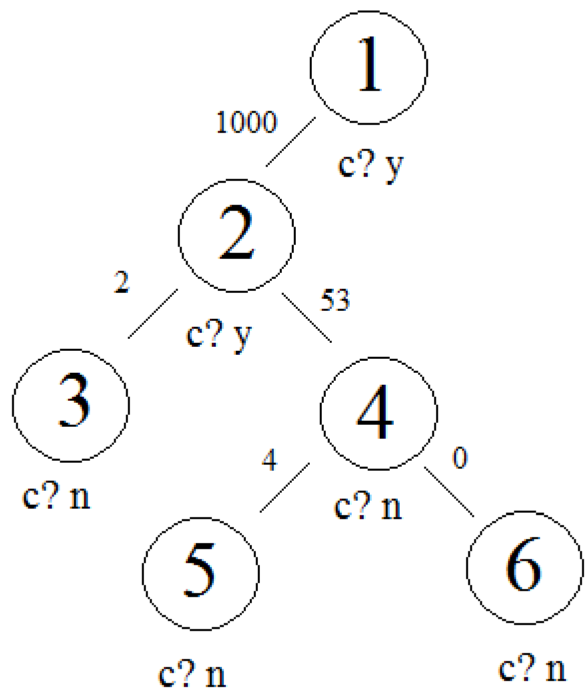


Uninformed Search Revisited: Depth First Search

The whole graph

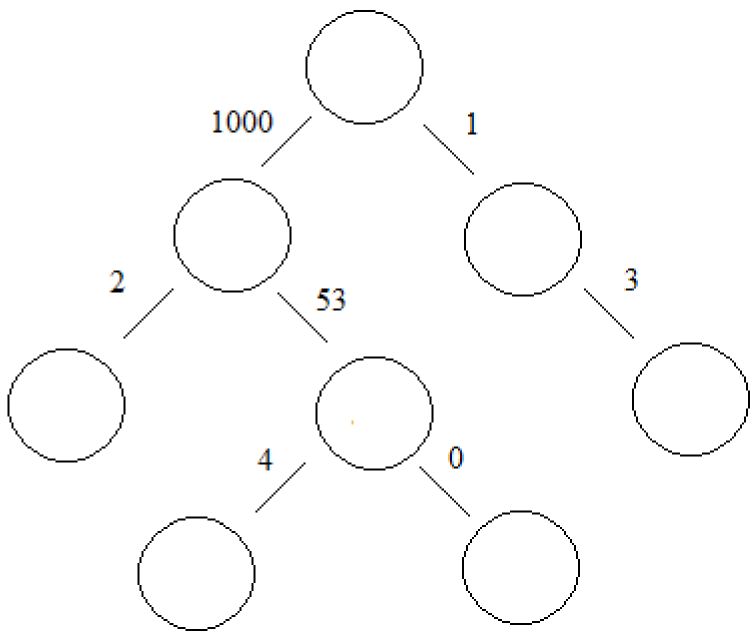


What we know

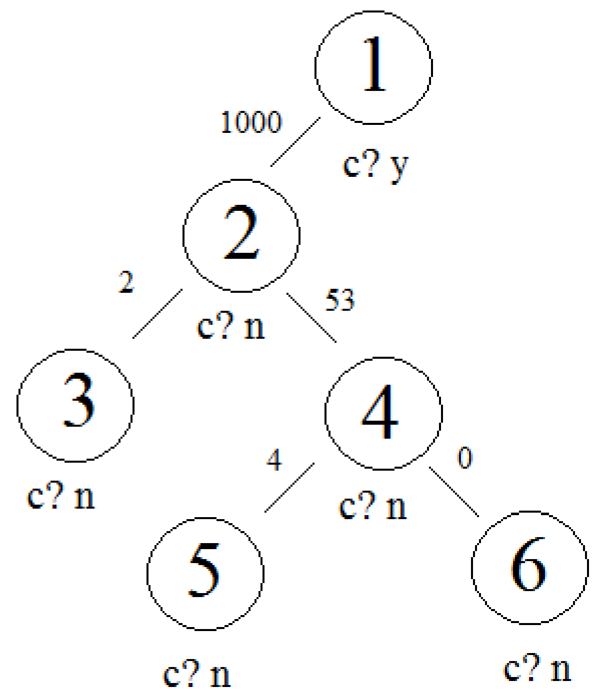


Uninformed Search Revisited: Depth First Search

The whole graph

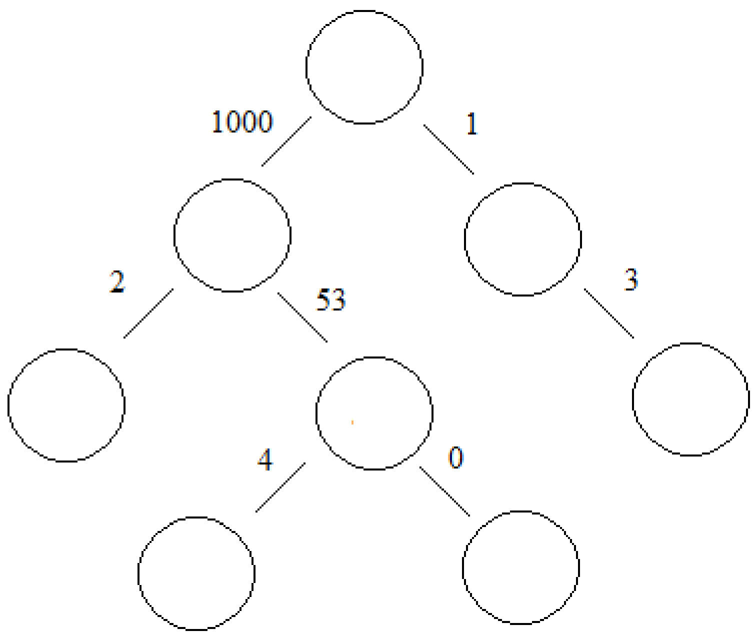


What we know

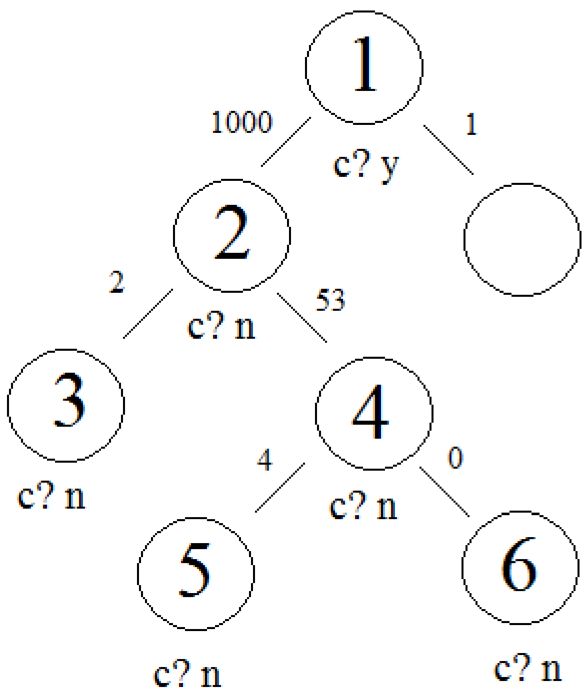


Uninformed Search Revisited: Depth First Search

The whole graph

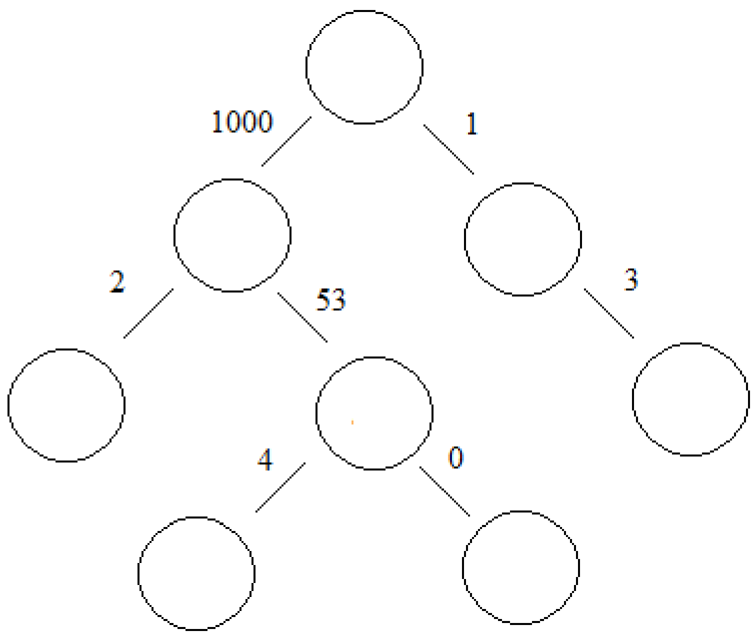


What we know

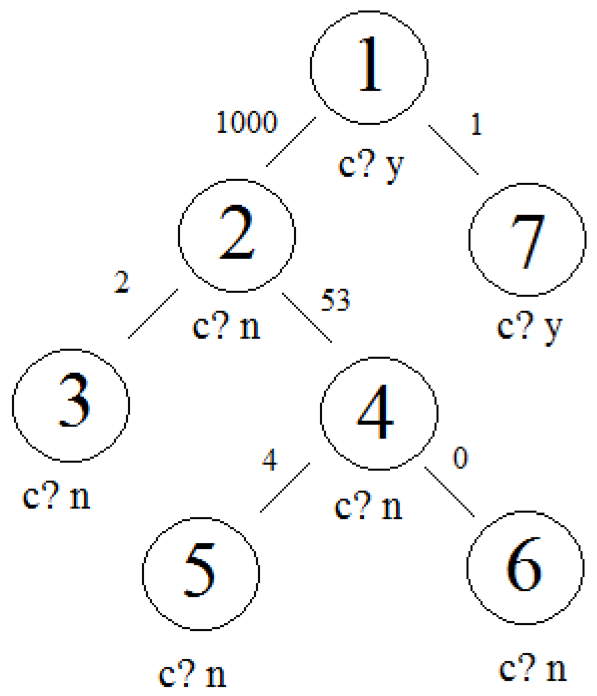


Uninformed Search Revisited: Depth First Search

The whole graph

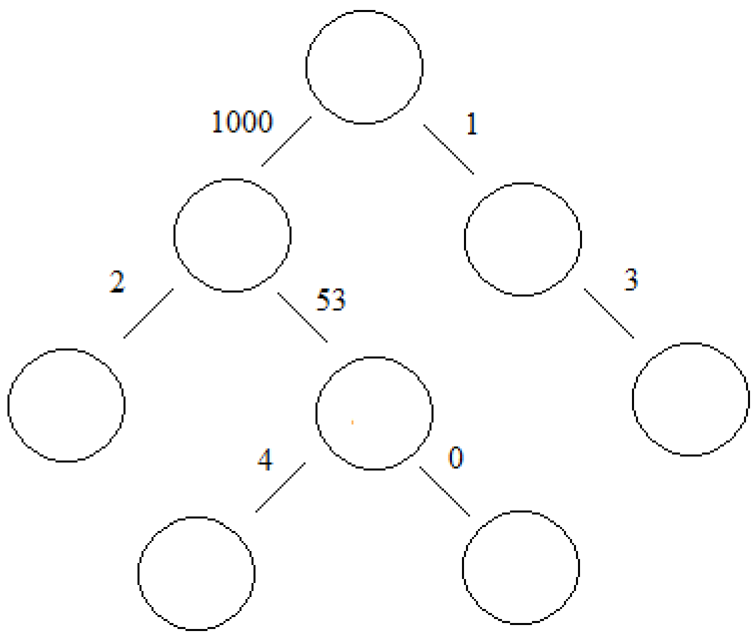


What we know

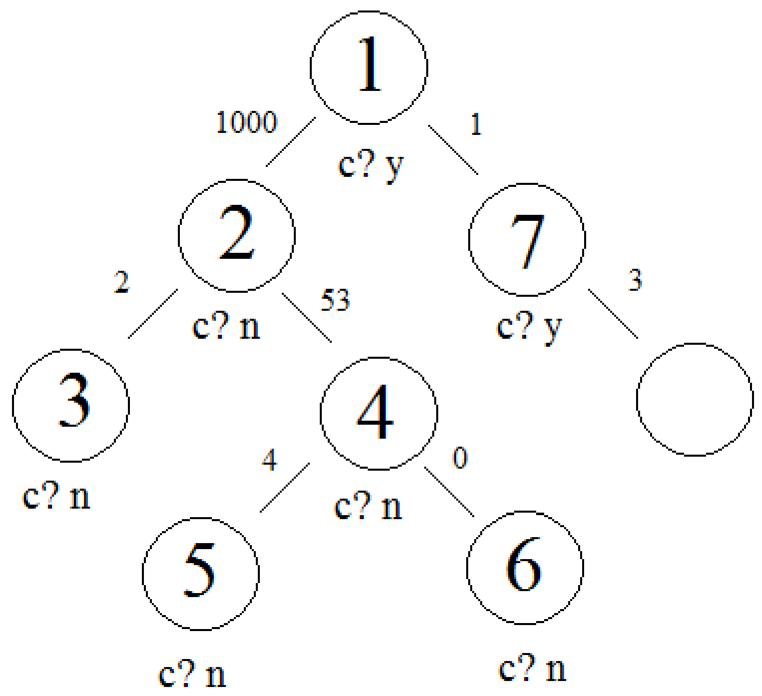


Uninformed Search Revisited: Depth First Search

The whole graph

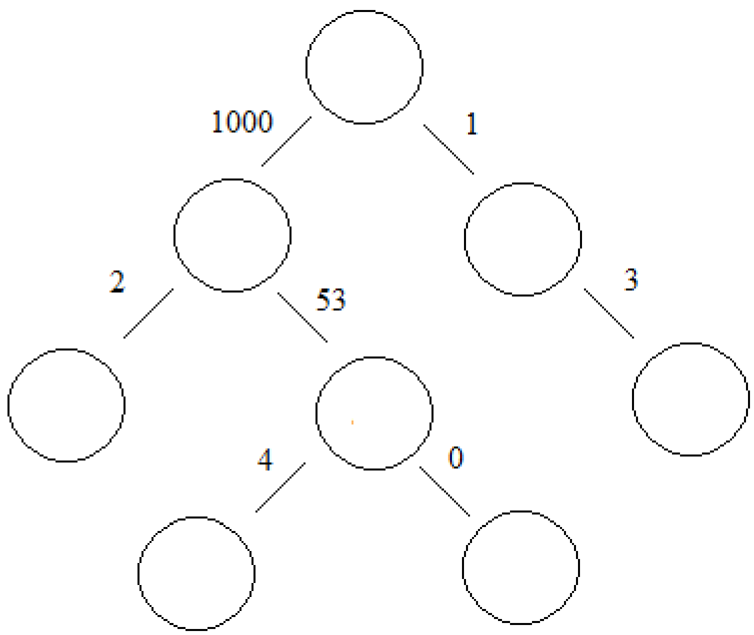


What we know

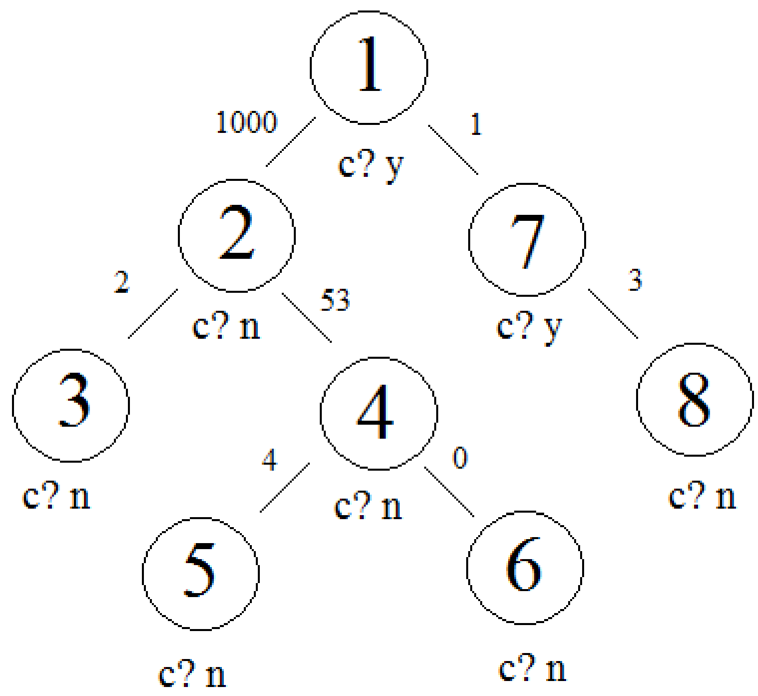


Uninformed Search Revisited: Depth First Search

The whole graph

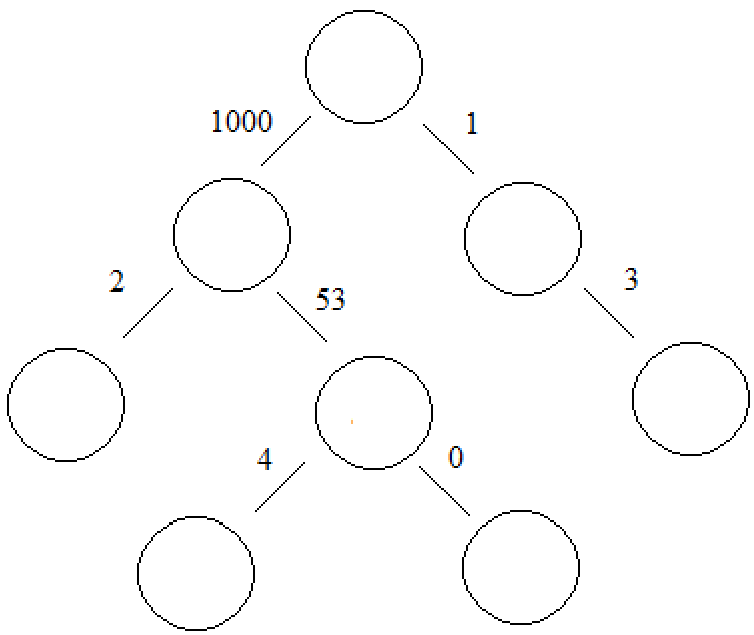


What we know

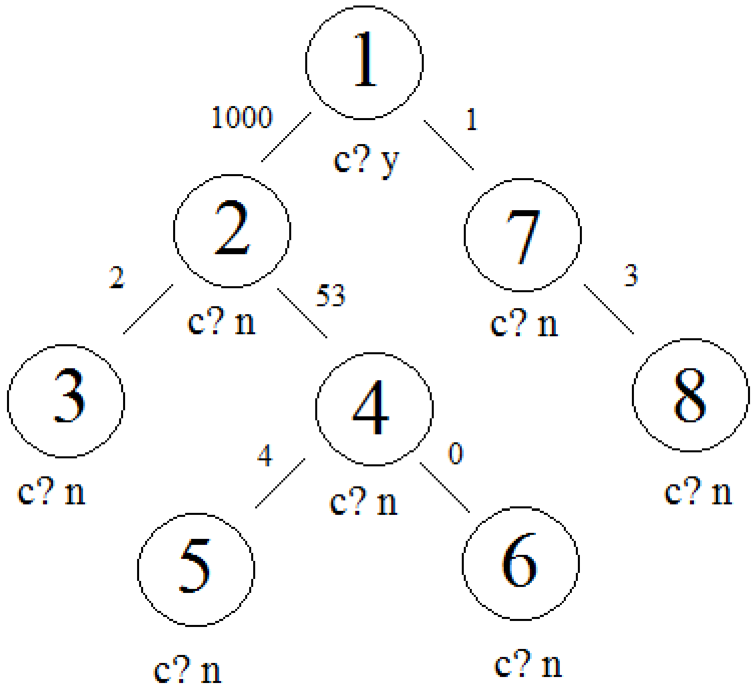


Uninformed Search Revisited: Depth First Search

The whole graph

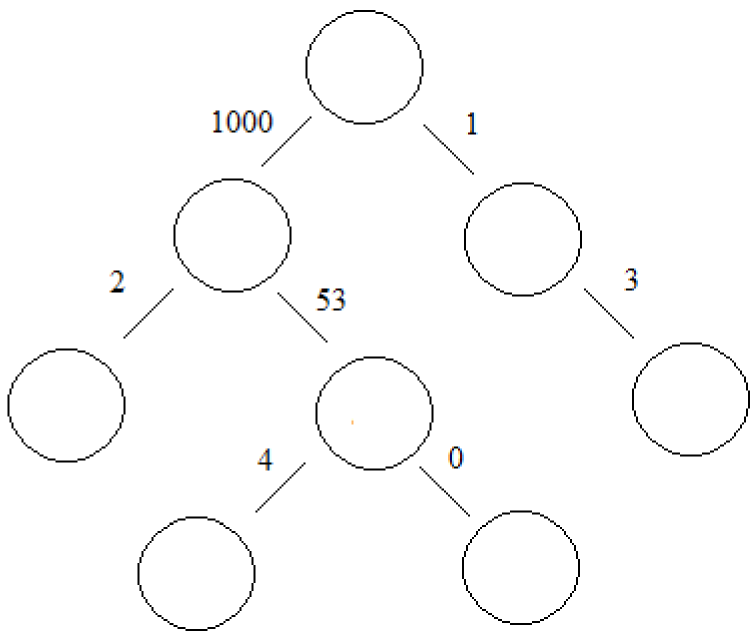


What we know

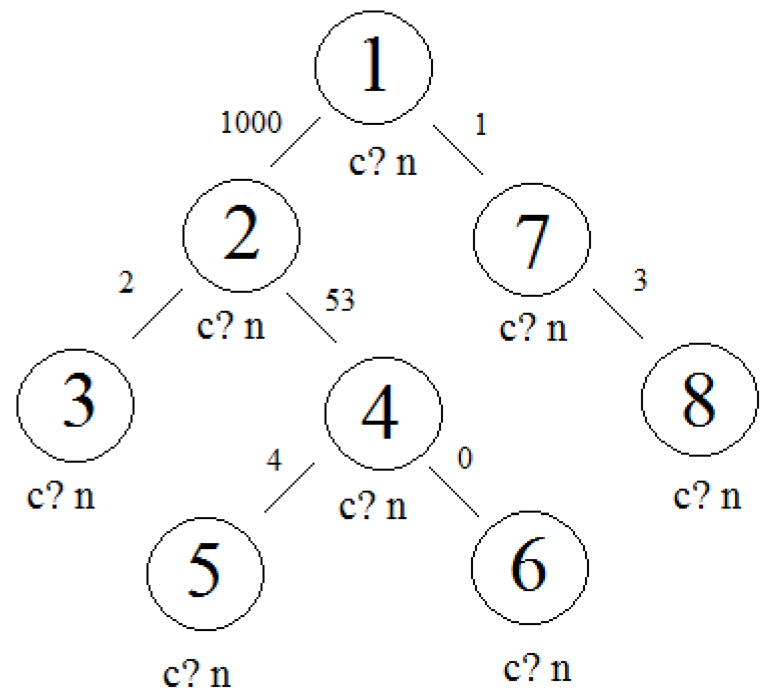


Uninformed Search Revisited: Depth First Search

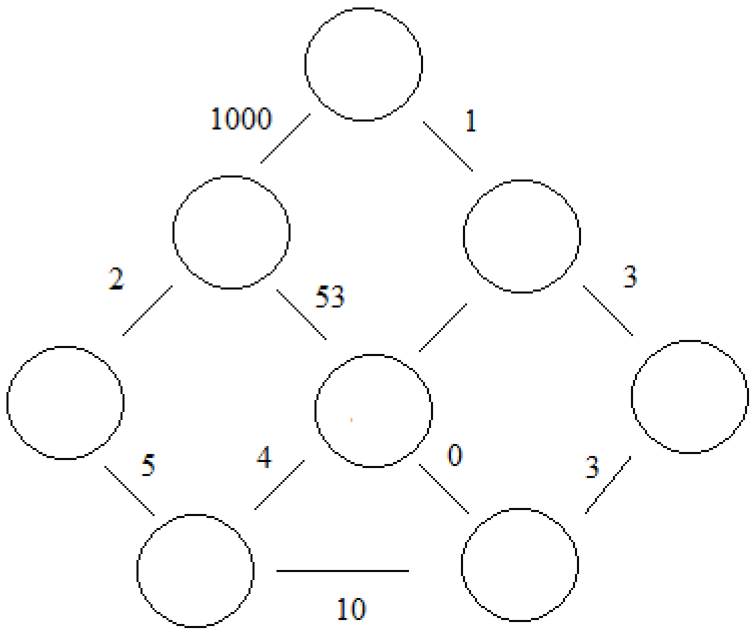
The whole graph



What we know



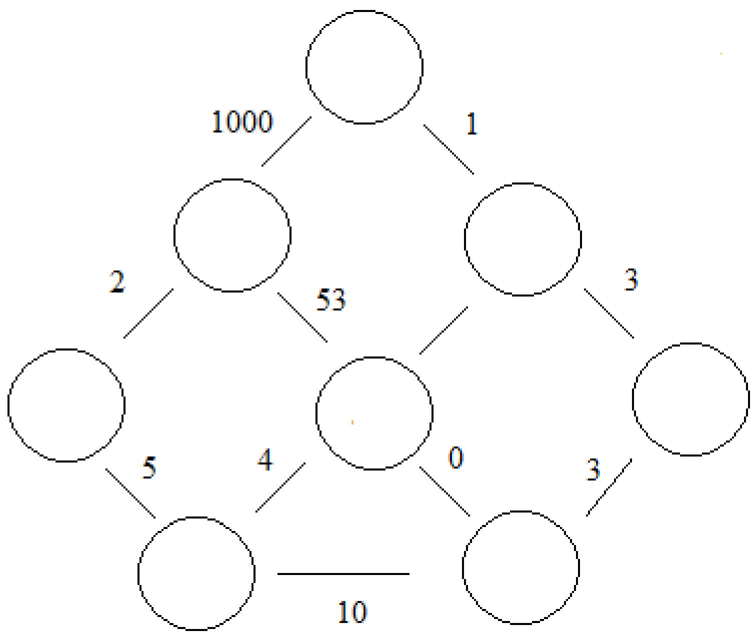
Uninformed Search Revisited: Depth First Search



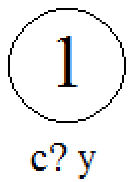
Suppose our successor function breaks ties by expanding nodes counter clockwise, starting at 12:00.

Uninformed Search Revisited: Depth First Search

The whole graph

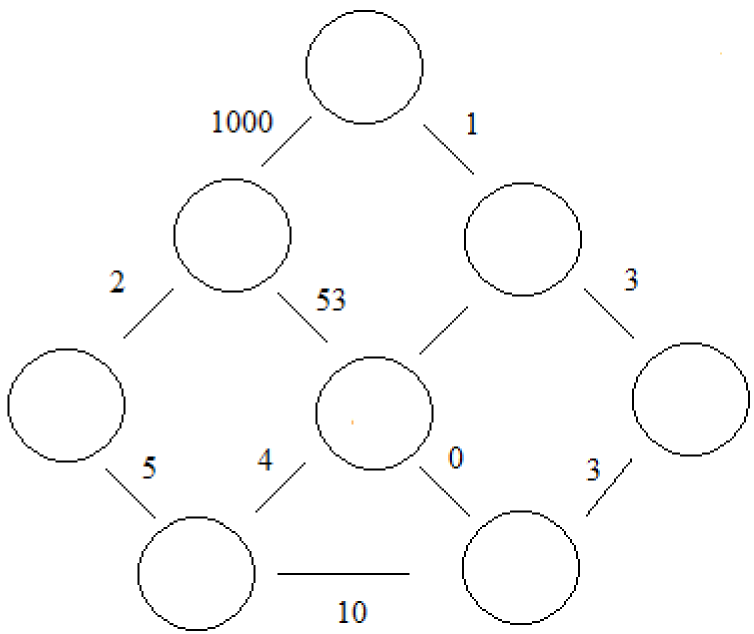


What we know

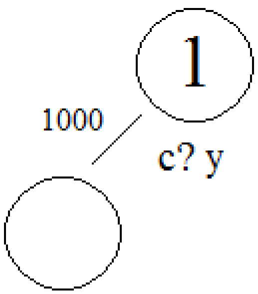


Uninformed Search Revisited: Depth First Search

The whole graph

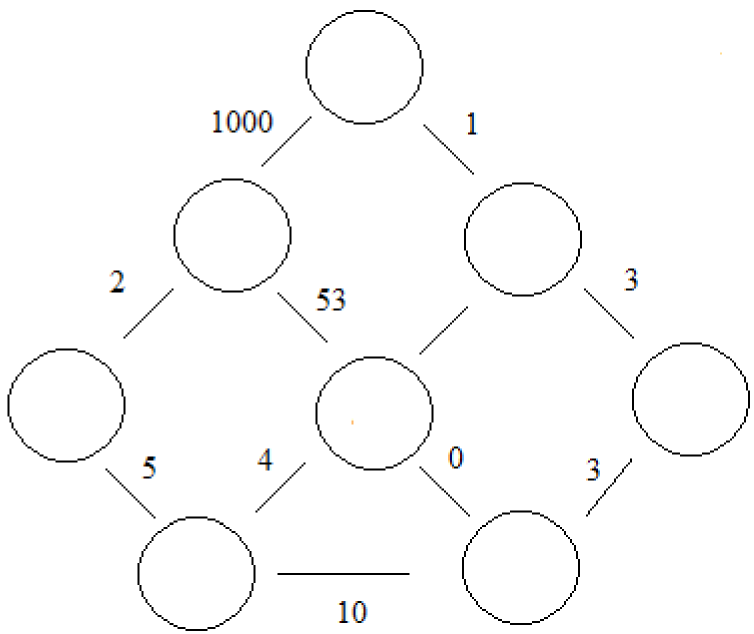


What we know

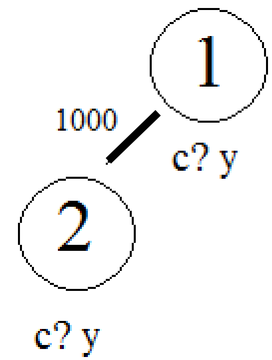


Uninformed Search Revisited: Depth First Search

The whole graph

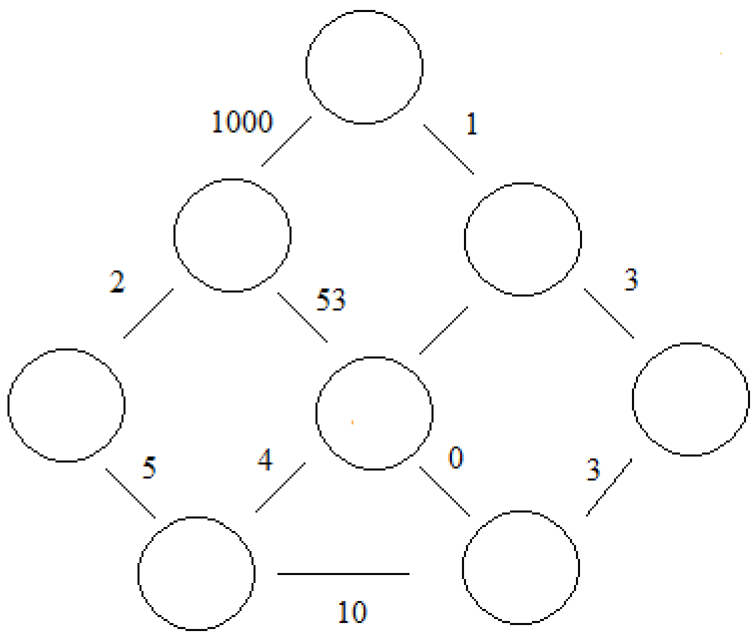


What we know

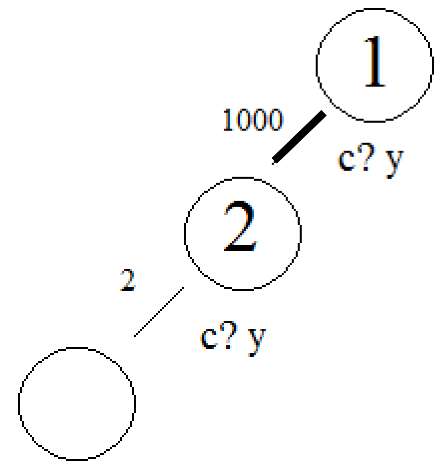


Uninformed Search Revisited: Depth First Search

The whole graph

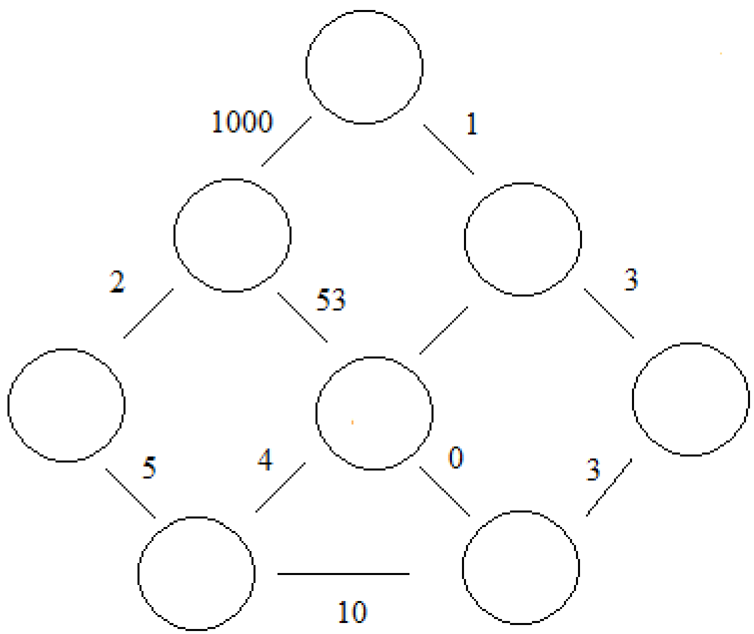


What we know

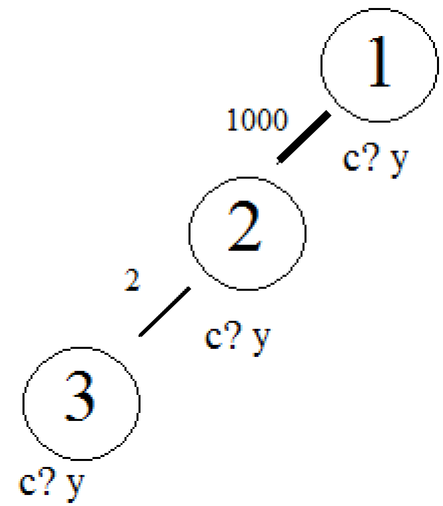


Uninformed Search Revisited: Depth First Search

The whole graph

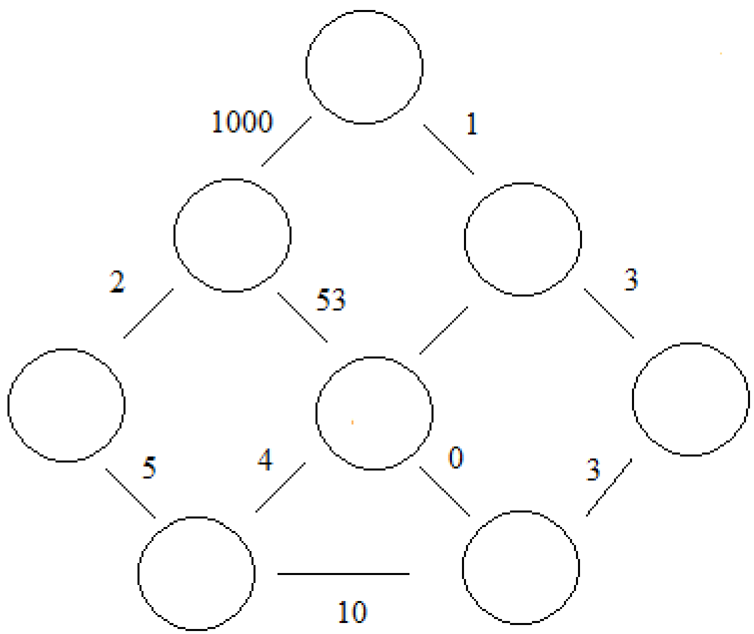


What we know

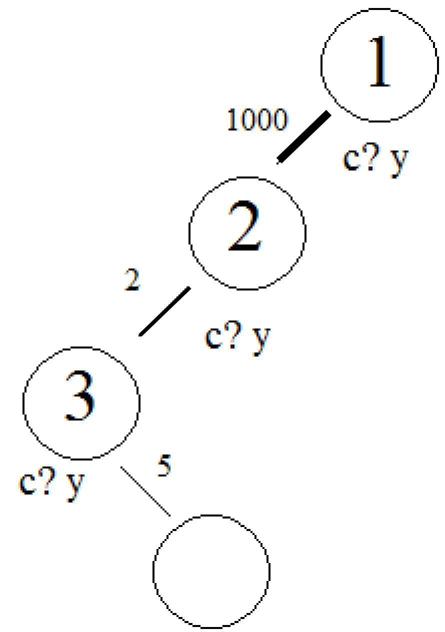


Uninformed Search Revisited: Depth First Search

The whole graph

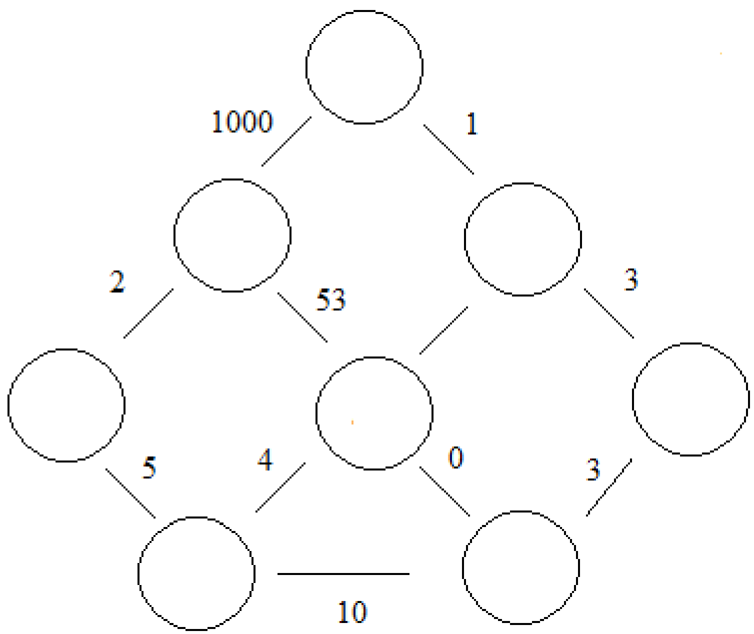


What we know

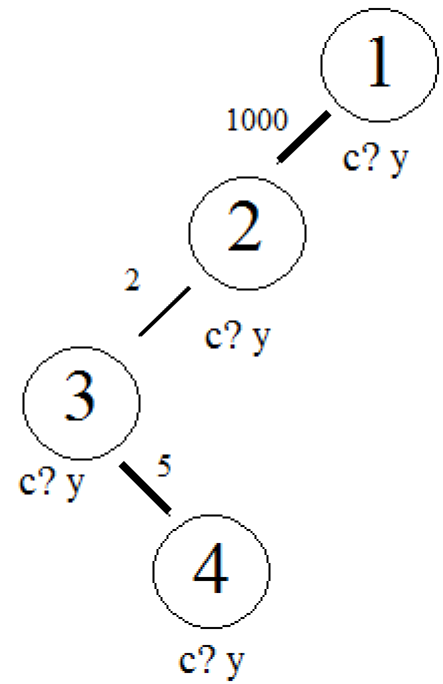


Uninformed Search Revisited: Depth First Search

The whole graph

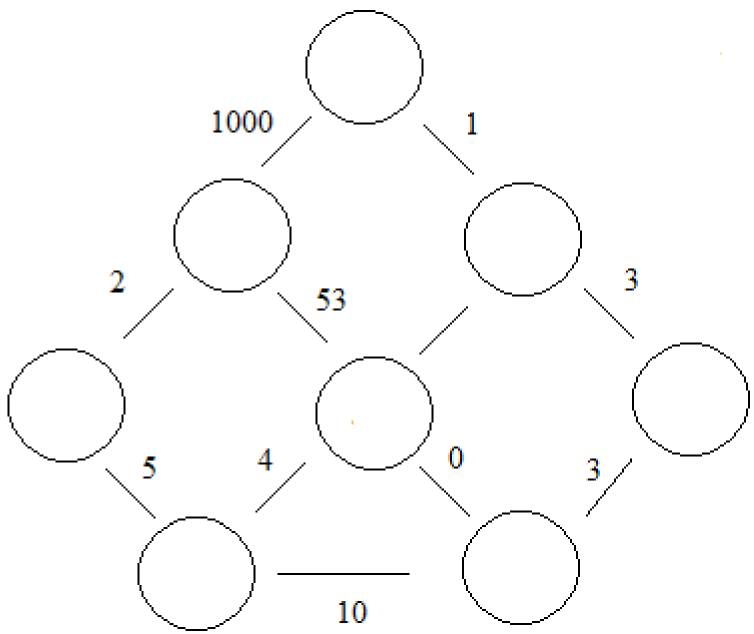


What we know

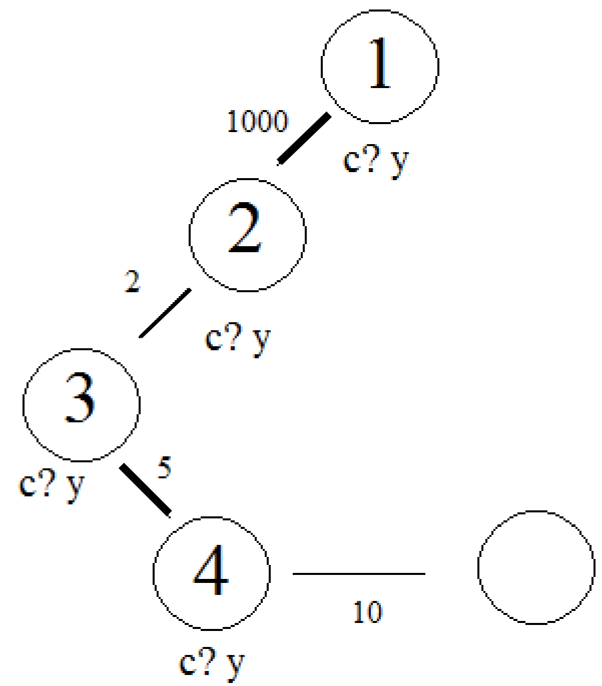


Uninformed Search Revisited: Depth First Search

The whole graph

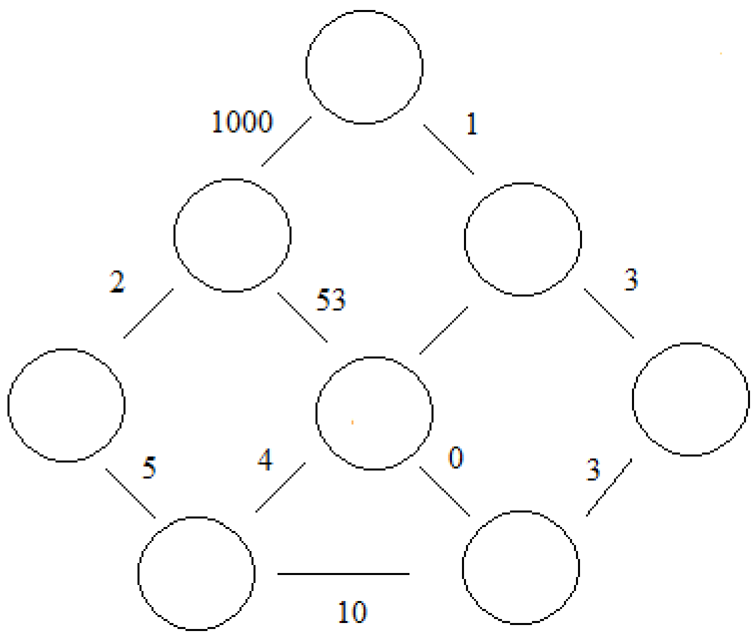


What we know

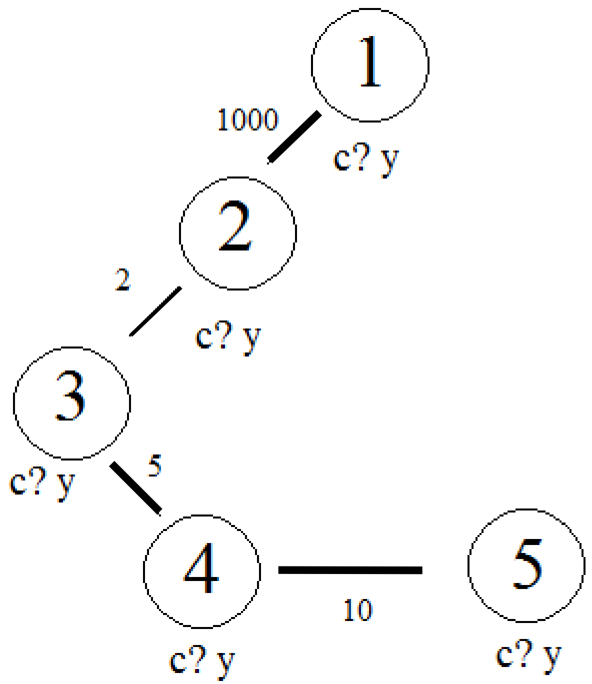


Uninformed Search Revisited: Depth First Search

The whole graph

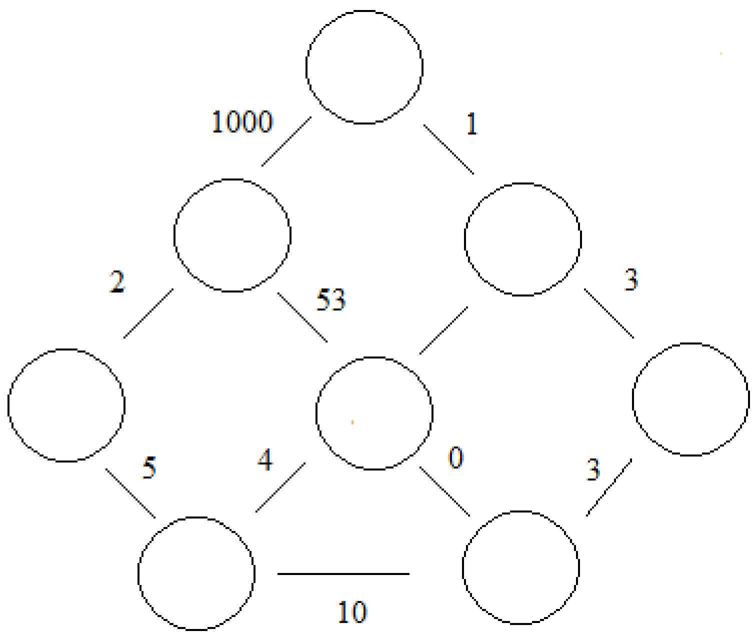


What we know

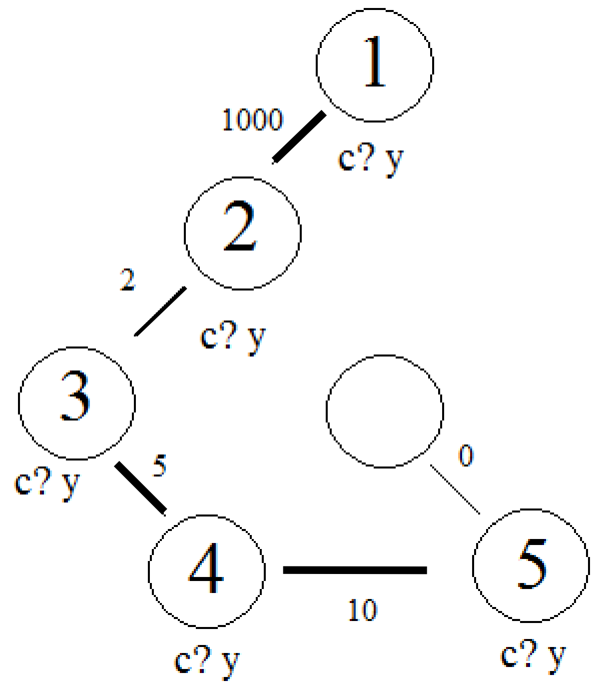


Uninformed Search Revisited: Depth First Search

The whole graph

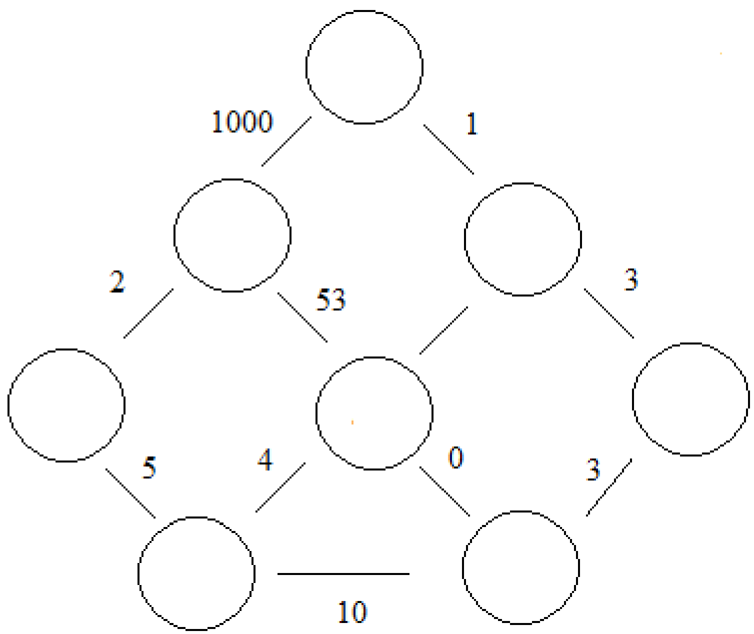


What we know

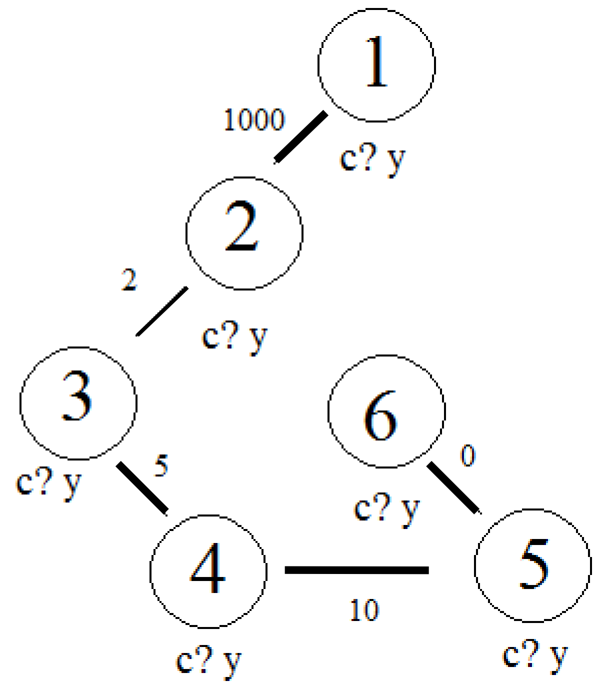


Uninformed Search Revisited: Depth First Search

The whole graph

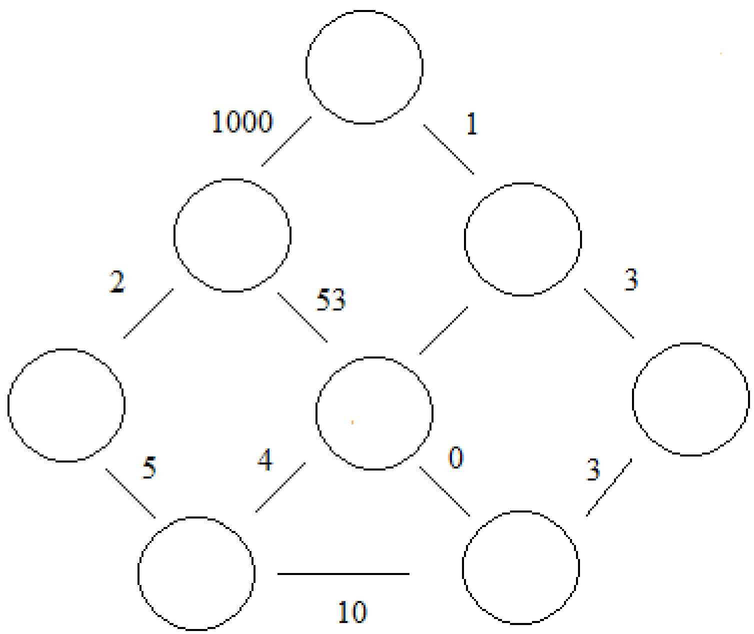


What we know

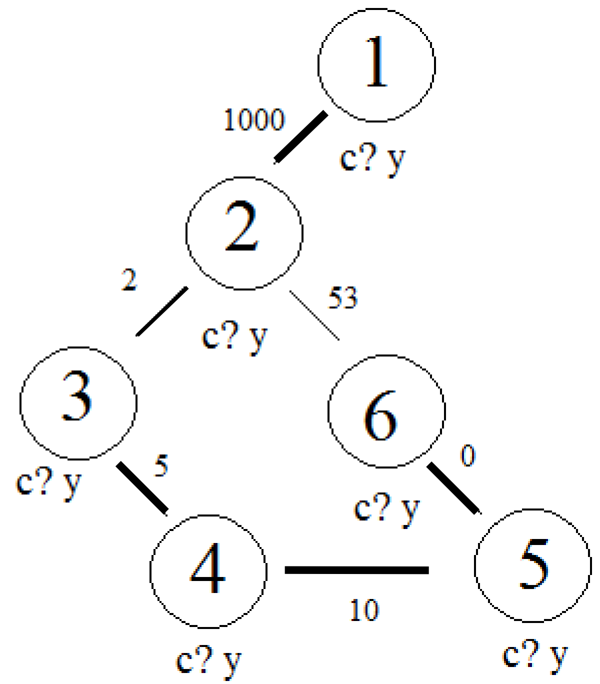


Uninformed Search Revisited: Depth First Search

The whole graph

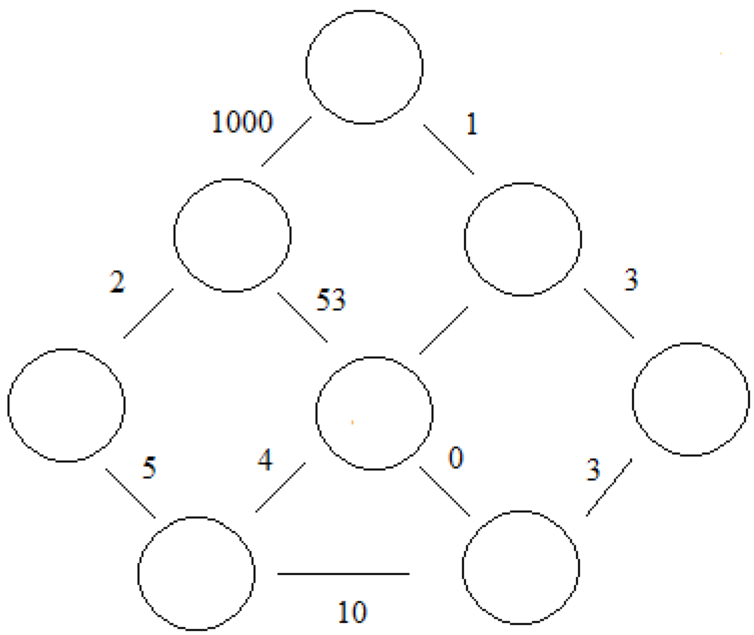


What we know

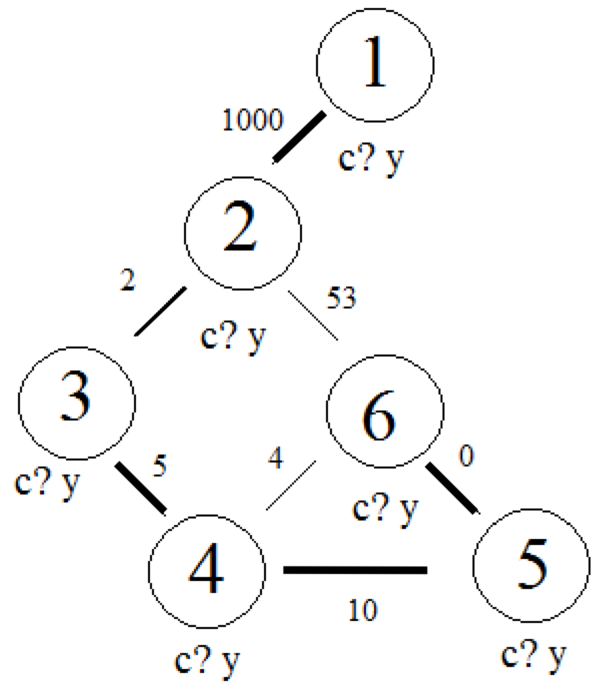


Uninformed Search Revisited: Depth First Search

The whole graph

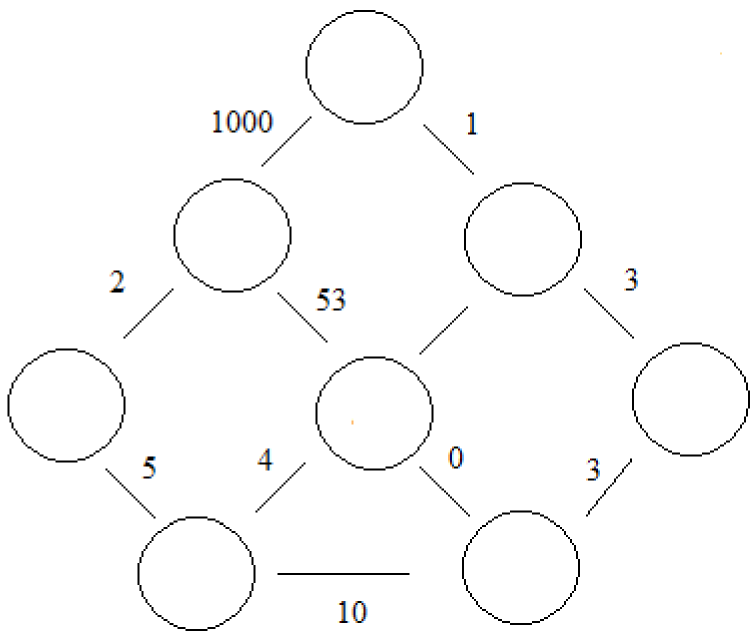


What we know

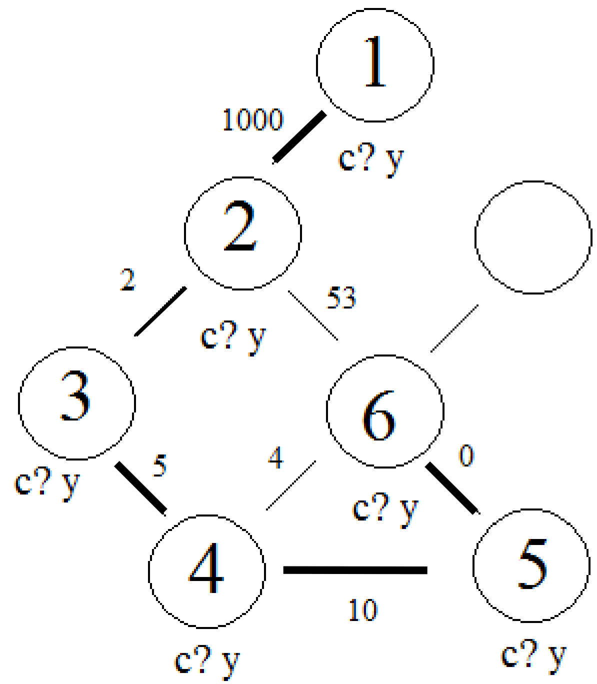


Uninformed Search Revisited: Depth First Search

The whole graph

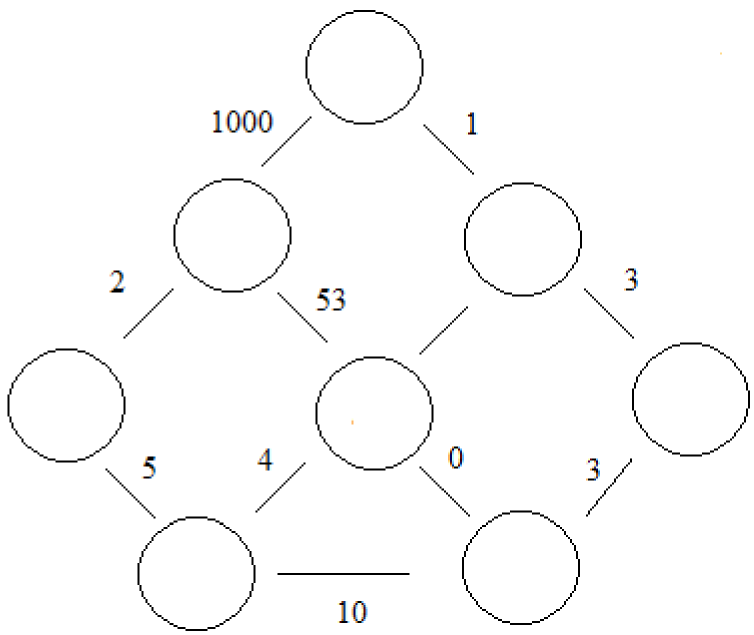


What we know

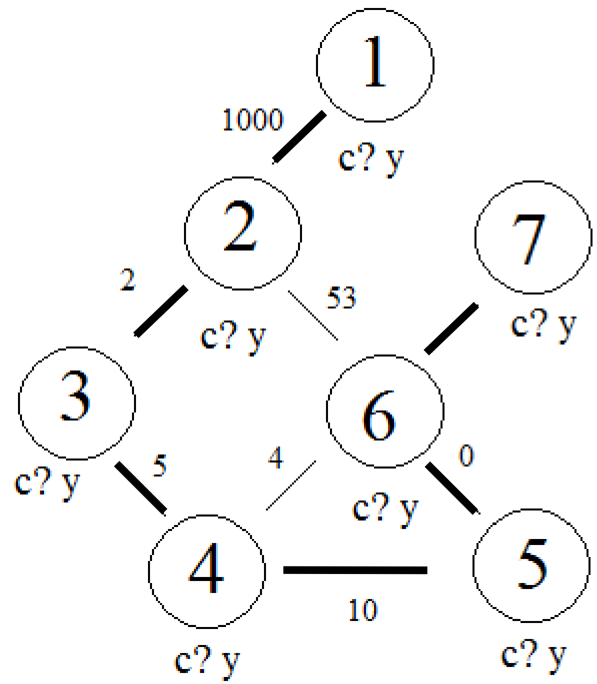


Uninformed Search Revisited: Depth First Search

The whole graph

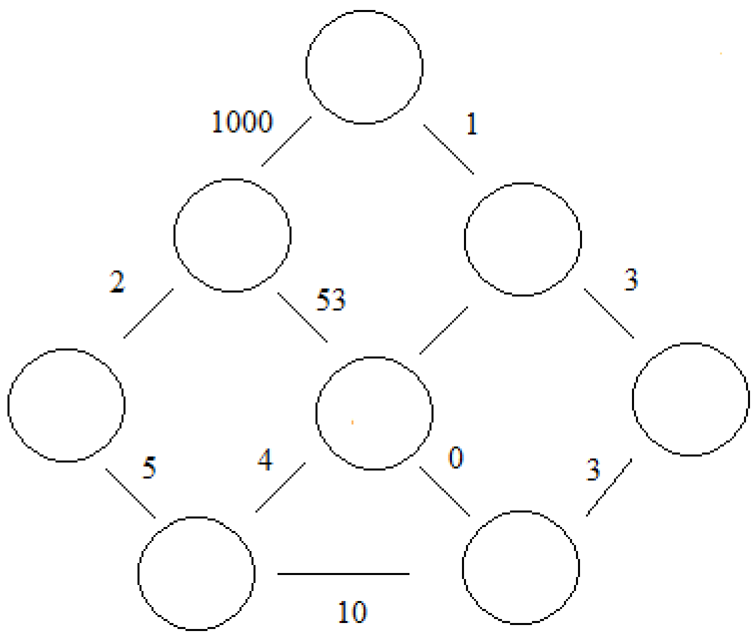


What we know

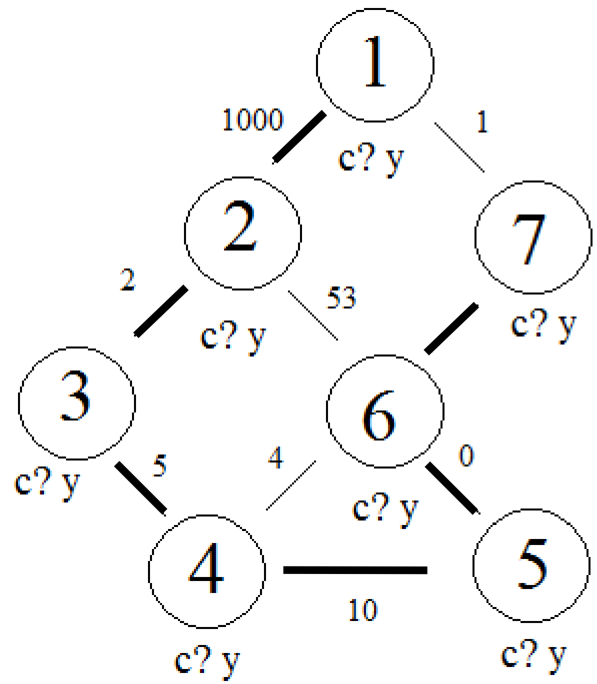


Uninformed Search Revisited: Depth First Search

The whole graph

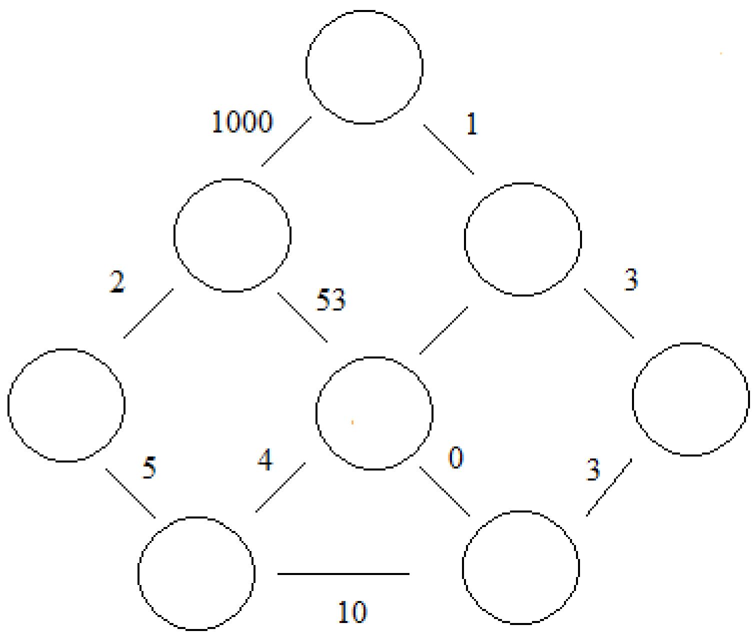


What we know

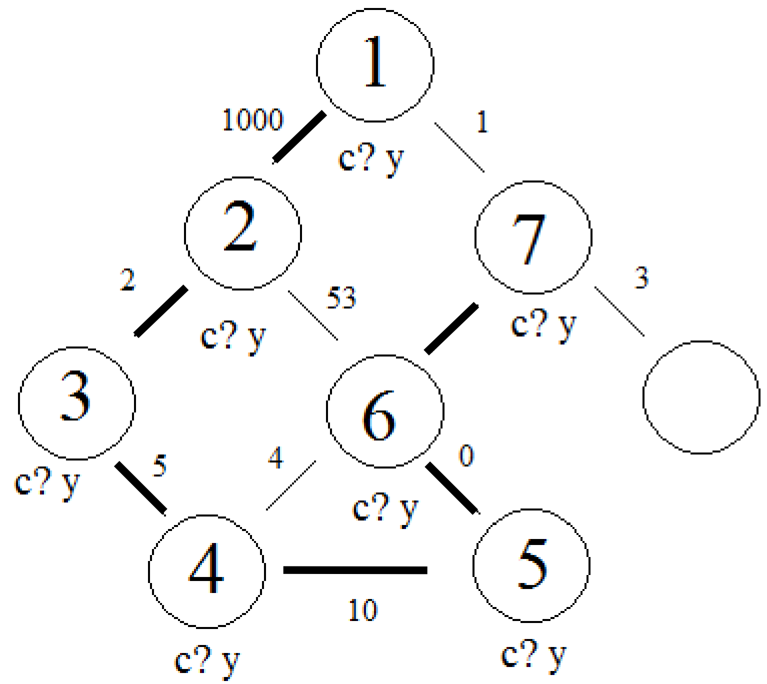


Uninformed Search Revisited: Depth First Search

The whole graph

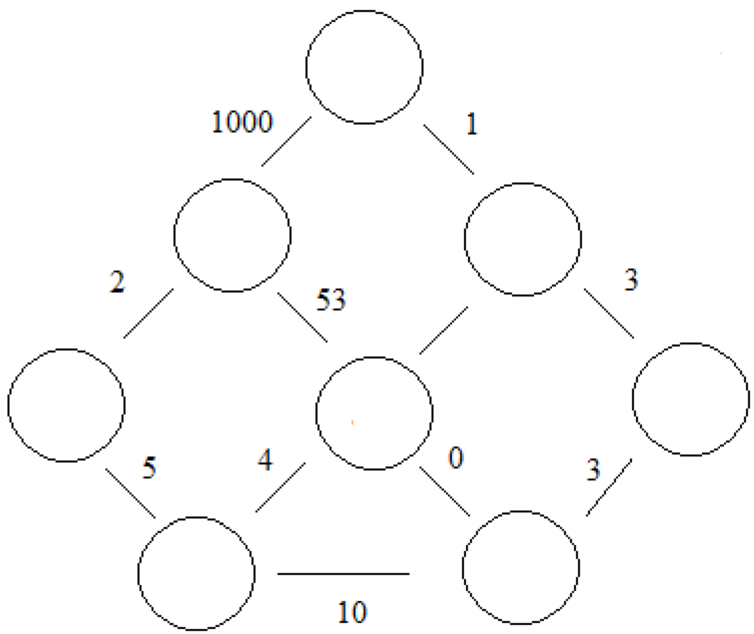


What we know

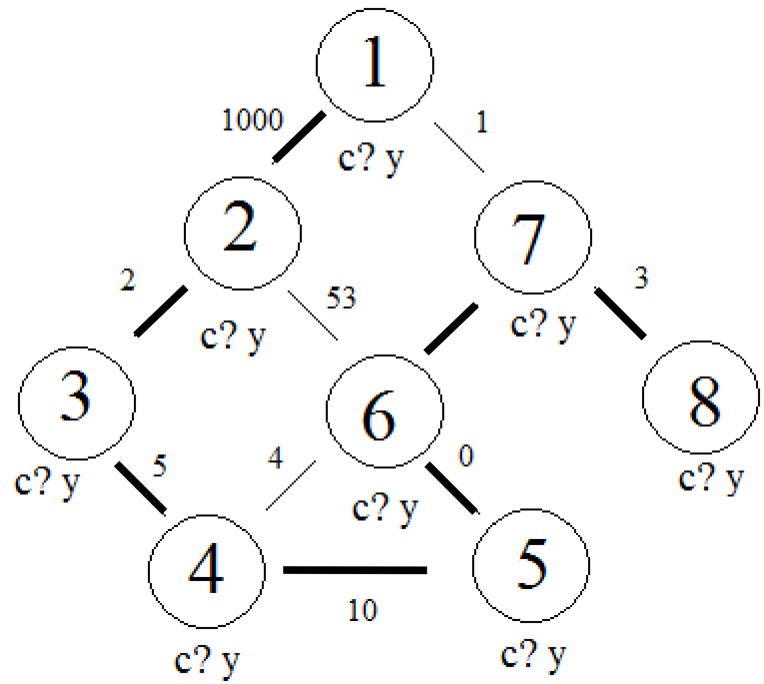


Uninformed Search Revisited: Depth First Search

The whole graph

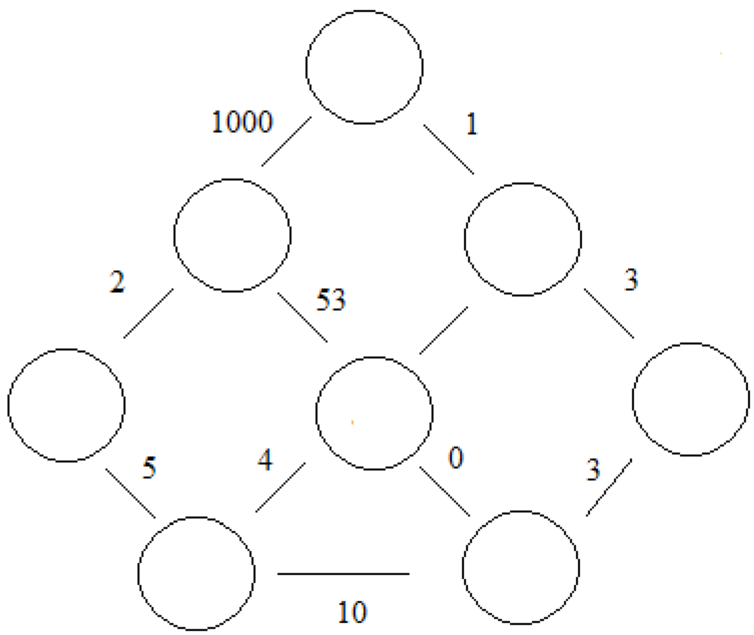


What we know

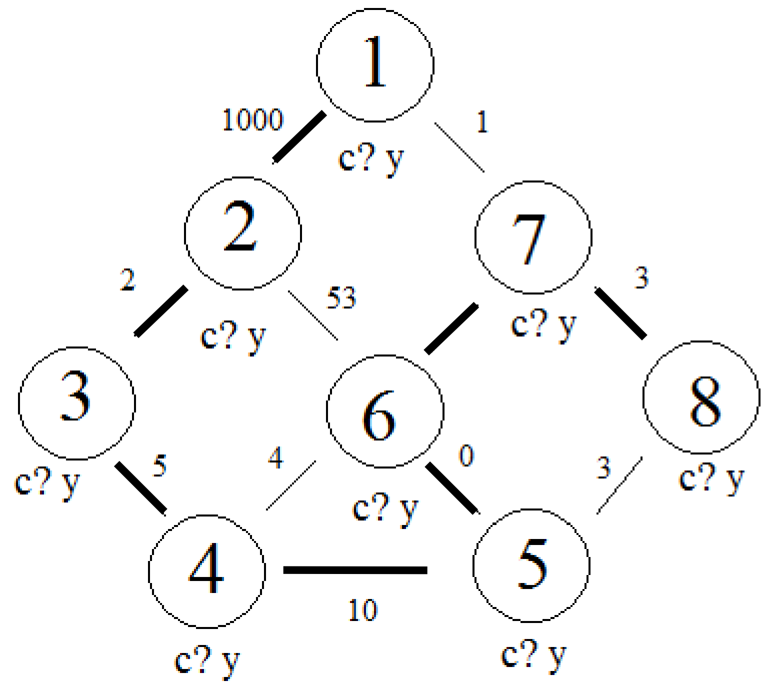


Uninformed Search Revisited: Depth First Search

The whole graph

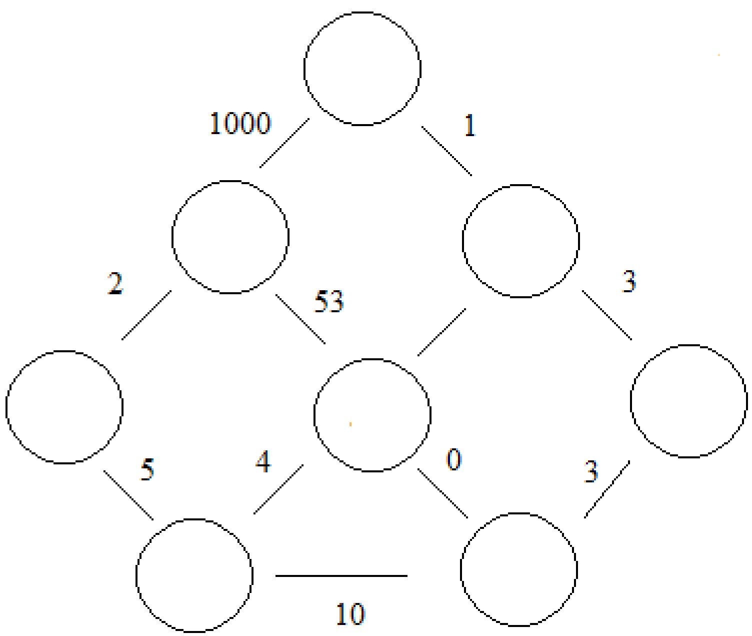


What we know

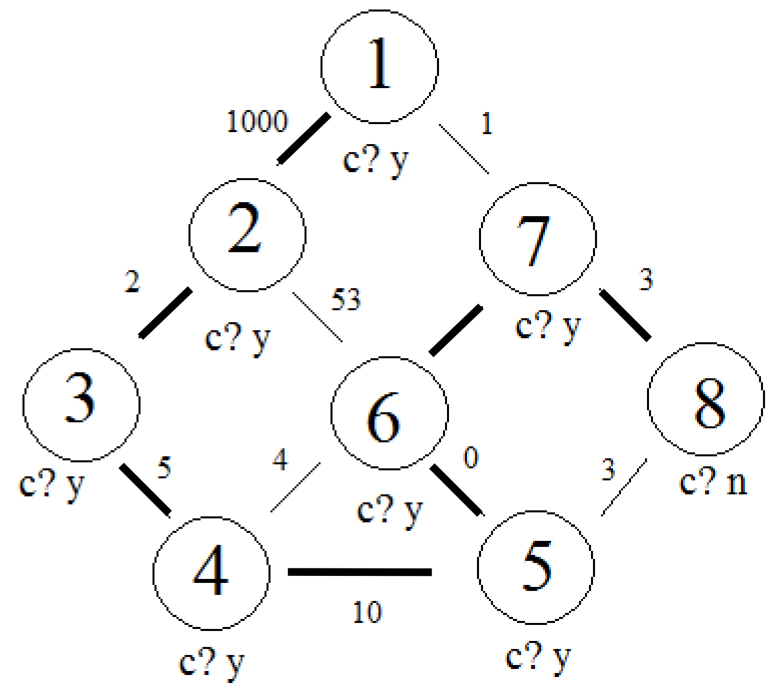


Uninformed Search Revisited: Depth First Search

The whole graph

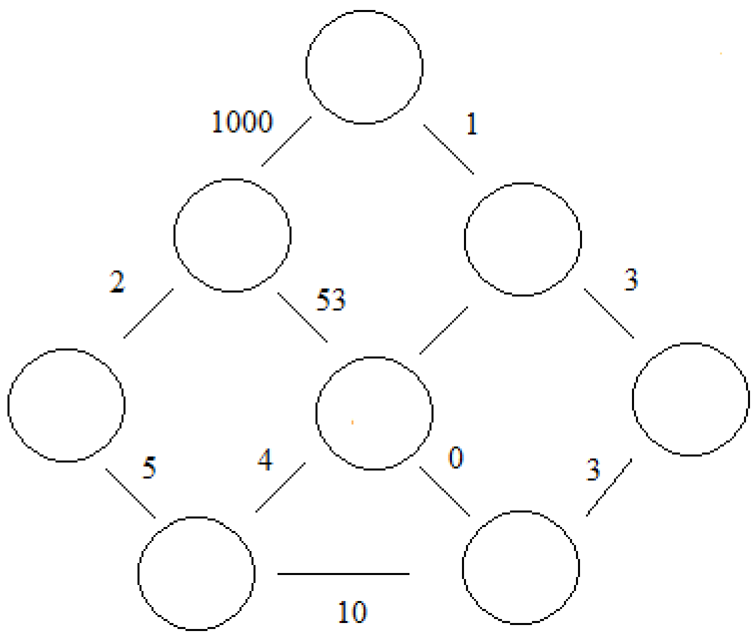


What we know

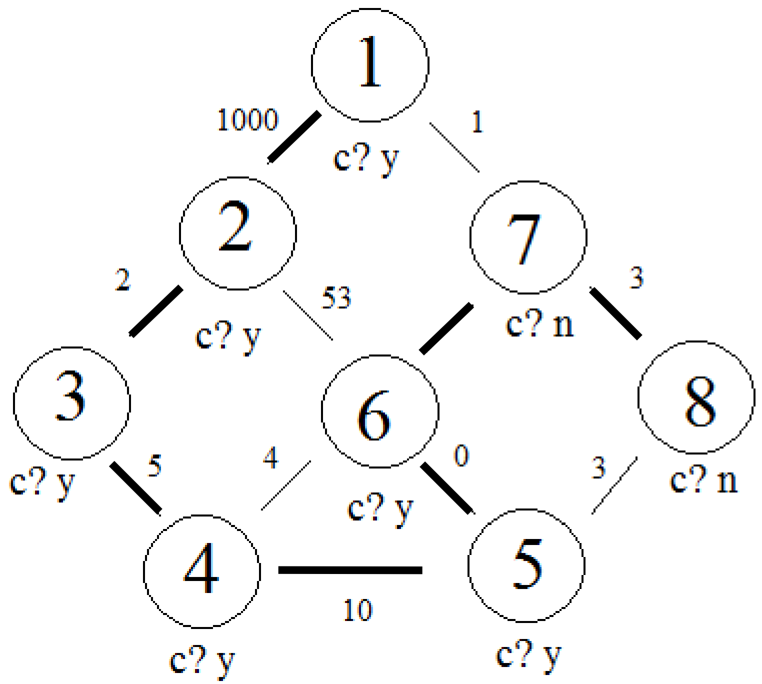


Uninformed Search Revisited: Depth First Search

The whole graph

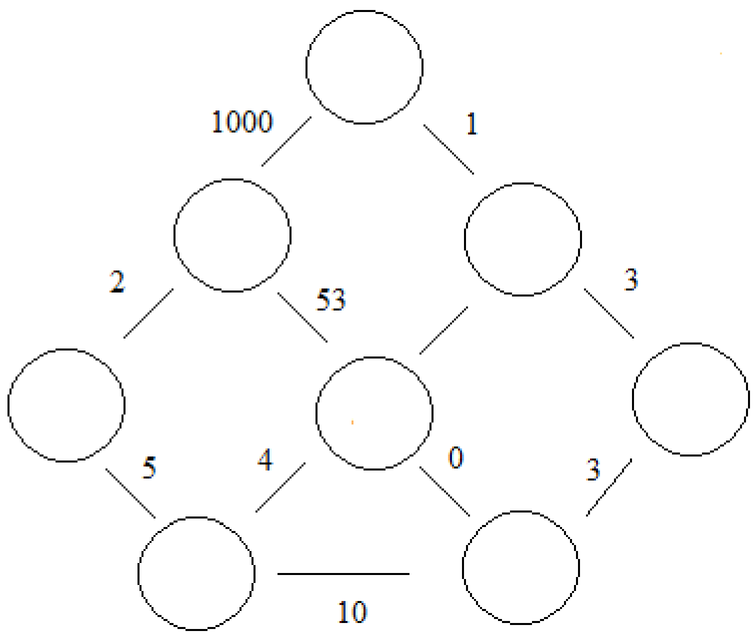


What we know

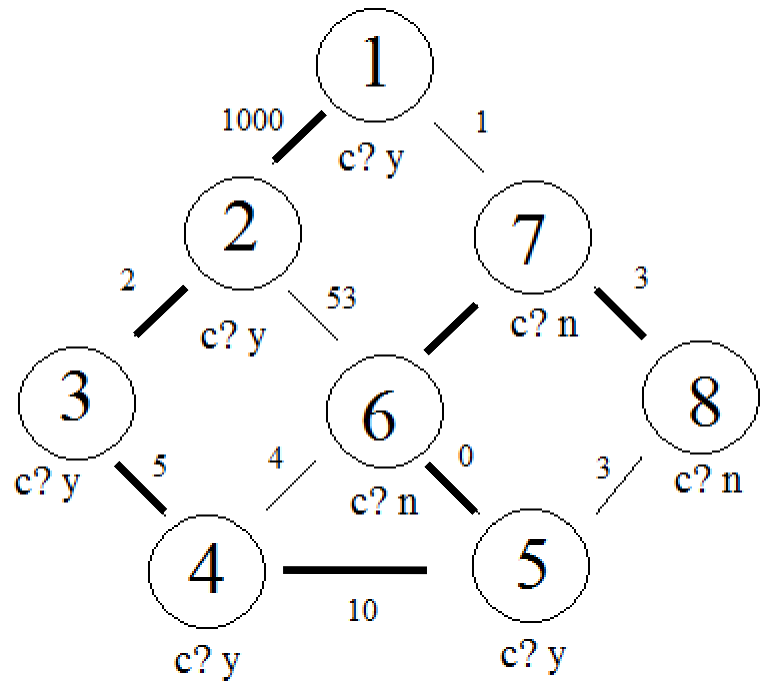


Uninformed Search Revisited: Depth First Search

The whole graph

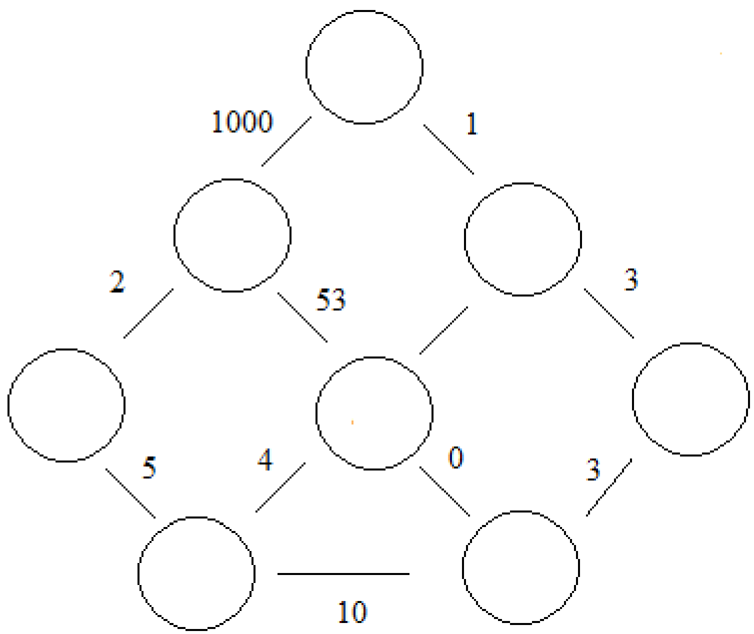


What we know

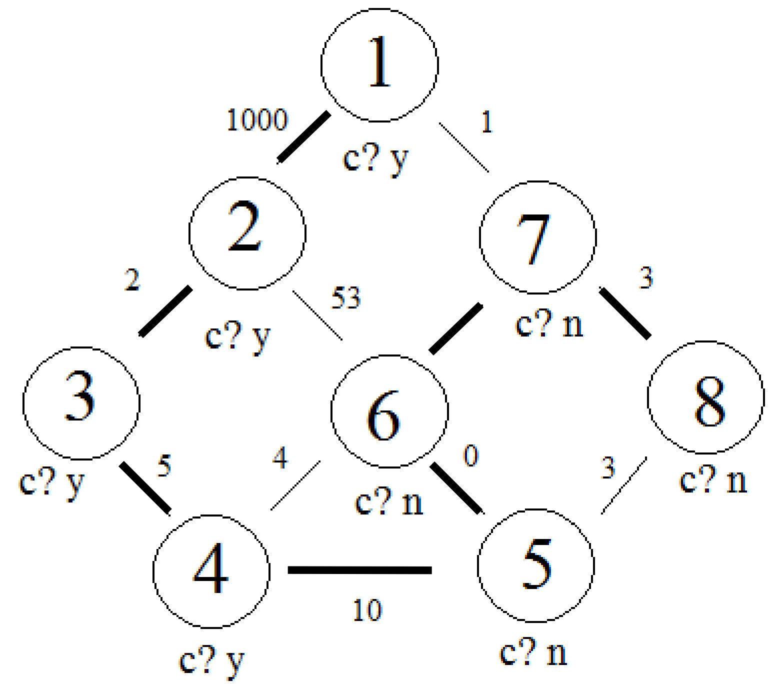


Uninformed Search Revisited: Depth First Search

The whole graph

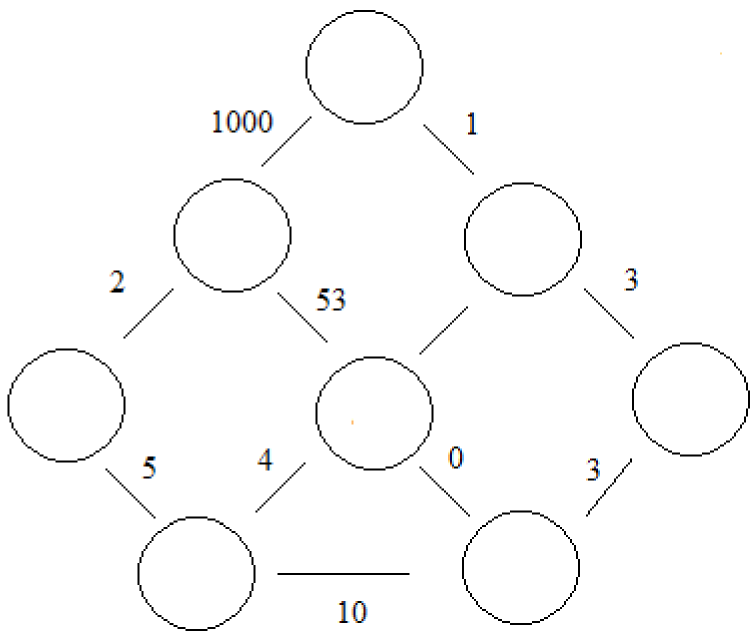


What we know

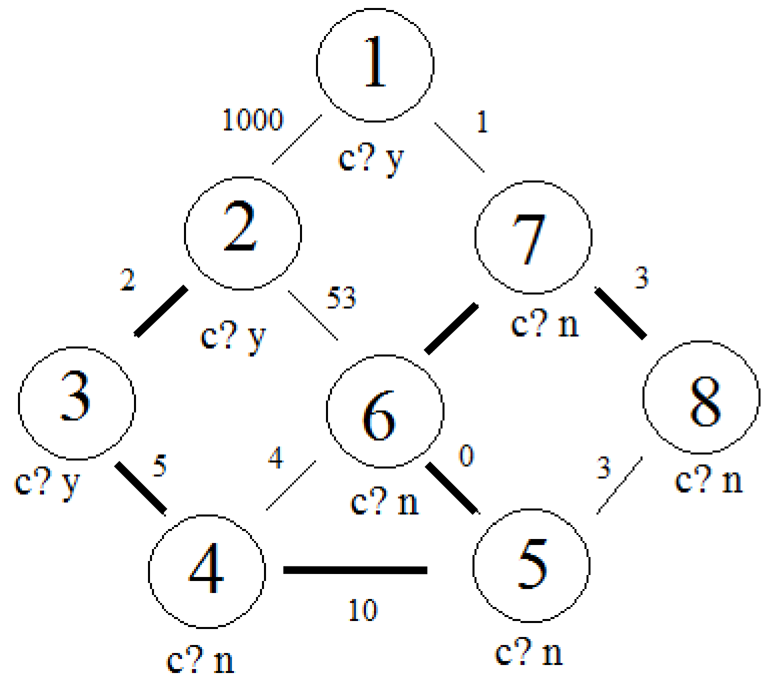


Uninformed Search Revisited: Depth First Search

The whole graph

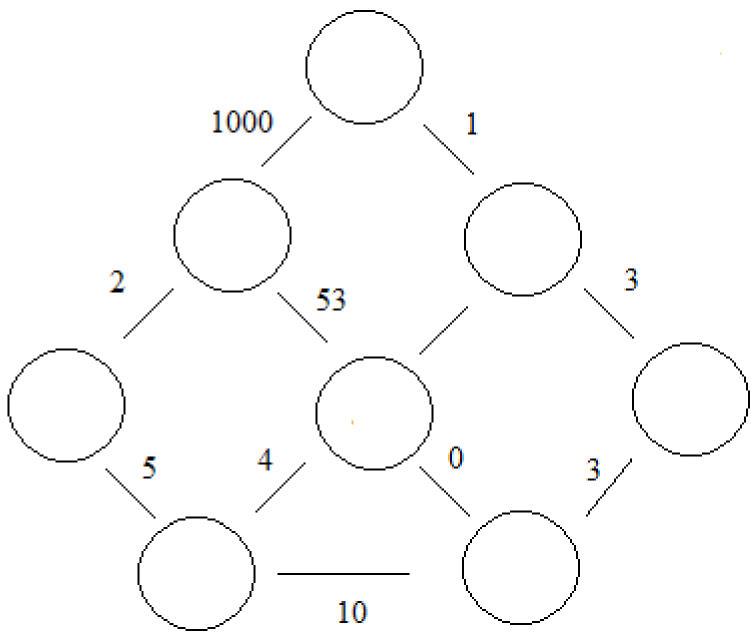


What we know

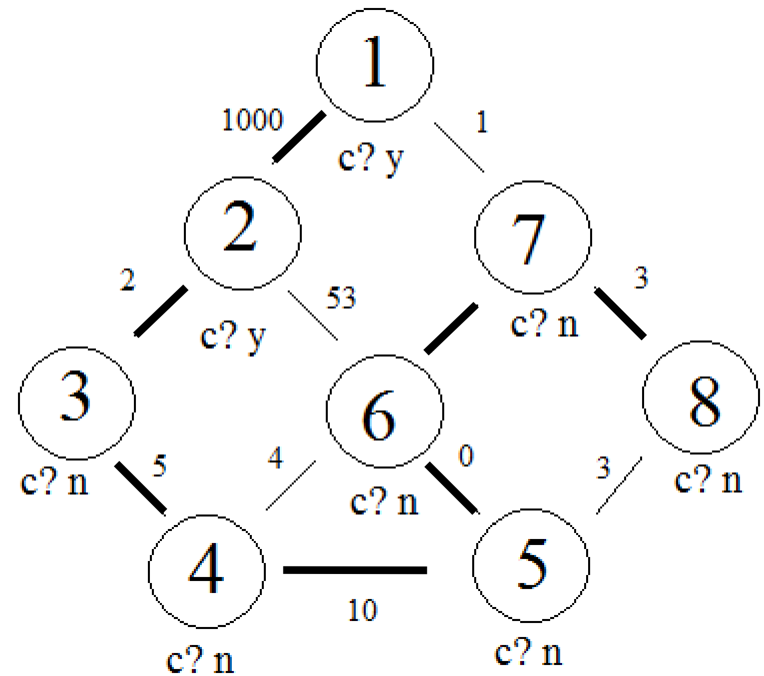


Uninformed Search Revisited: Depth First Search

The whole graph

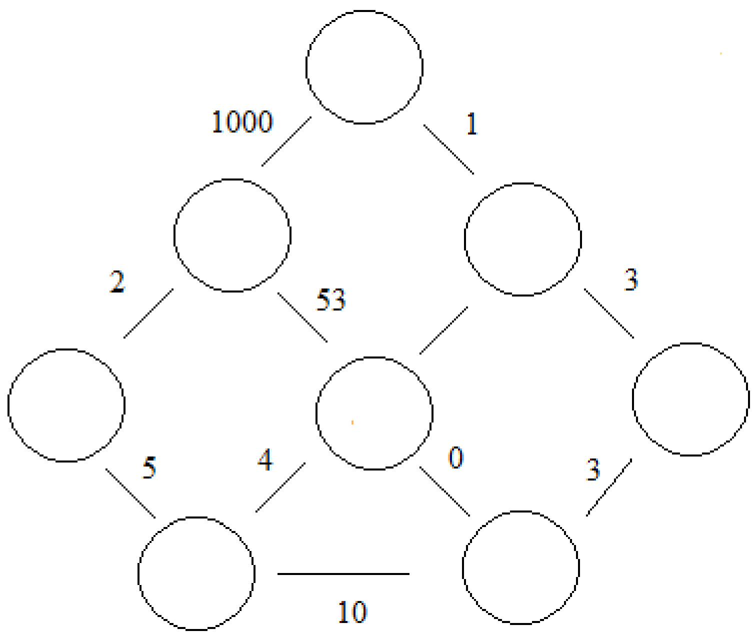


What we know

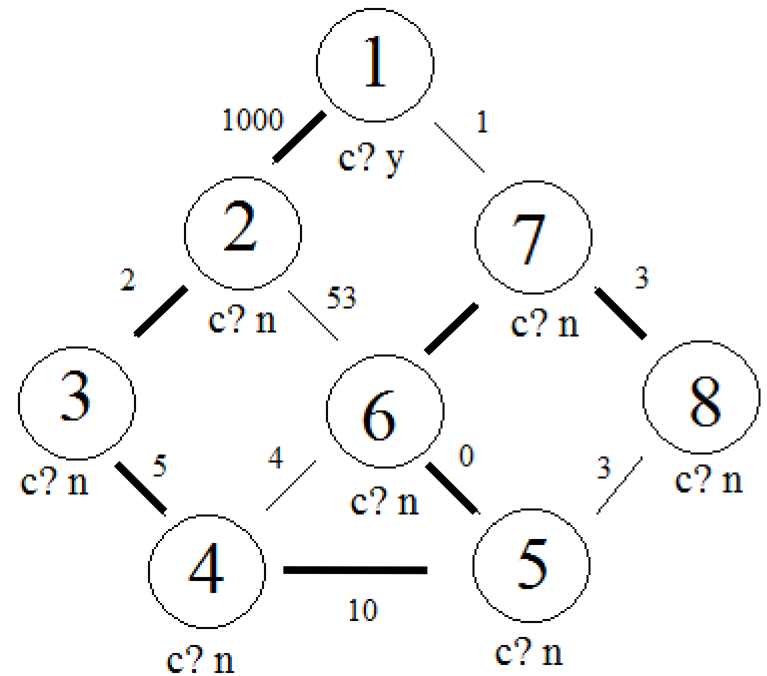


Uninformed Search Revisited: Depth First Search

The whole graph

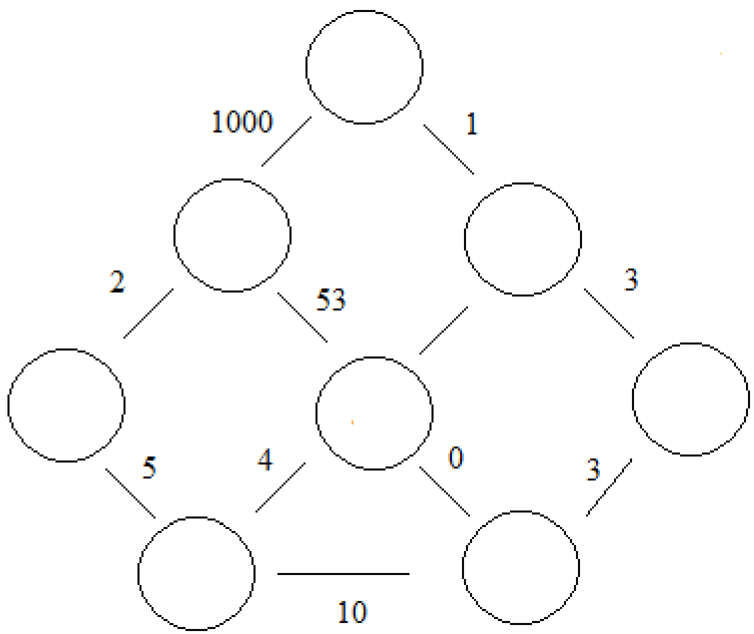


What we know

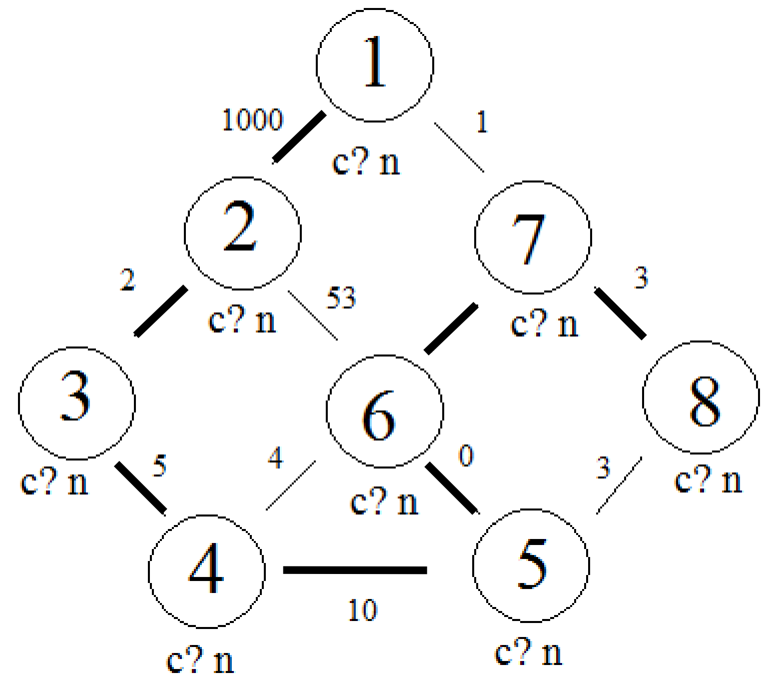


Uninformed Search Revisited: Depth First Search

The whole graph



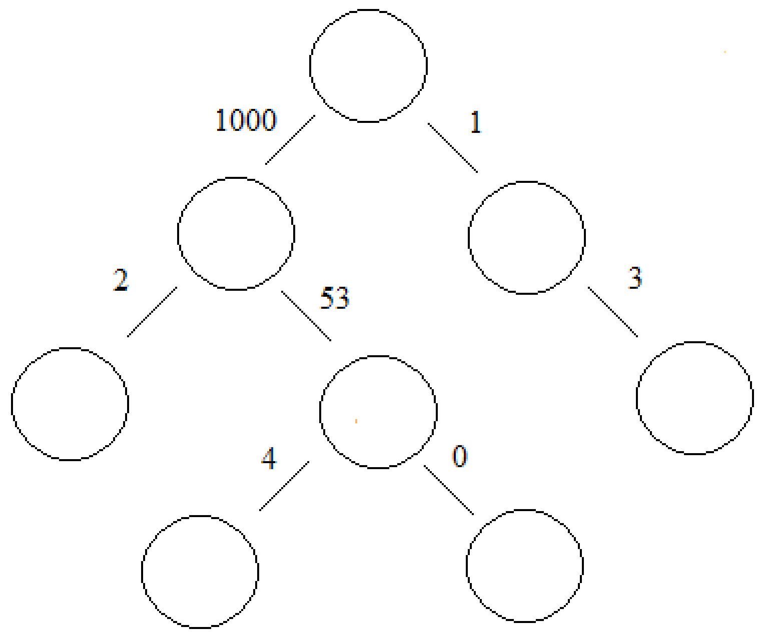
What we know



Uninformed Search Revisited: Breadth First Search

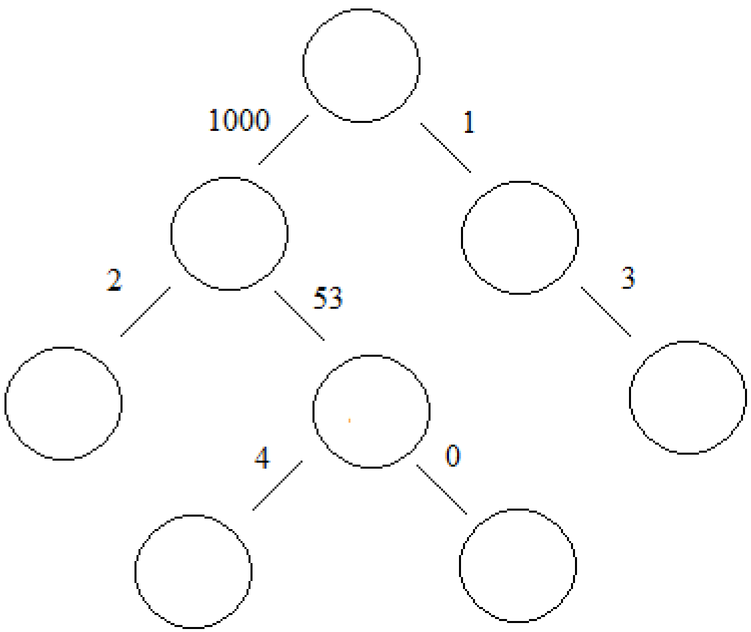
- Another way of thinking about it:
  - First, look at all nodes 1 arc away from the start
  - Then, look at all nodes 2 arcs away from the start
  - Next, look at all nodes 3 arcs away from the start
  - etc ...
- When you are told about nodes that are too far away, save them for later.

Uninformed Search Revisited: Breadth First Search

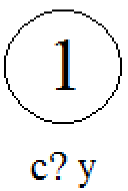


Uninformed Search Revisited: Breadth First Search

The whole graph

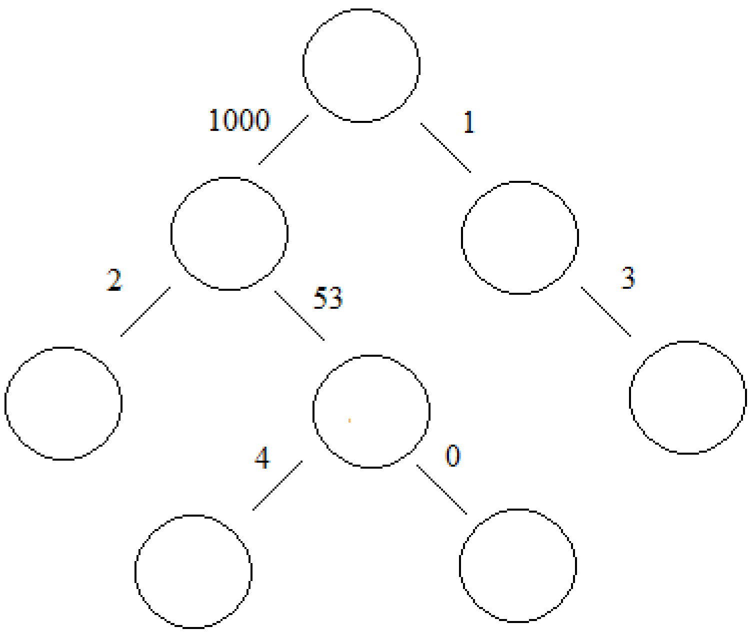


What we know

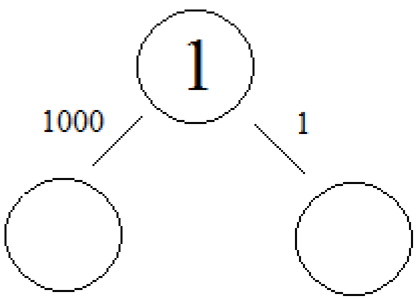


Uninformed Search Revisited: Breadth First Search

The whole graph

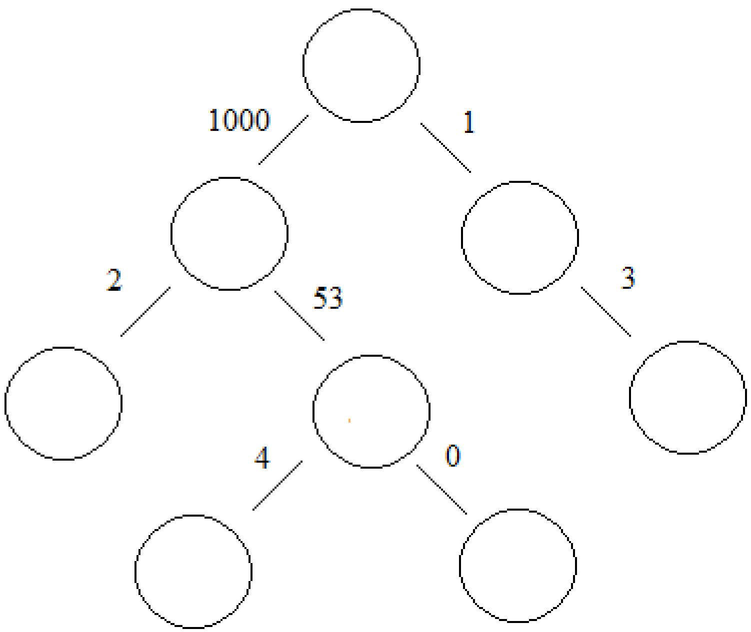


What we know

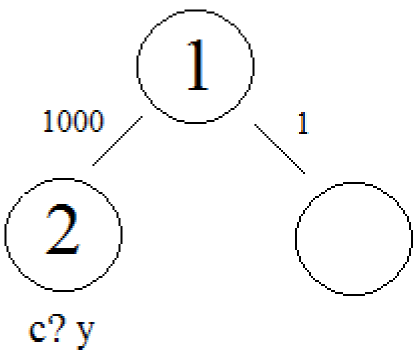


Uninformed Search Revisited: Breadth First Search

The whole graph

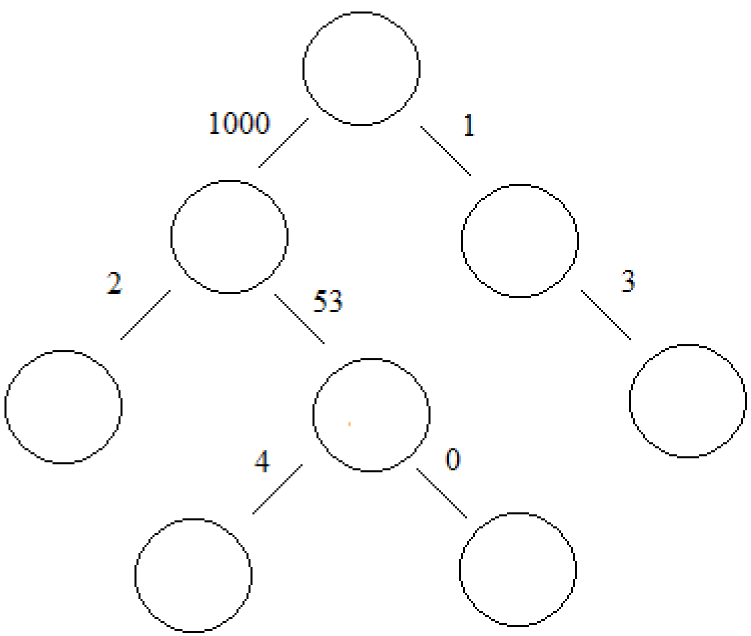


What we know

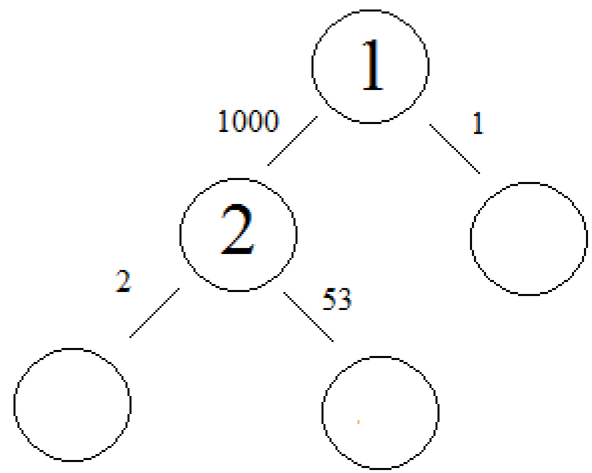


Uninformed Search Revisited: Breadth First Search

The whole graph

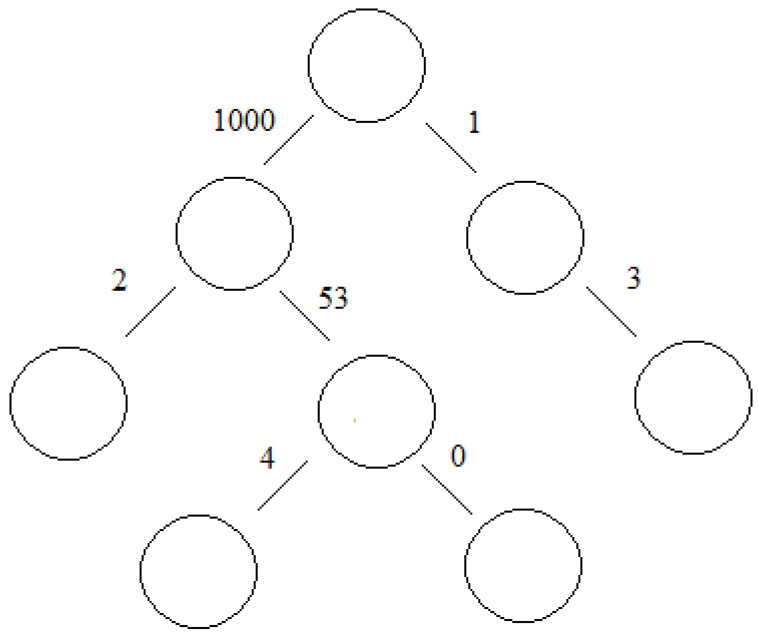


What we know

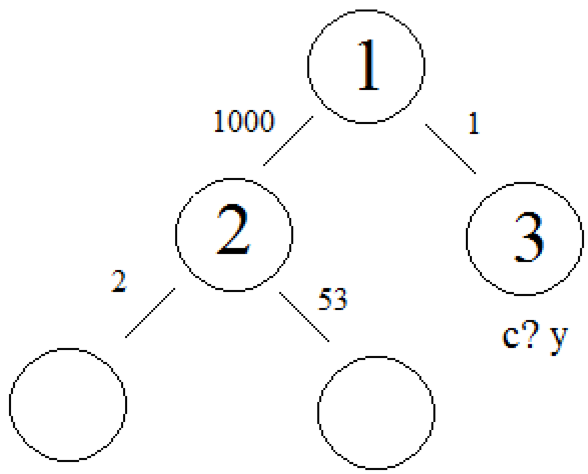


Uninformed Search Revisited: Breadth First Search

The whole graph

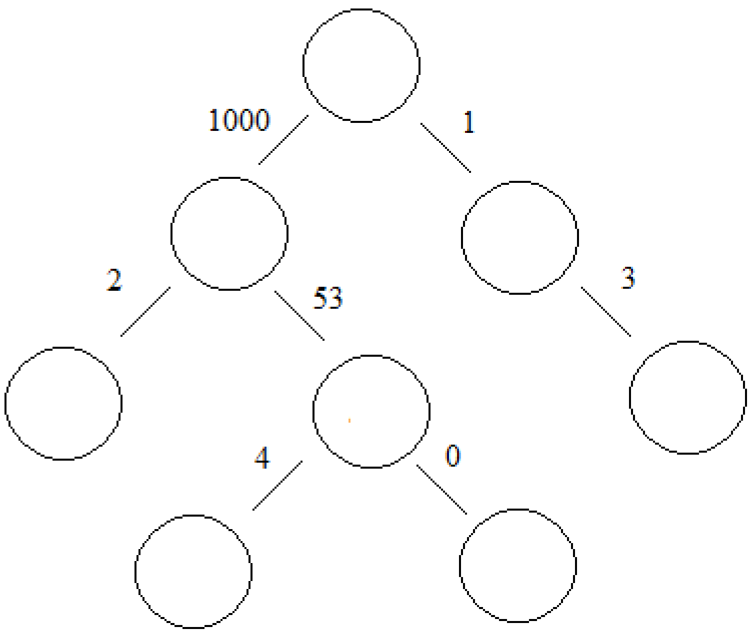


What we know

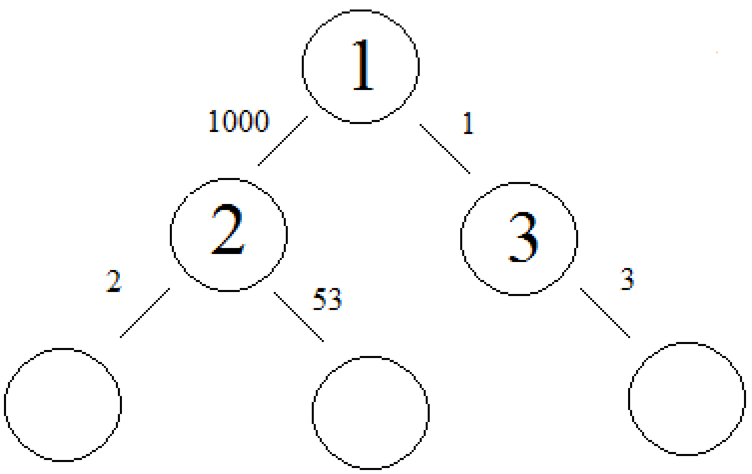


Uninformed Search Revisited: Breadth First Search

The whole graph

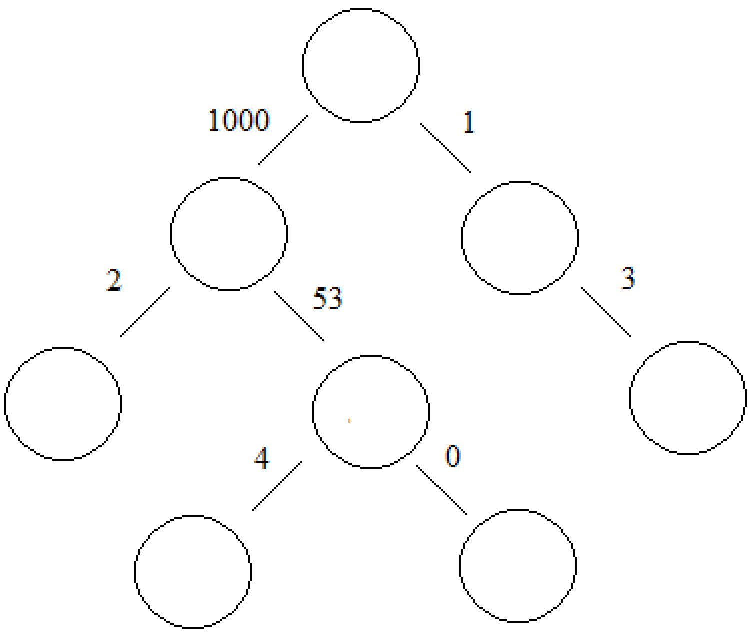


What we know

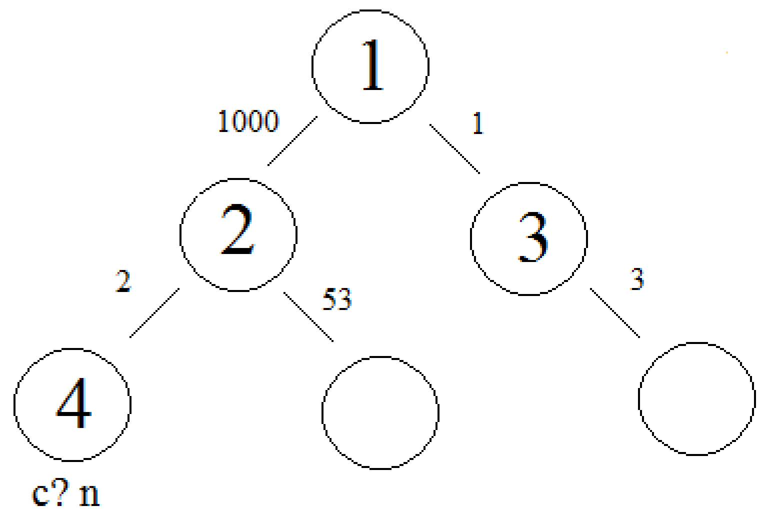


Uninformed Search Revisited: Breadth First Search

The whole graph

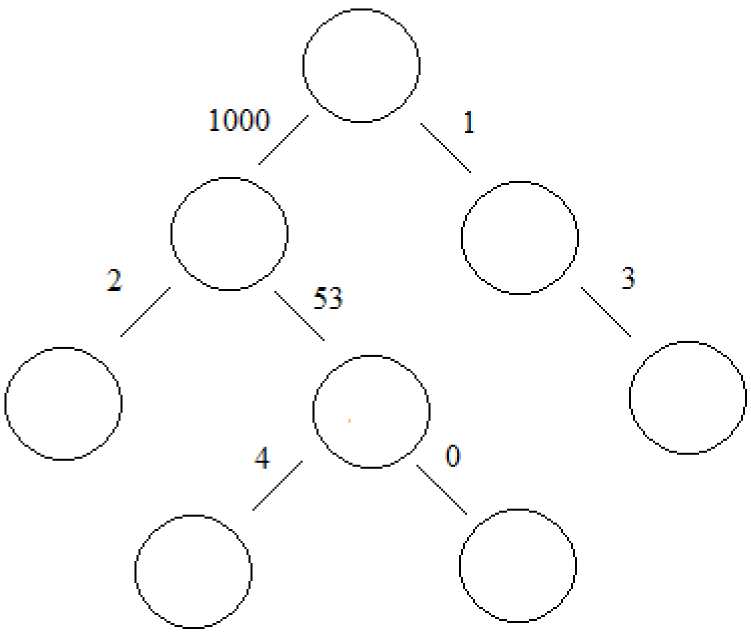


What we know

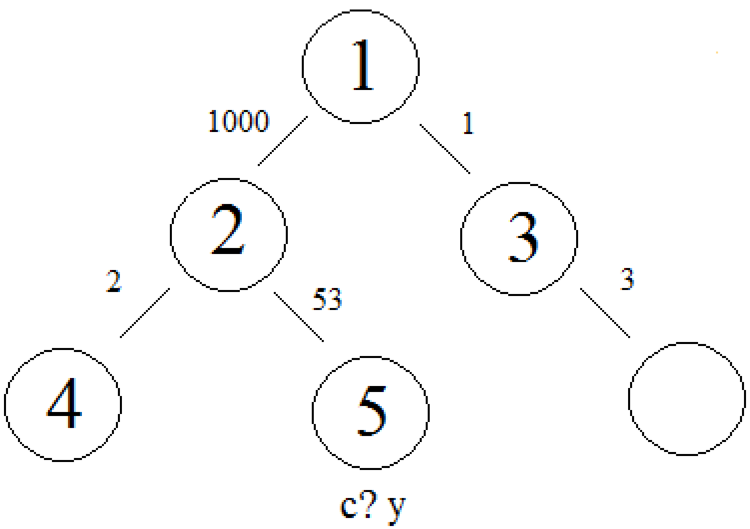


Uninformed Search Revisited: Breadth First Search

The whole graph

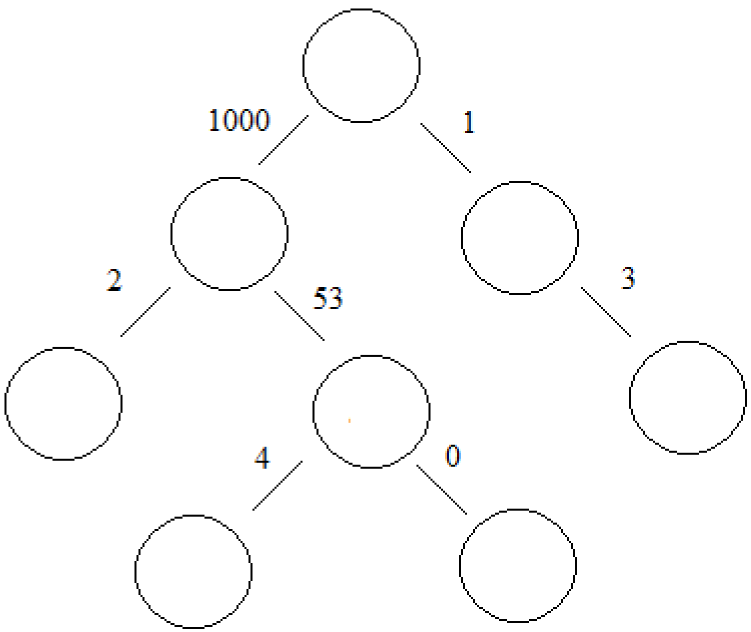


What we know

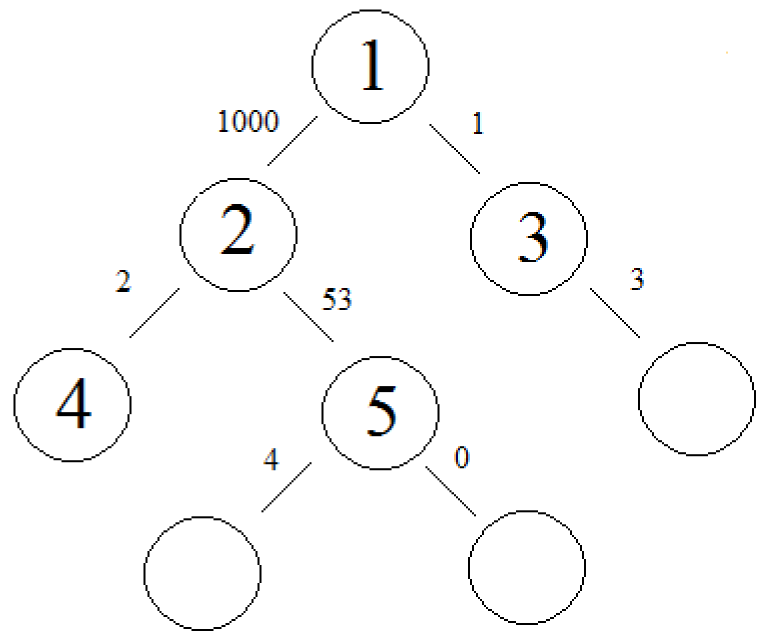


Uninformed Search Revisited: Breadth First Search

The whole graph

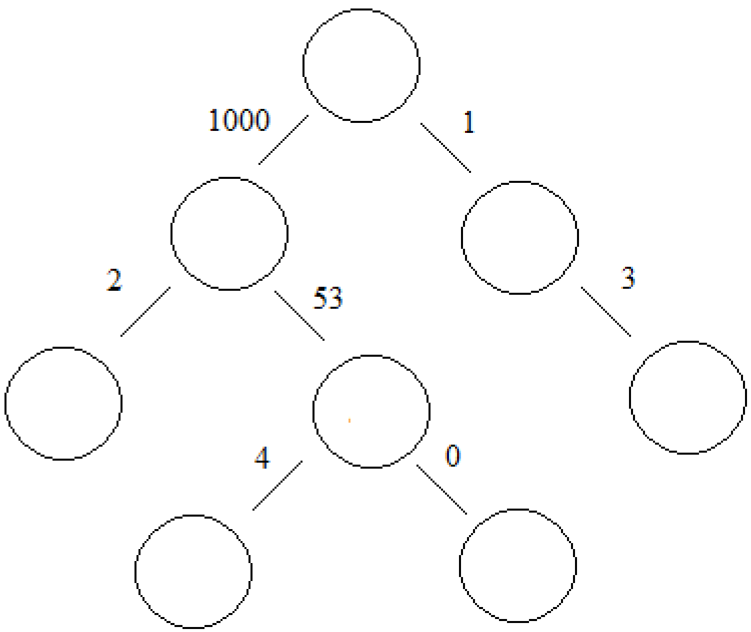


What we know

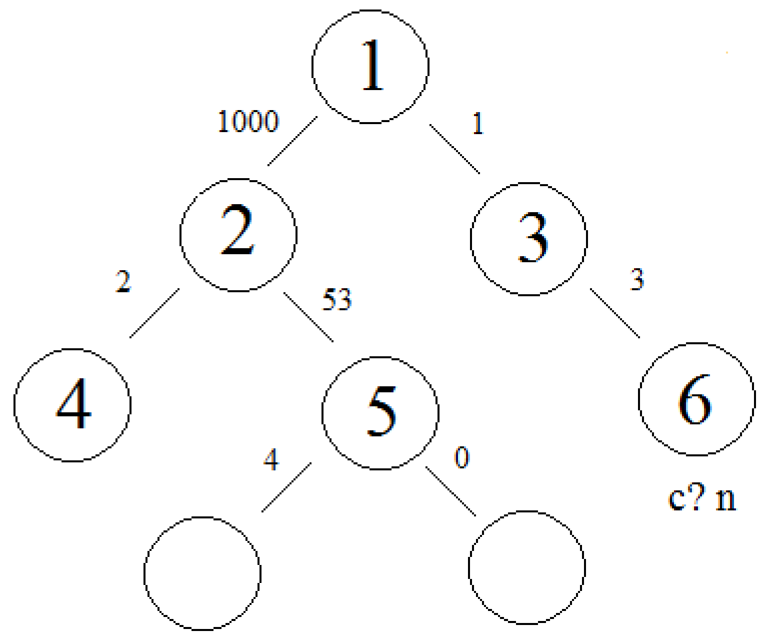


Uninformed Search Revisited: Breadth First Search

The whole graph

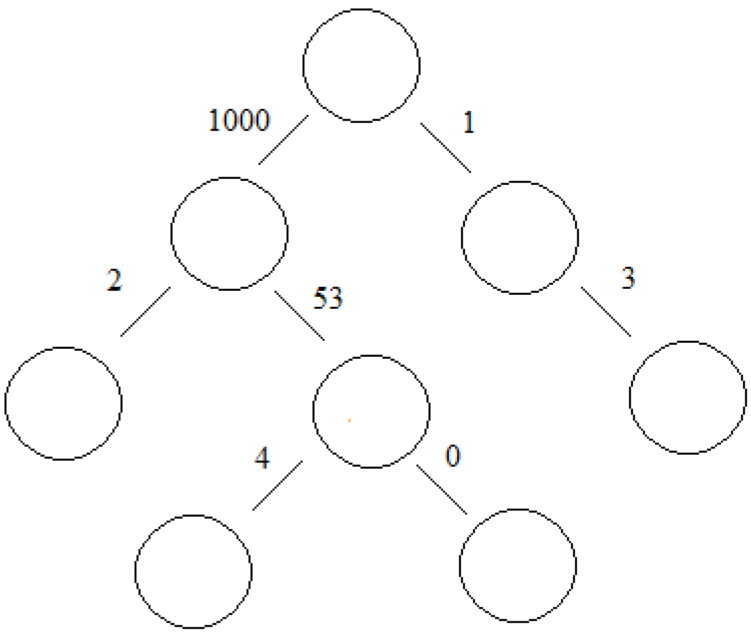


What we know

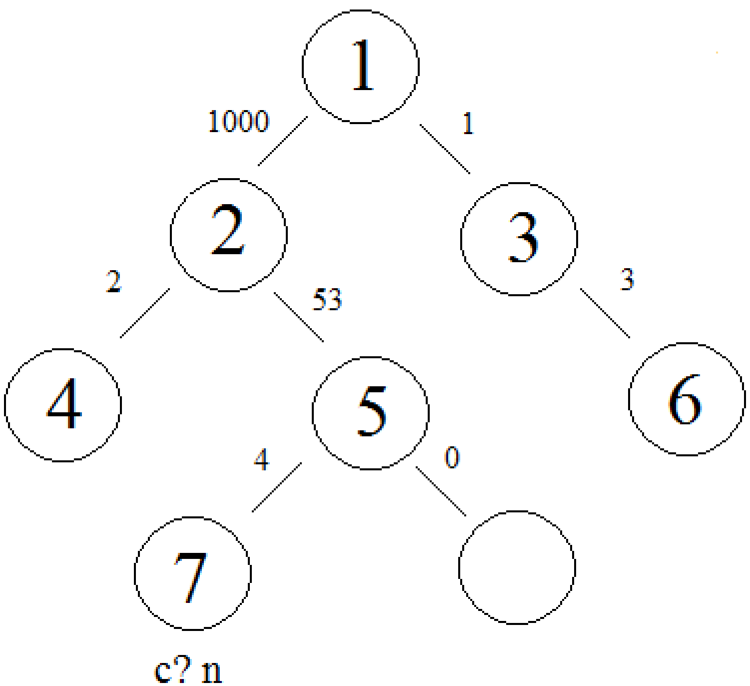


Uninformed Search Revisited: Breadth First Search

The whole graph

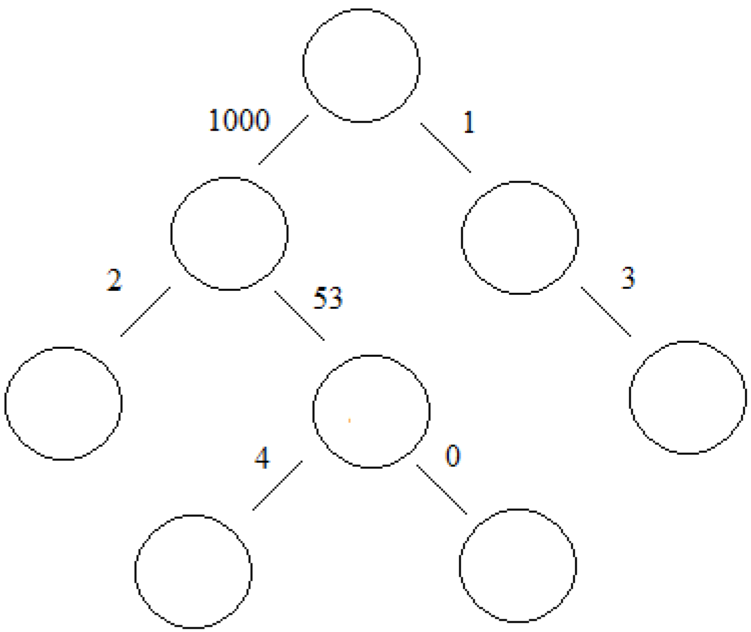


What we know

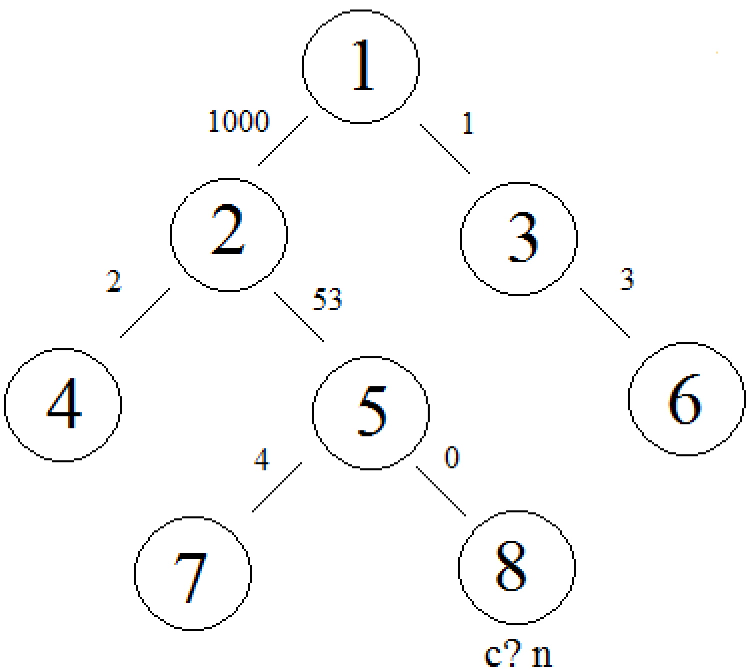


Uninformed Search Revisited: Breadth First Search

The whole graph

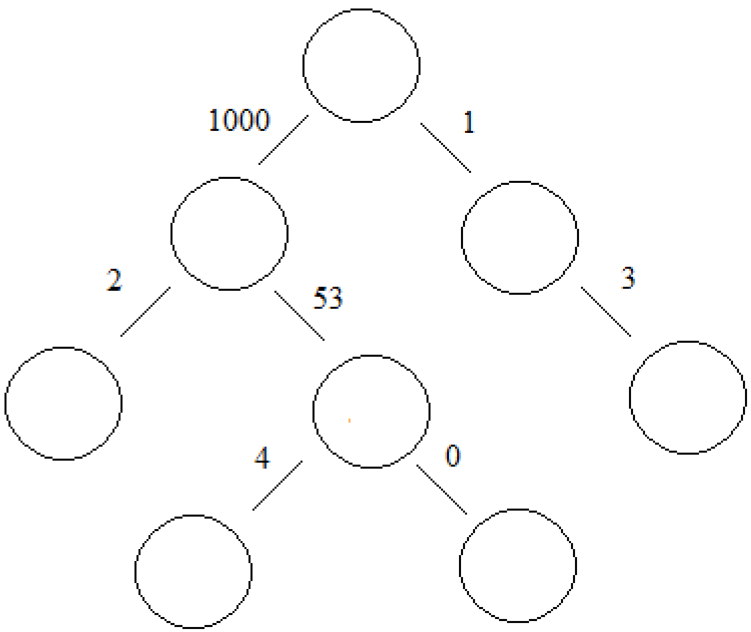


What we know

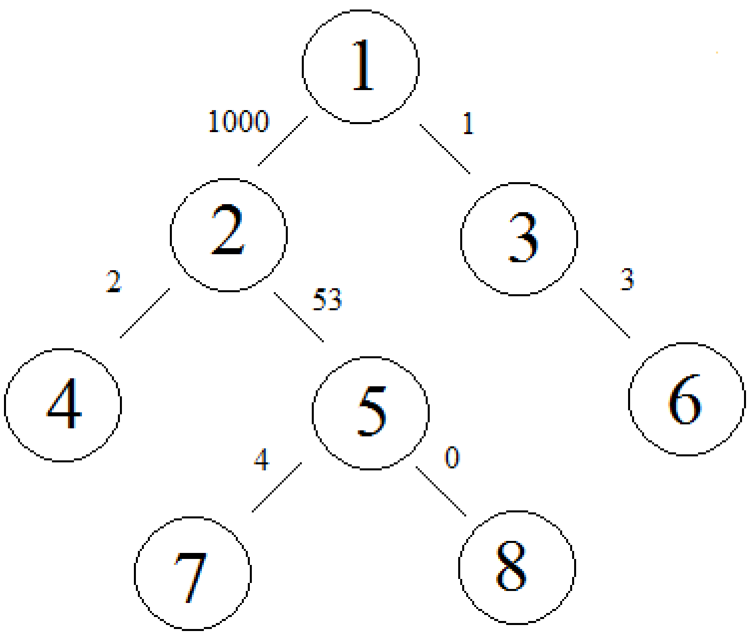


Uninformed Search Revisited: Breadth First Search

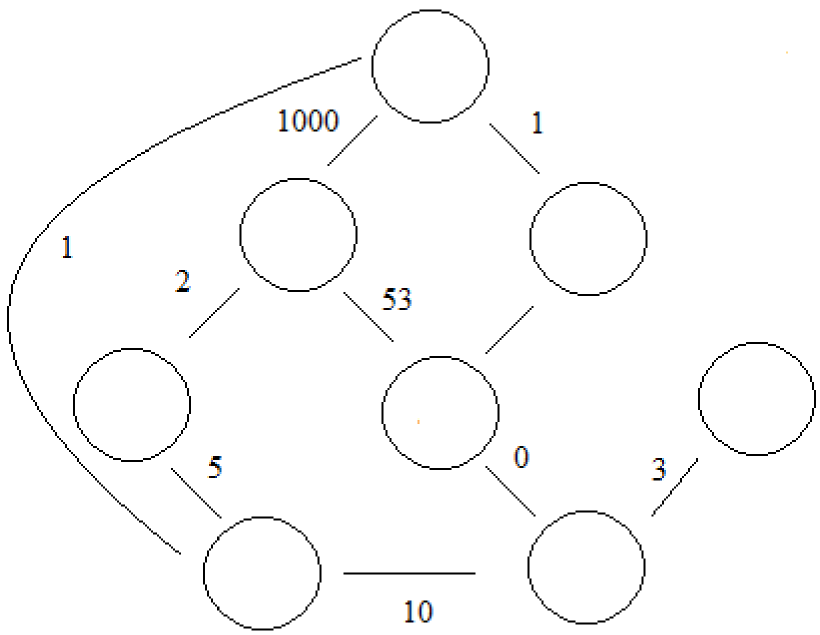
The whole graph



What we know

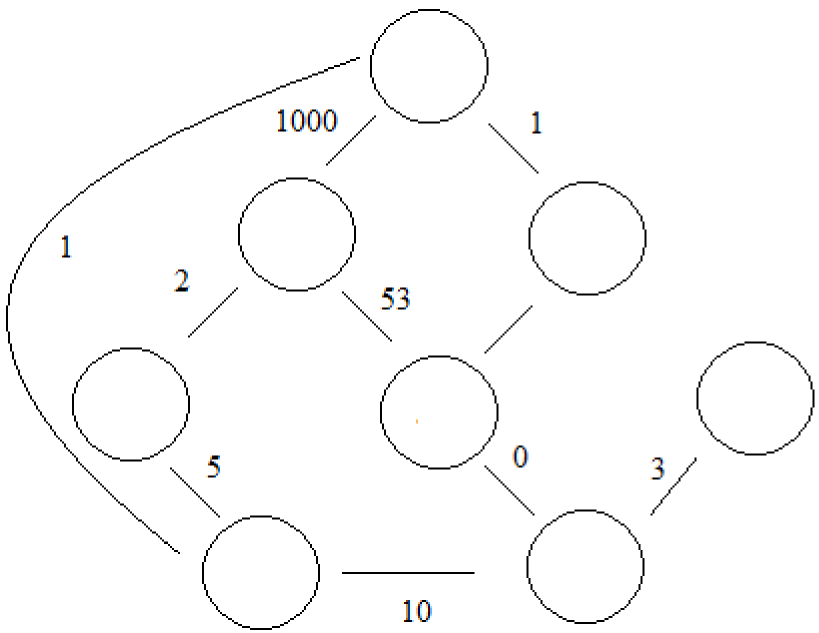


Uninformed Search Revisited: Breadth First Search



Uninformed Search Revisited: Breadth First Search

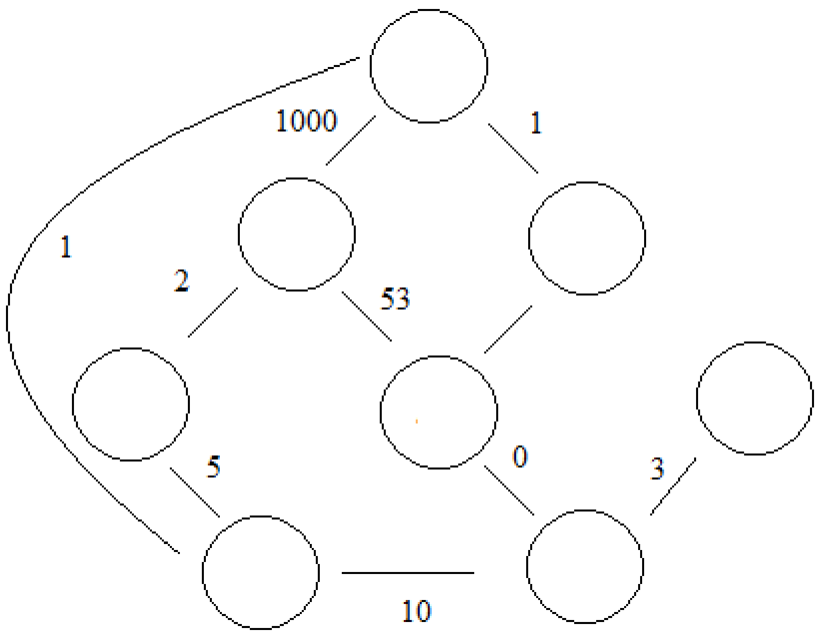
The whole graph



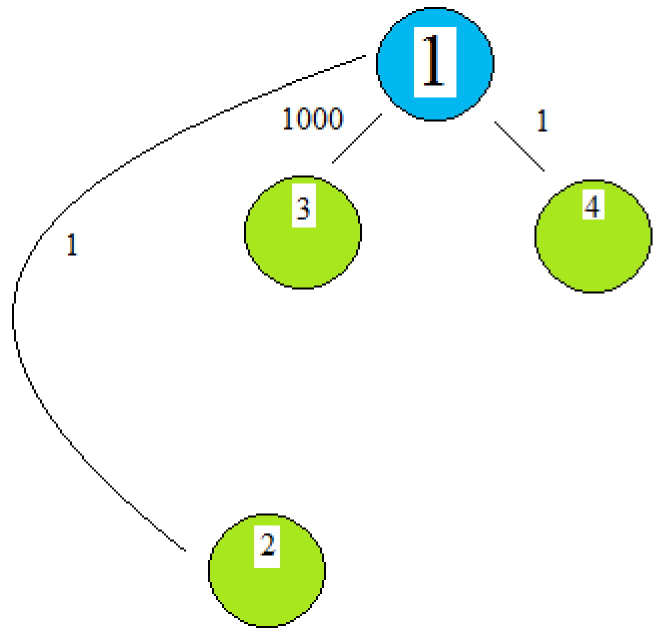
What we know

Uninformed Search Revisited: Breadth First Search

The whole graph

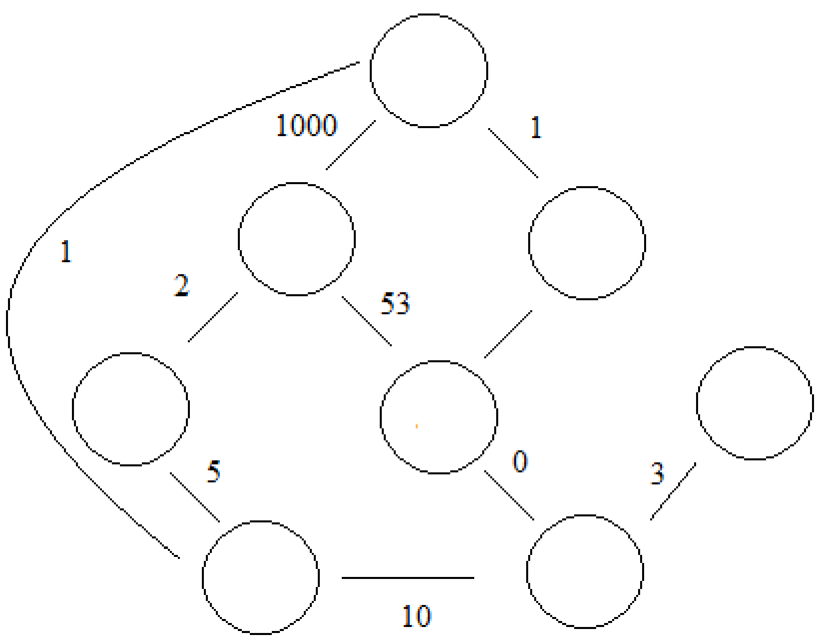


What we know

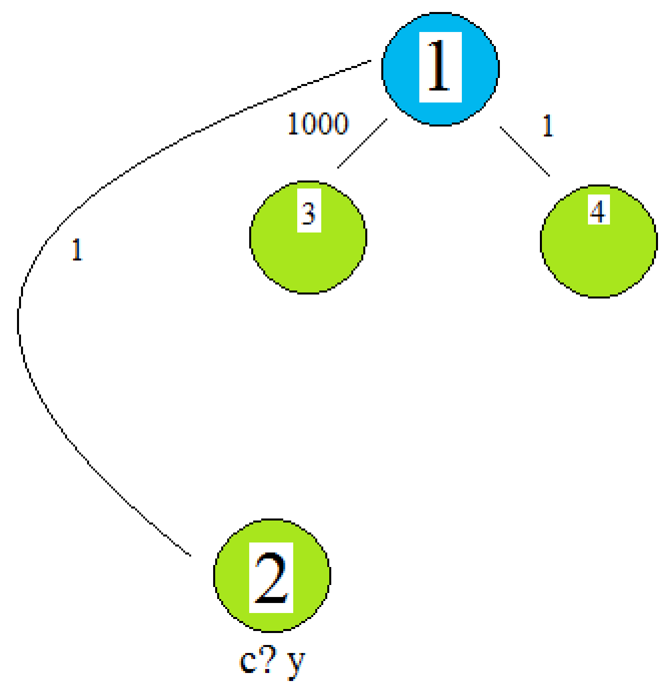


Uninformed Search Revisited: Breadth First Search

The whole graph

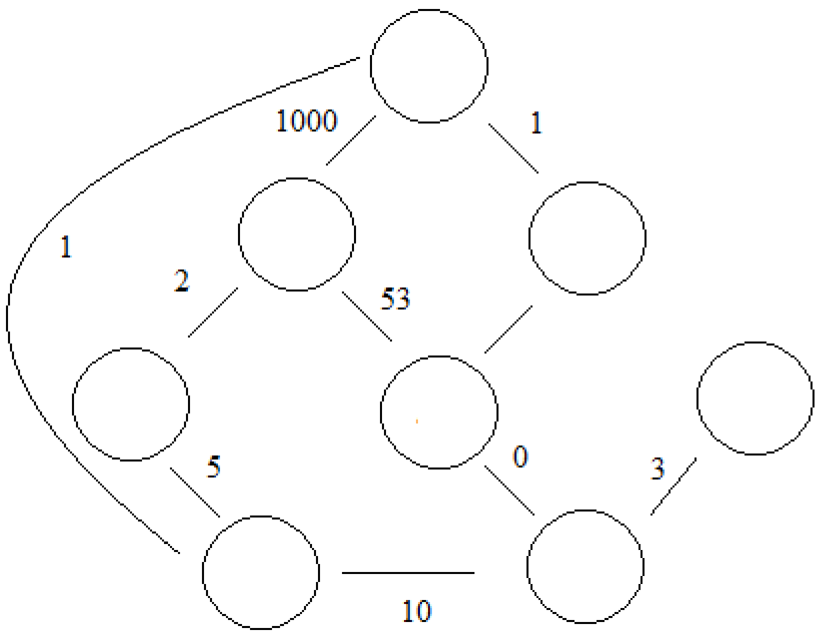


What we know

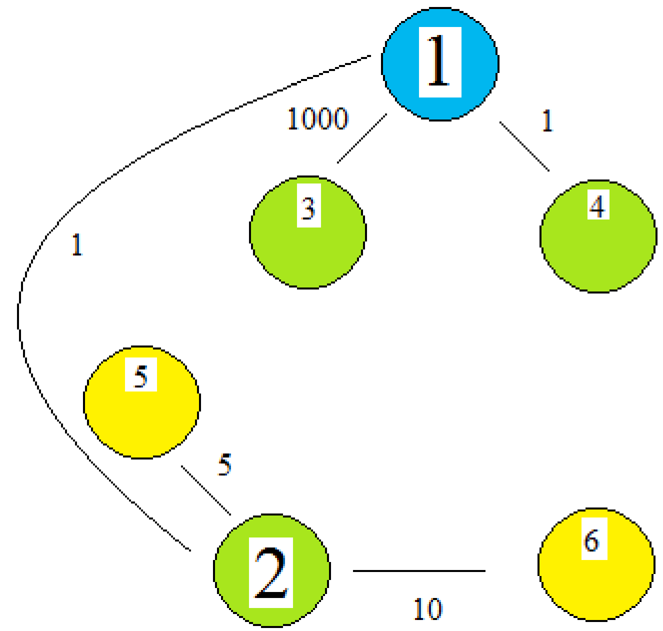


Uninformed Search Revisited: Breadth First Search

The whole graph

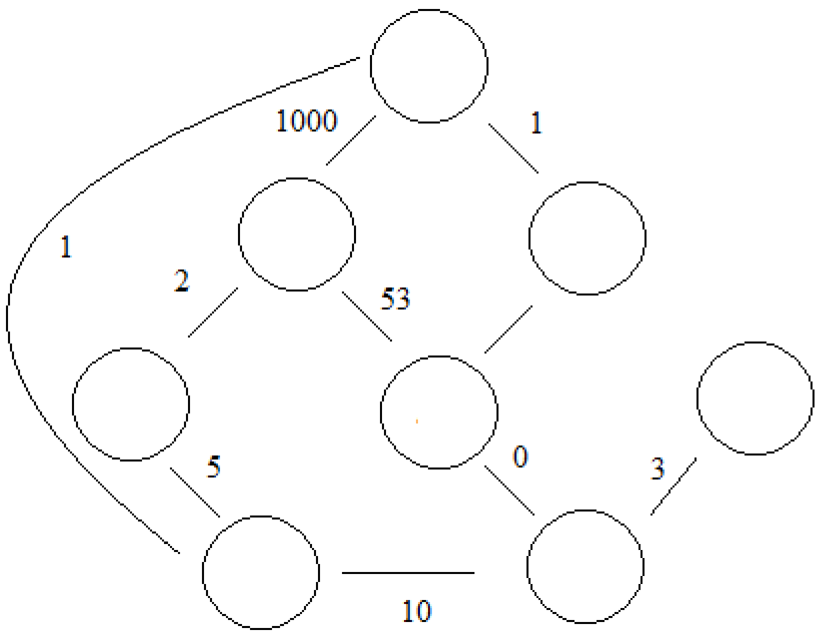


What we know

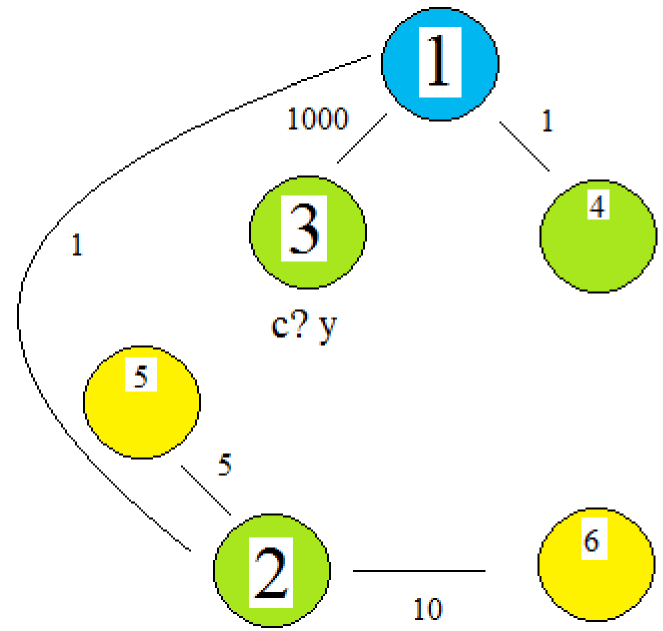


Uninformed Search Revisited: Breadth First Search

The whole graph

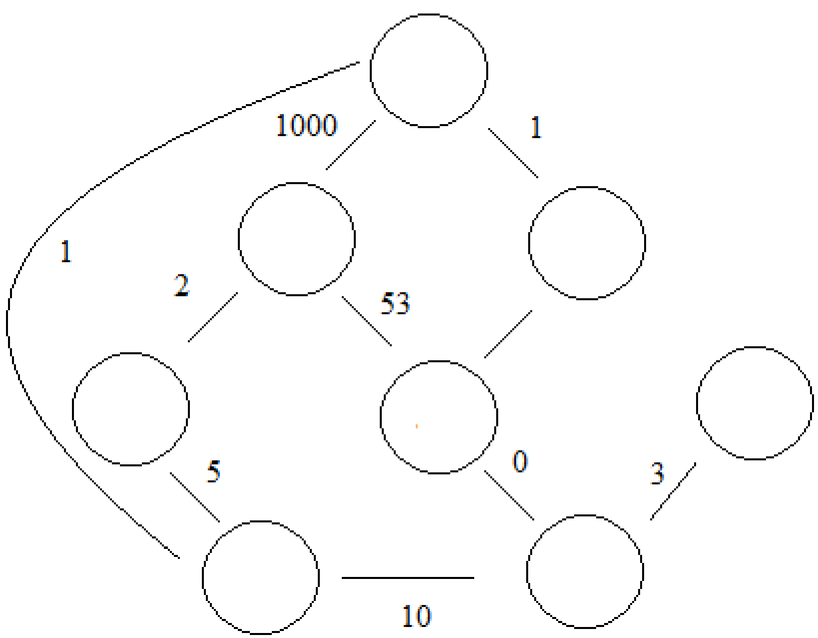


What we know

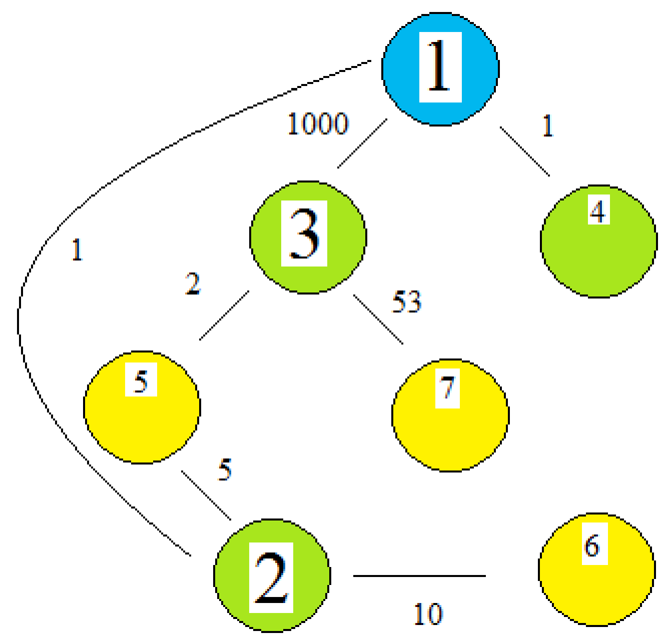


Uninformed Search Revisited: Breadth First Search

The whole graph

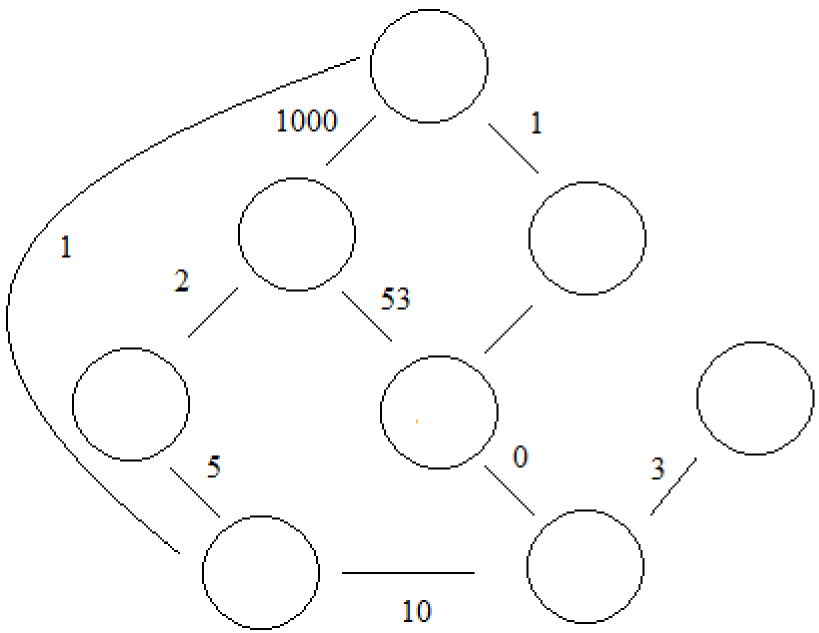


What we know

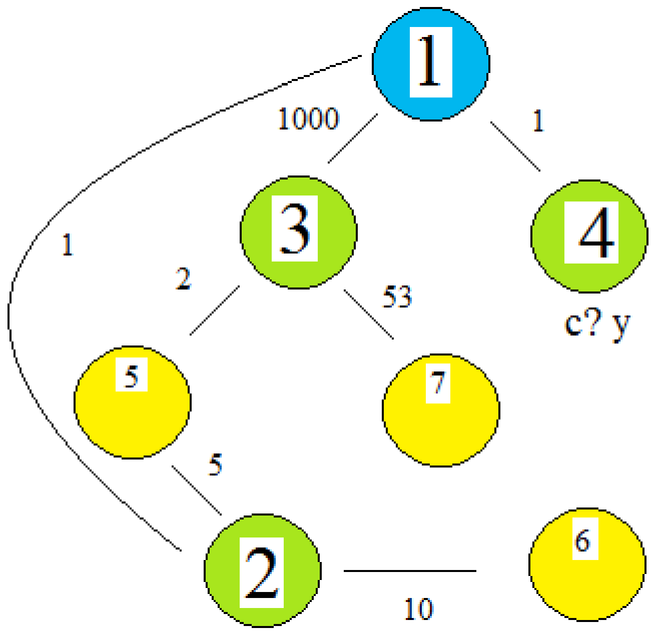


Uninformed Search Revisited: Breadth First Search

The whole graph

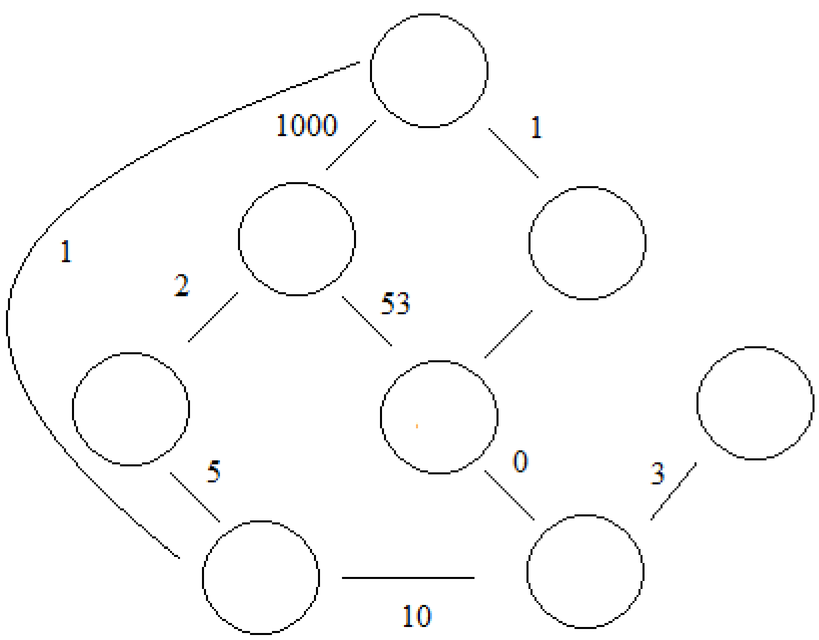


What we know

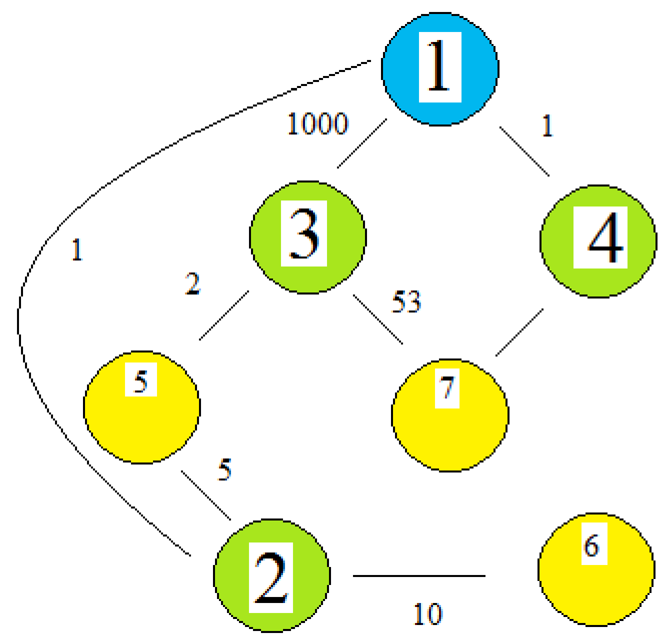


Uninformed Search Revisited: Breadth First Search

The whole graph

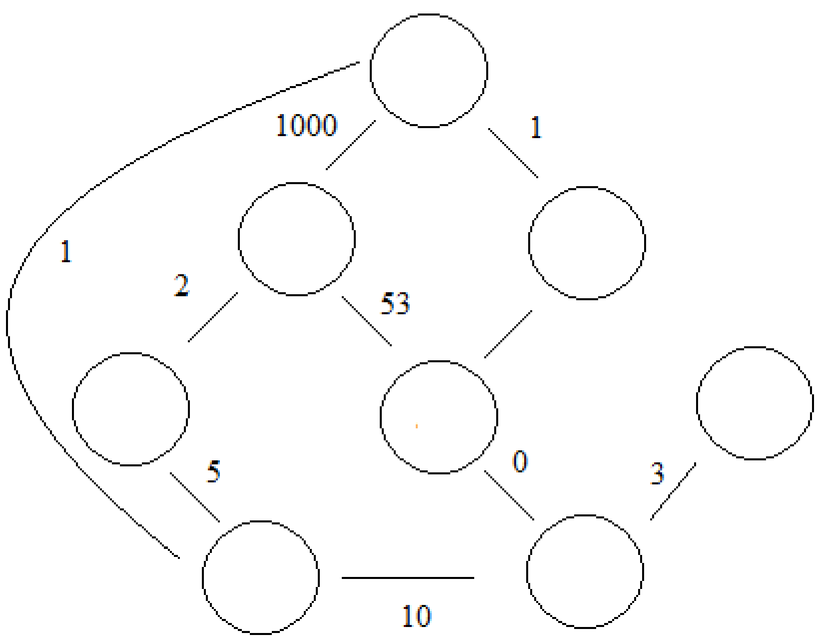


What we know

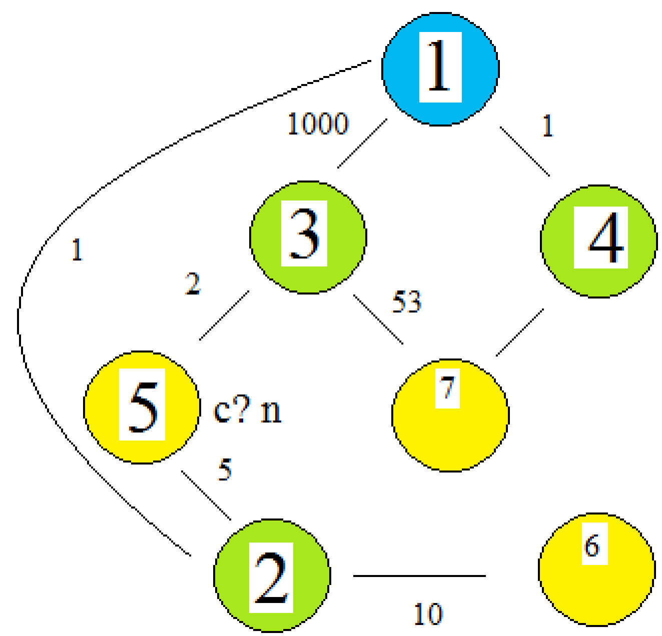


Uninformed Search Revisited: Breadth First Search

The whole graph

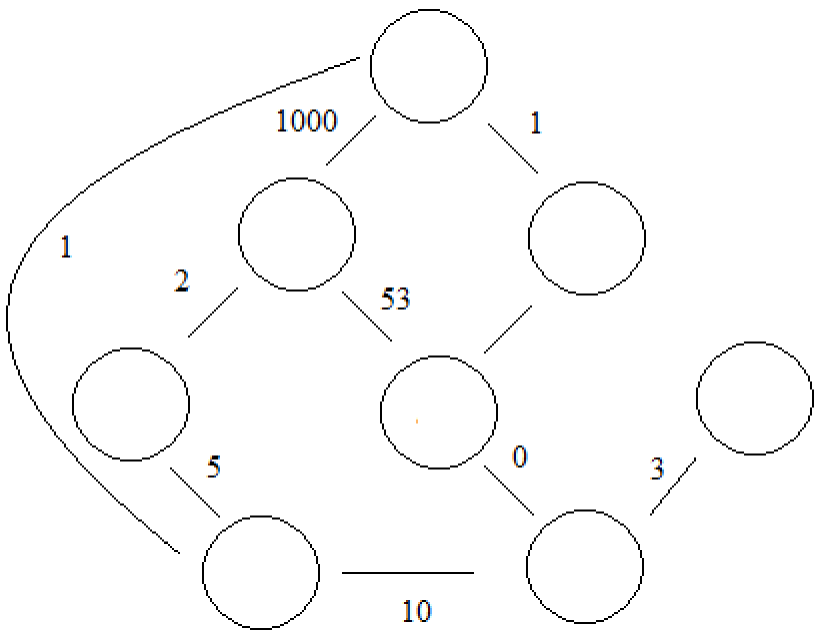


What we know

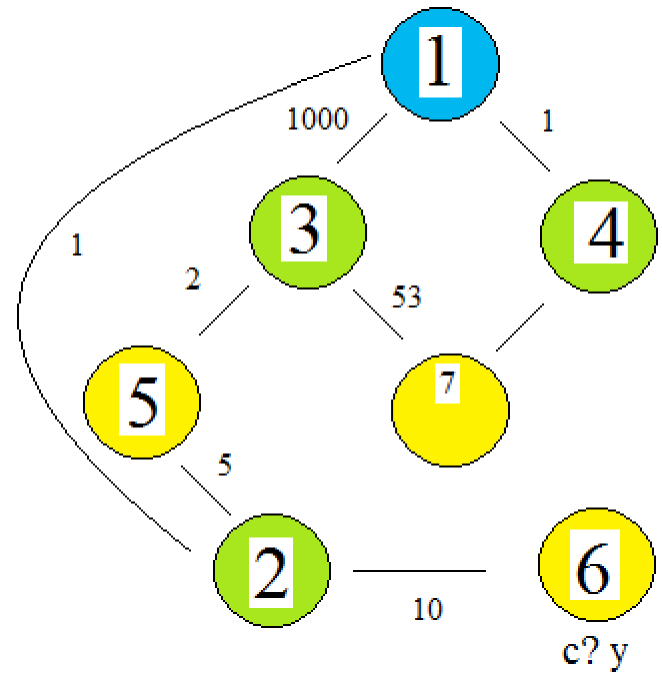


Uninformed Search Revisited: Breadth First Search

The whole graph

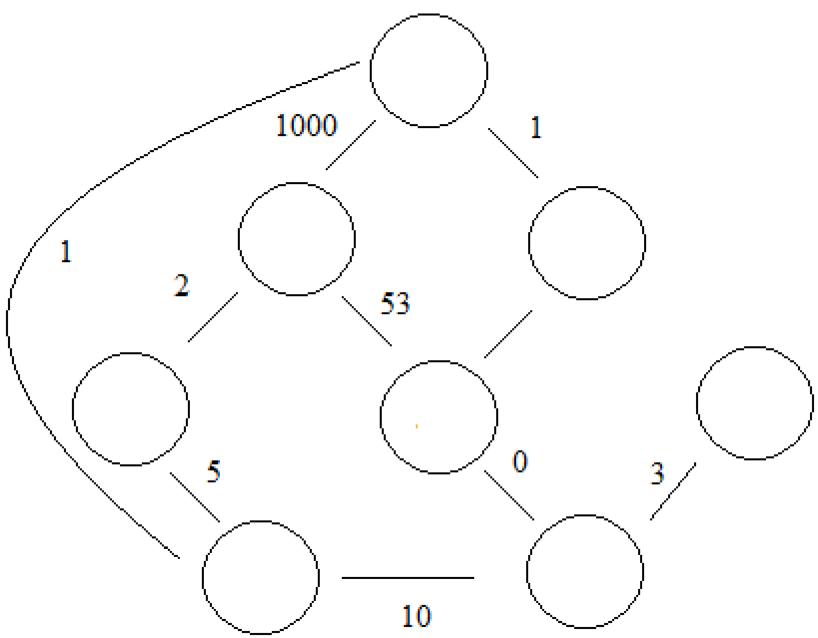


What we know

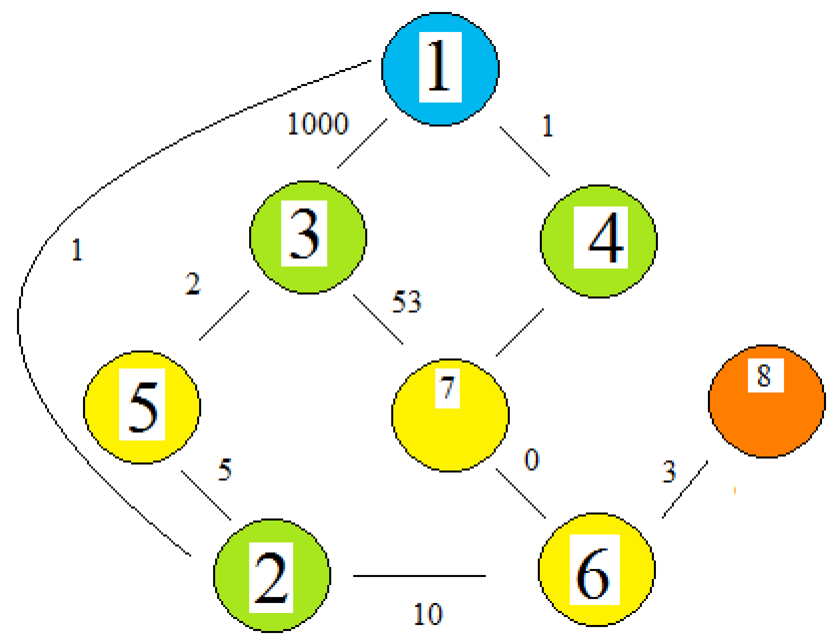


Uninformed Search Revisited: Breadth First Search

The whole graph

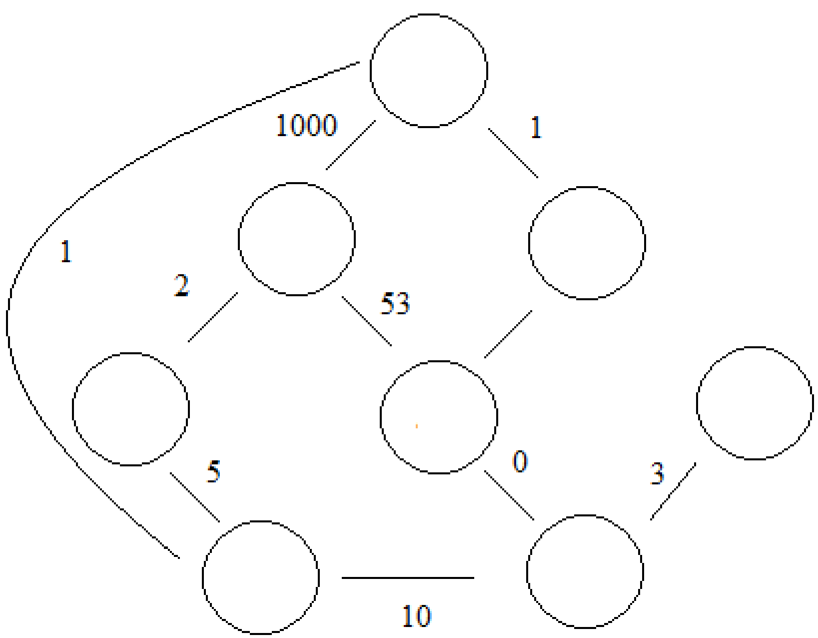


What we know

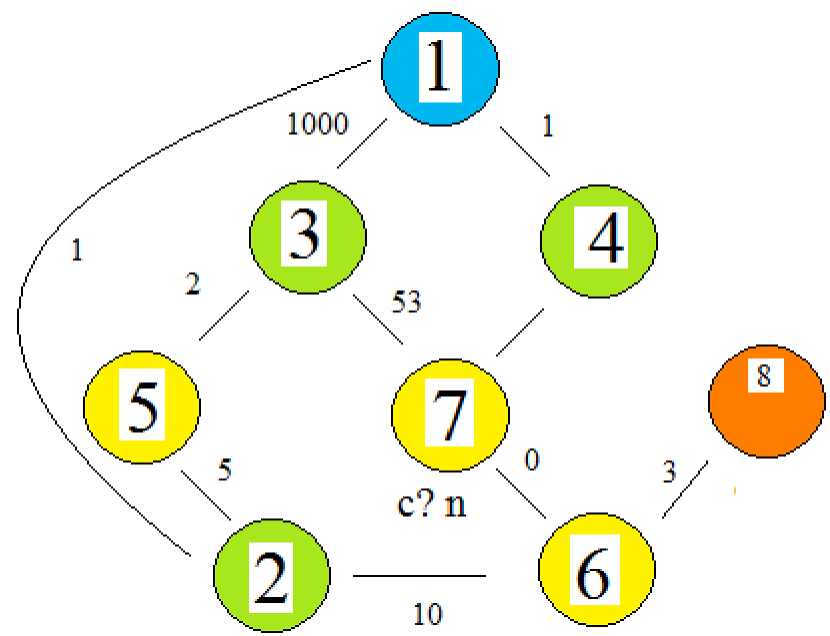


Uninformed Search Revisited: Breadth First Search

The whole graph

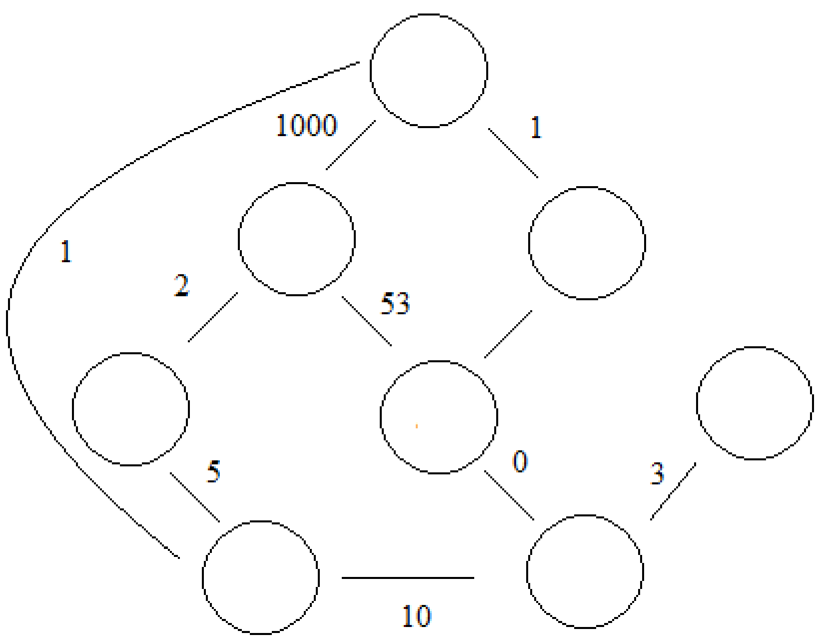


What we know

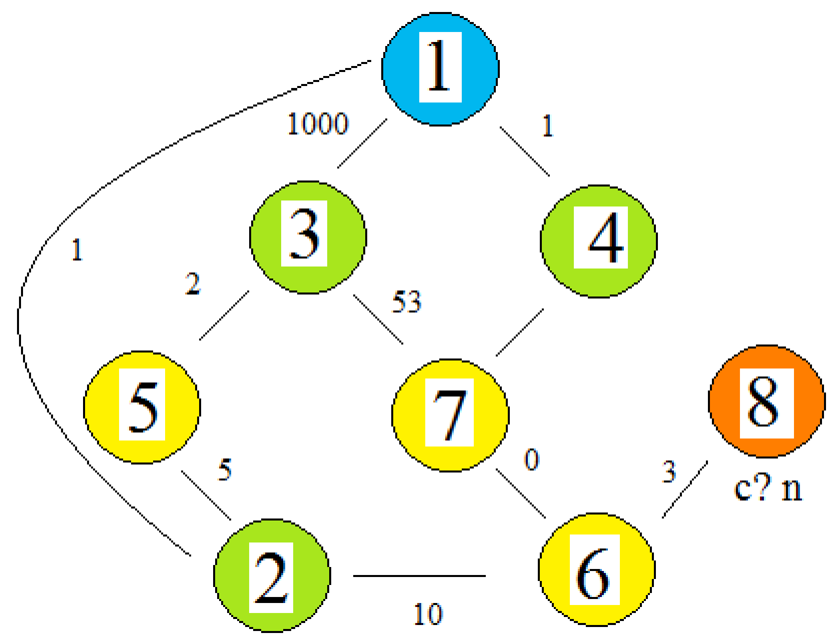


Uninformed Search Revisited: Breadth First Search

The whole graph

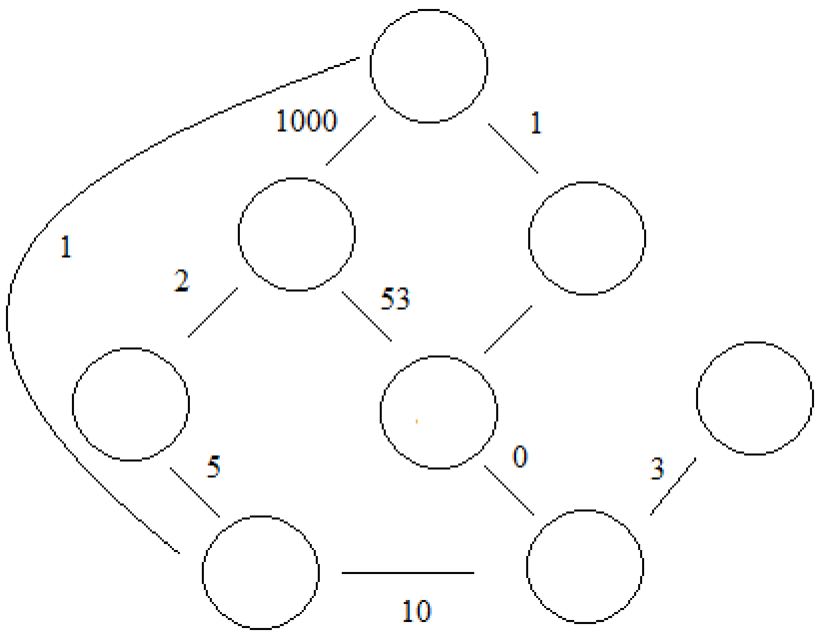


What we know

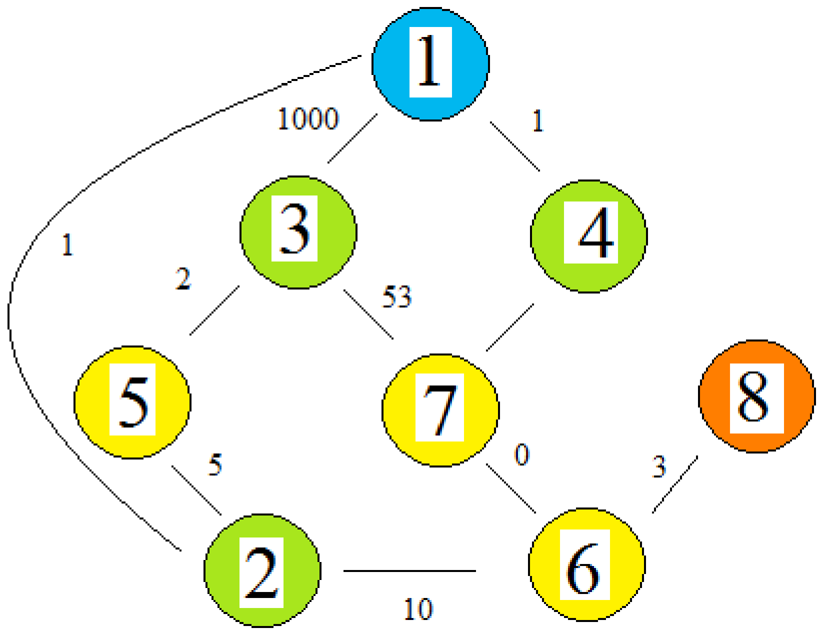


Uninformed Search Revisited: Breadth First Search

The whole graph



What we know



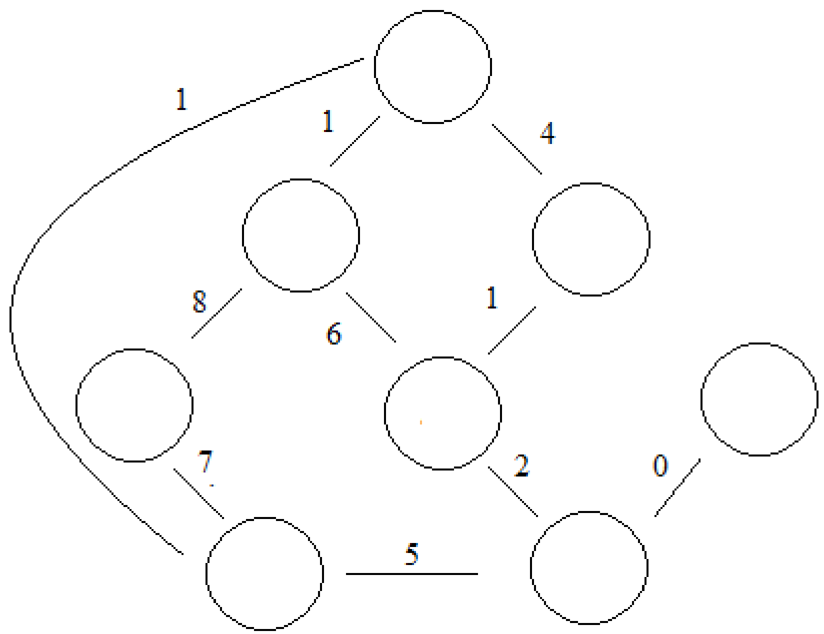
Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

- Fibonacci heap high level concepts (Wikipedia):
  - A forest of trees
    - The trees are arranged in a ring (doubly linked list)
    - Each tree is a doubly linked list
    - Always keep a pointer to the over all minimum
  - Lazy operations (some things are saved for later)
    - 'Merge' just combines forests into a larger ring
    - 'extract min' basically merges trees along the way
    - 'decrease key' makes new trees along the way
    - 'delete' sets the node's value to -inf then extracts min
  - Special constraints:
    - Size of a subtree rooted at node of degree k is at least F_{k+2}, where F_k is the kth Fibonacci number.
    - In their time analysis they have a notion of saving time for later.

Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

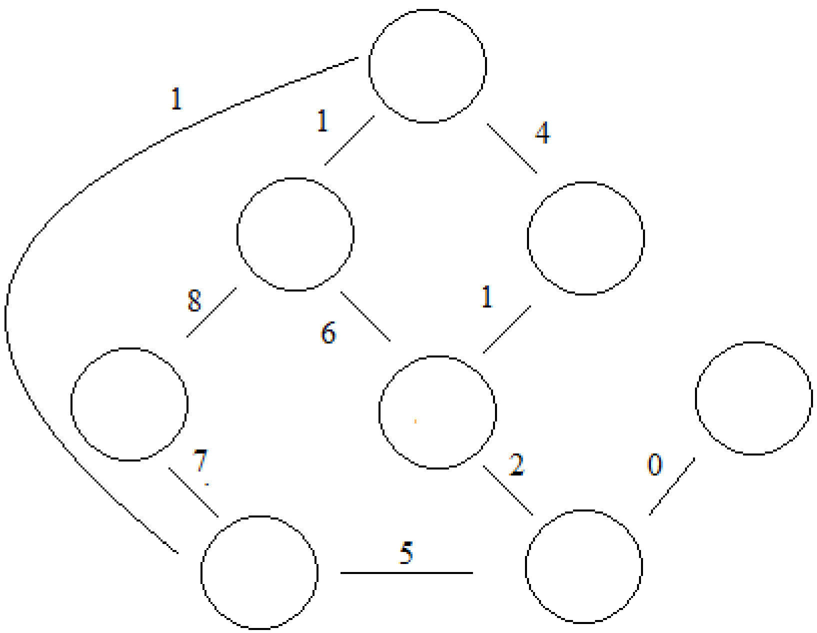
- Another way of thinking about it:
  - Always expand the node that is the closest to the start based on cost.
  - When you expand a node, save the back pointer along the cheapest path.
  - If during an expansion you find a cheaper way to get to a child (neighbor), update the child so that it knows about the cheapest way back to the start.

Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

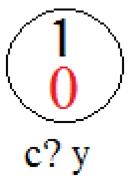


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

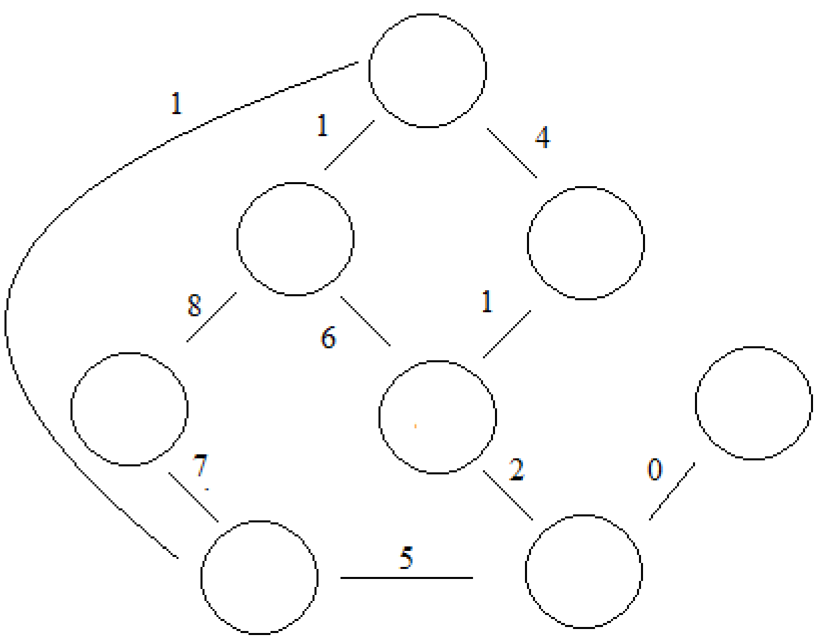


What we know

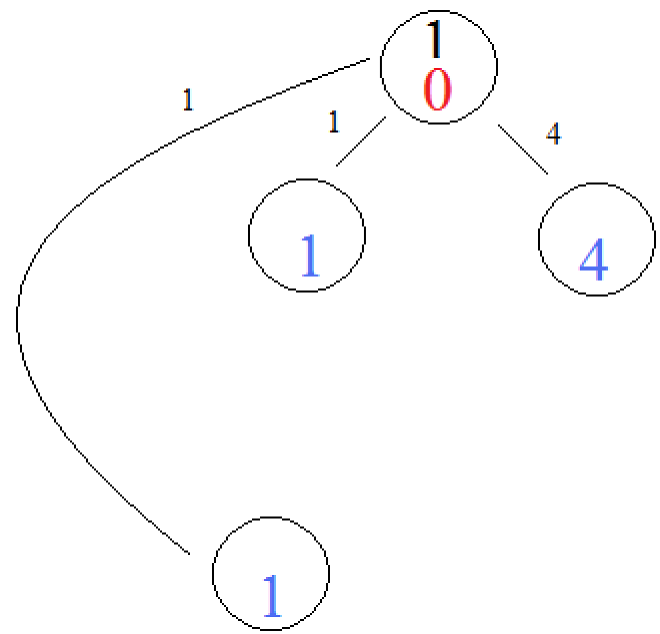


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

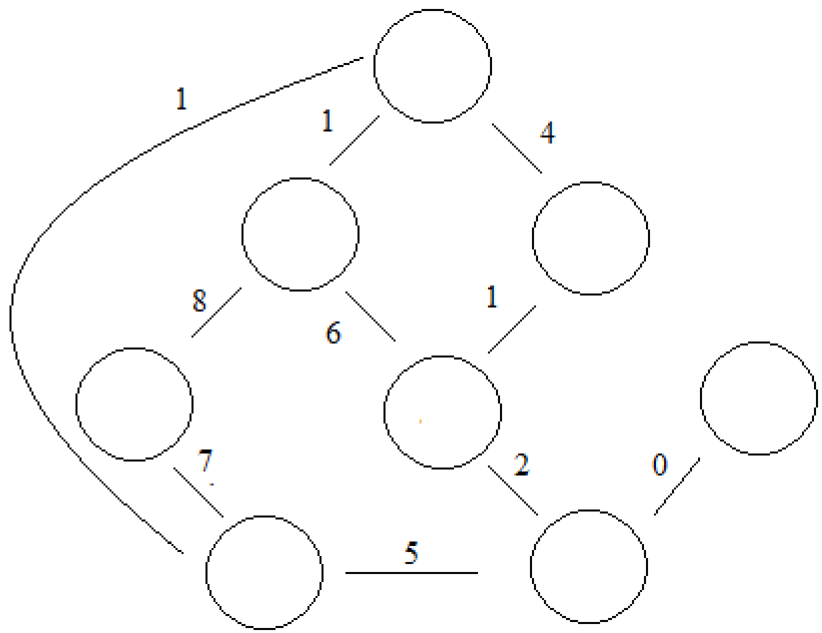


What we know

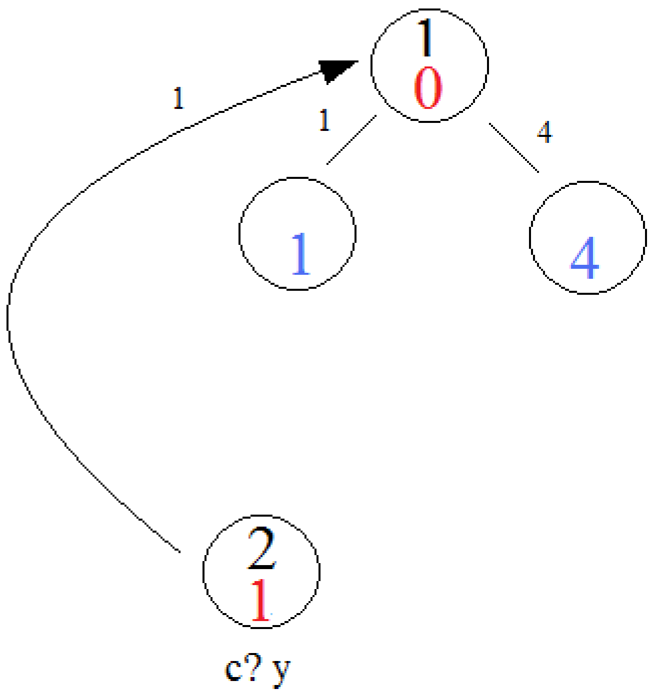


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

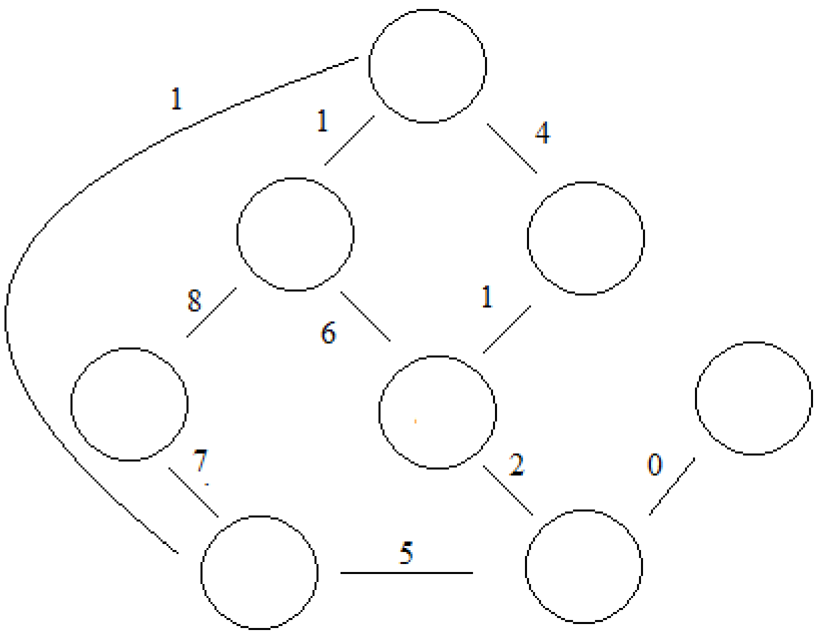


What we know

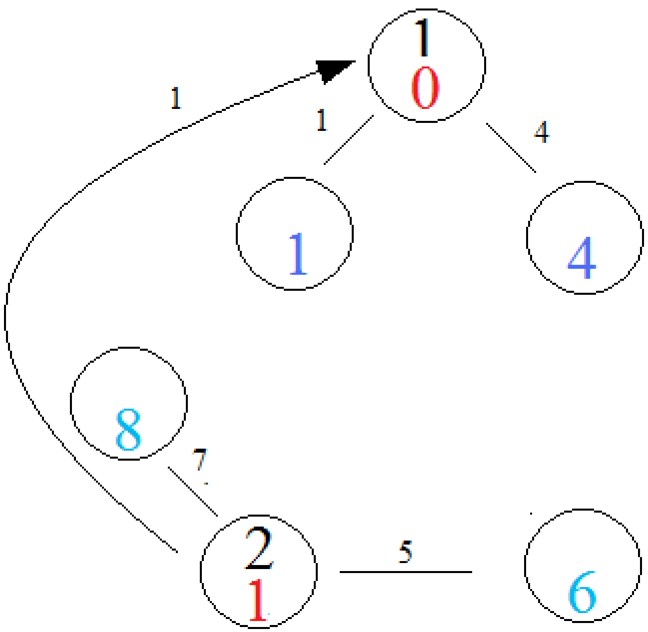


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

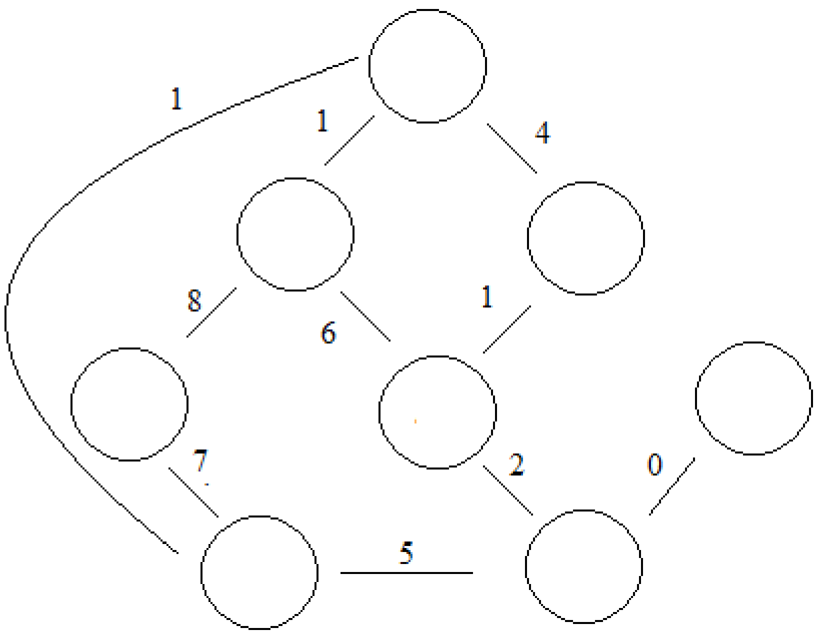


What we know

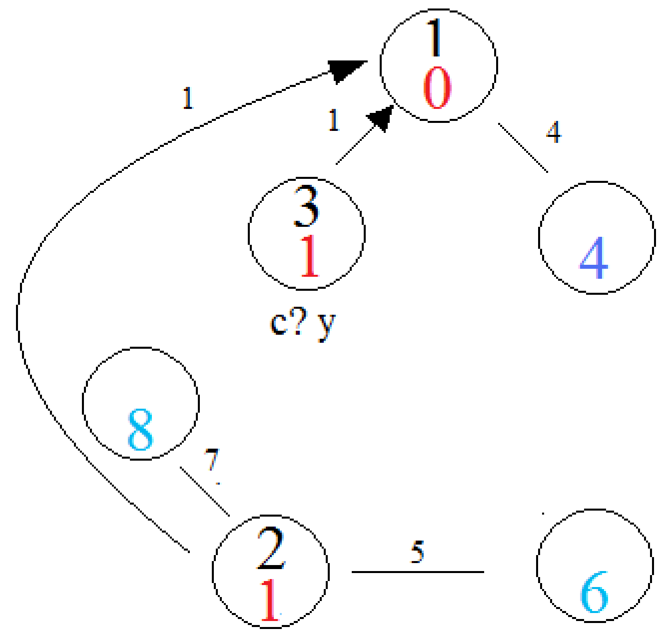


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

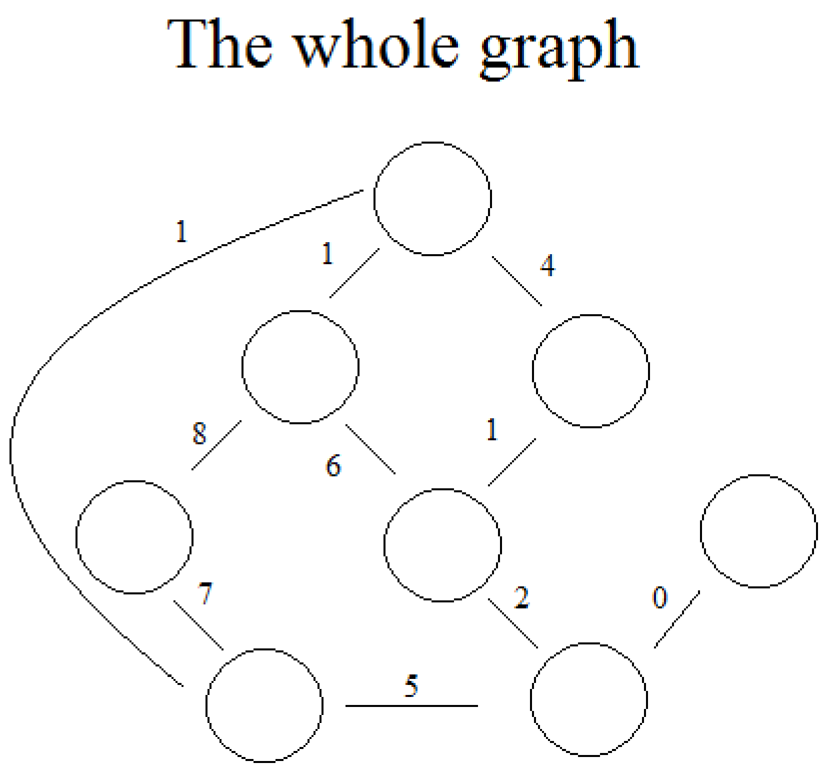
The whole graph

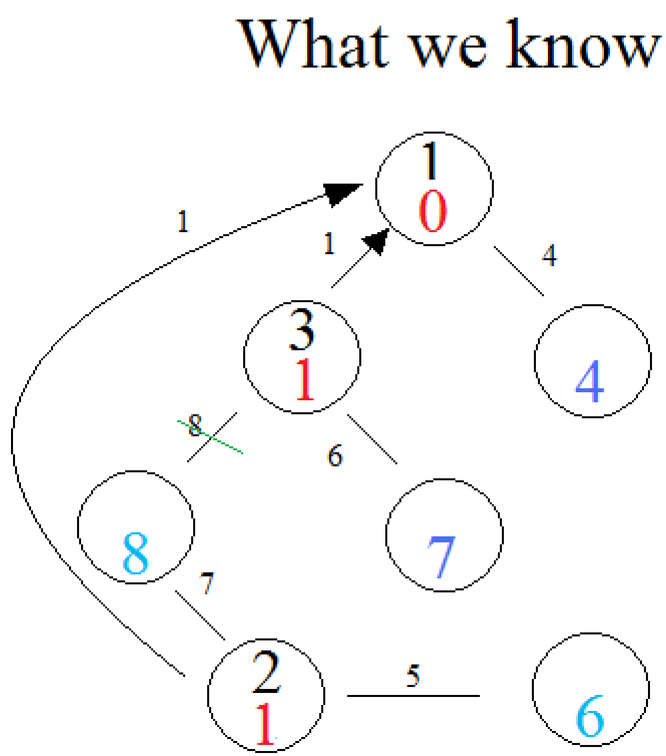


What we know



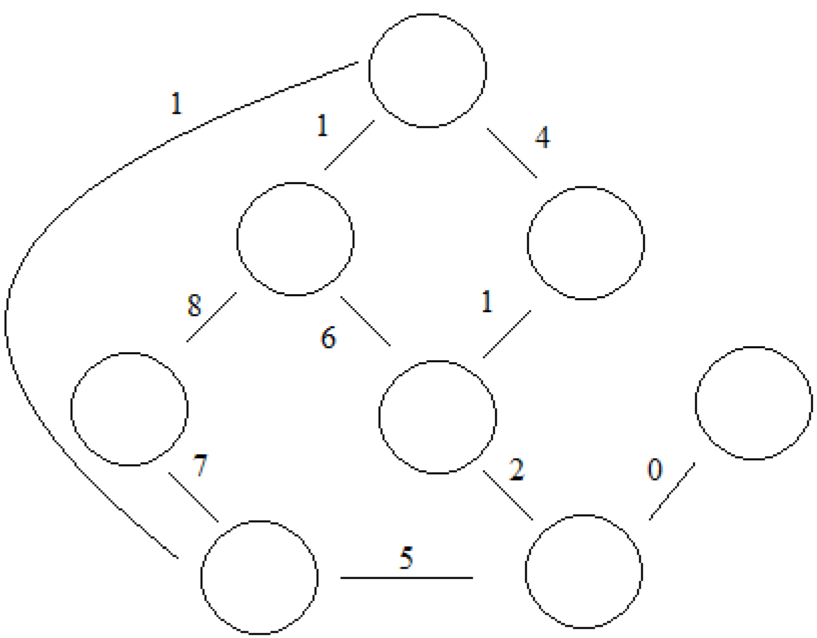
Uninformed Search Revisited: Dijkstra's (least-cost-first) Search



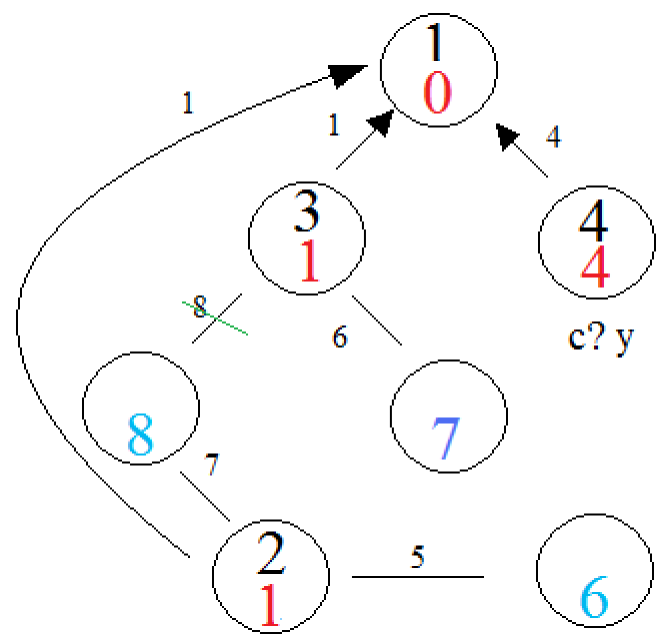


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

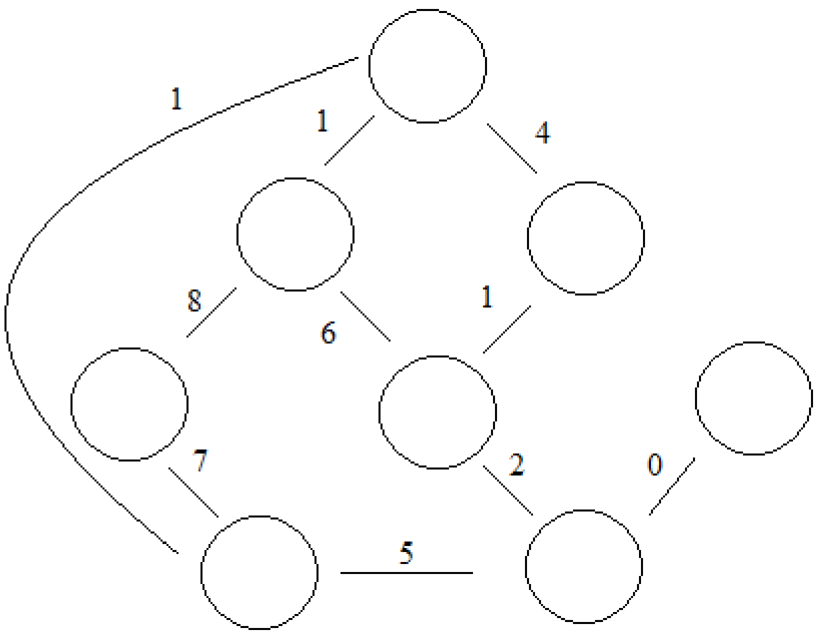


What we know

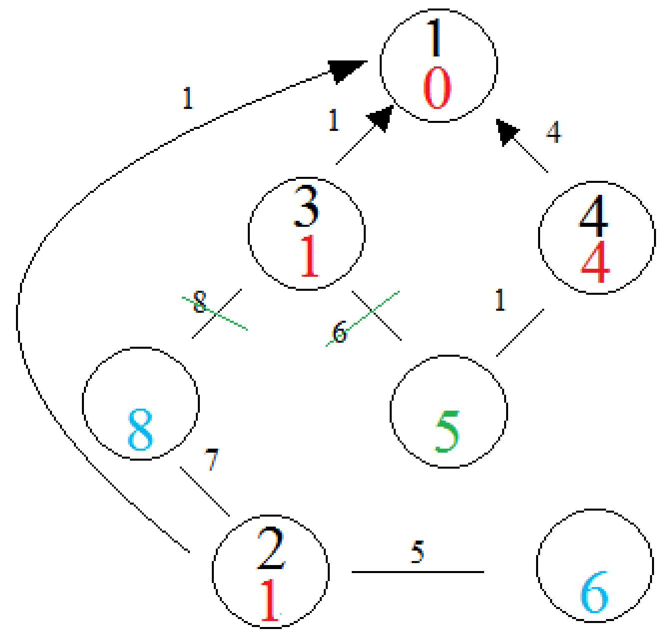


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

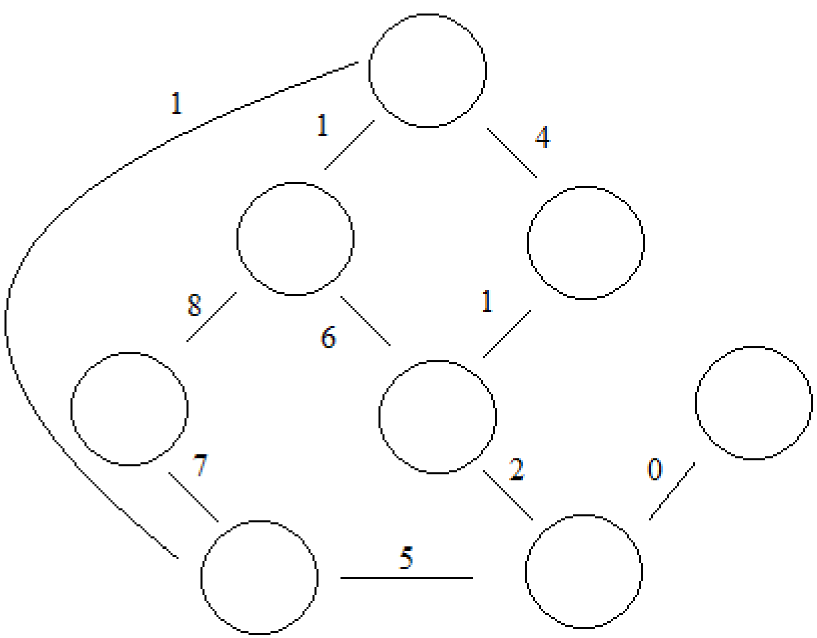


What we know

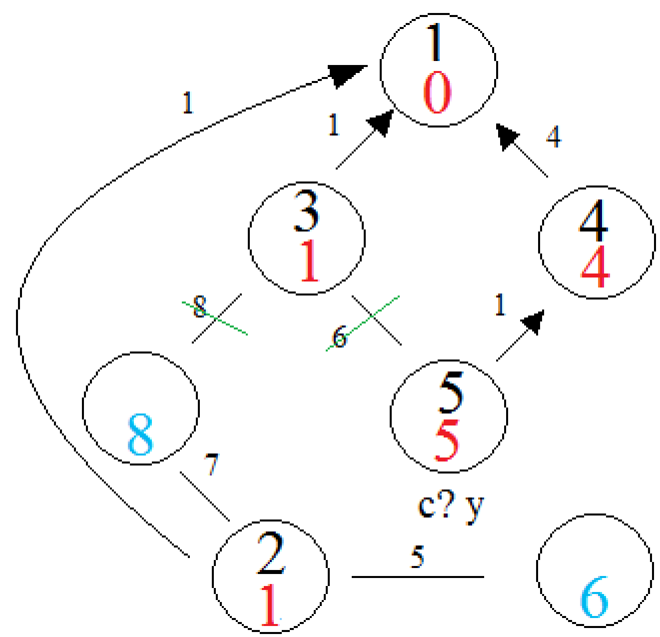


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

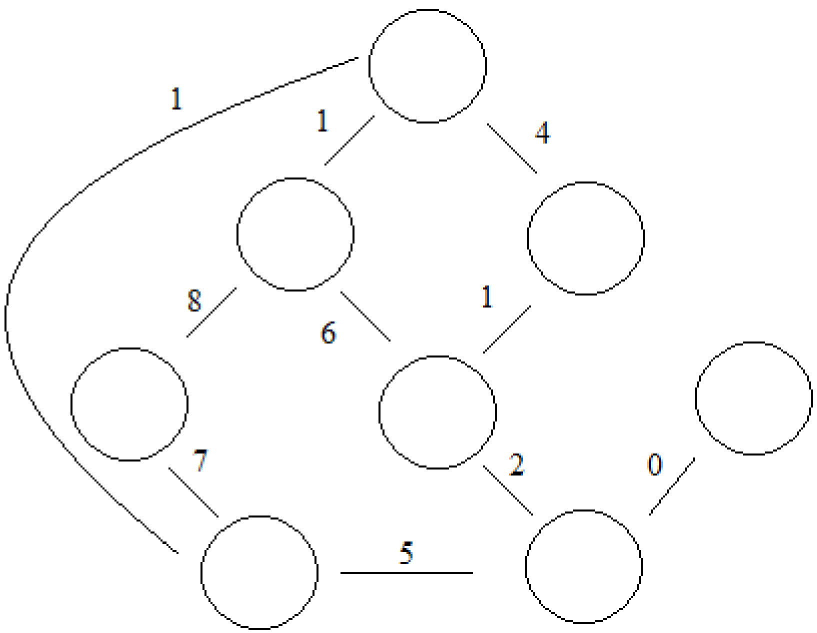


What we know

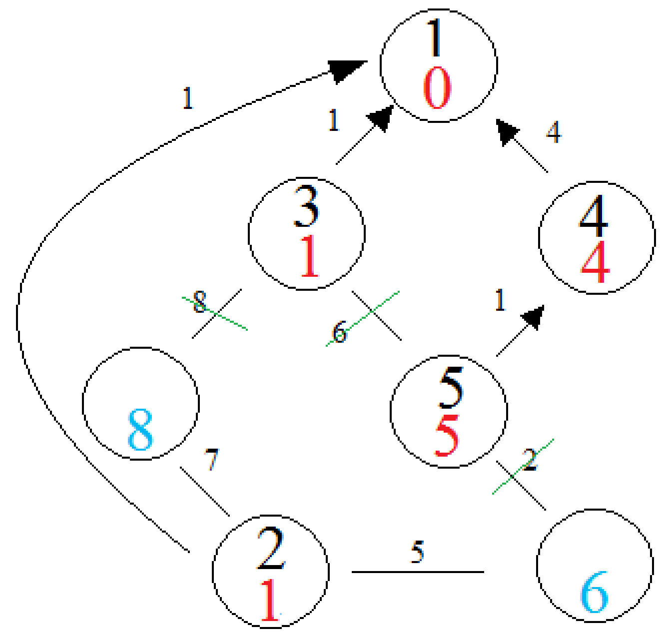


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

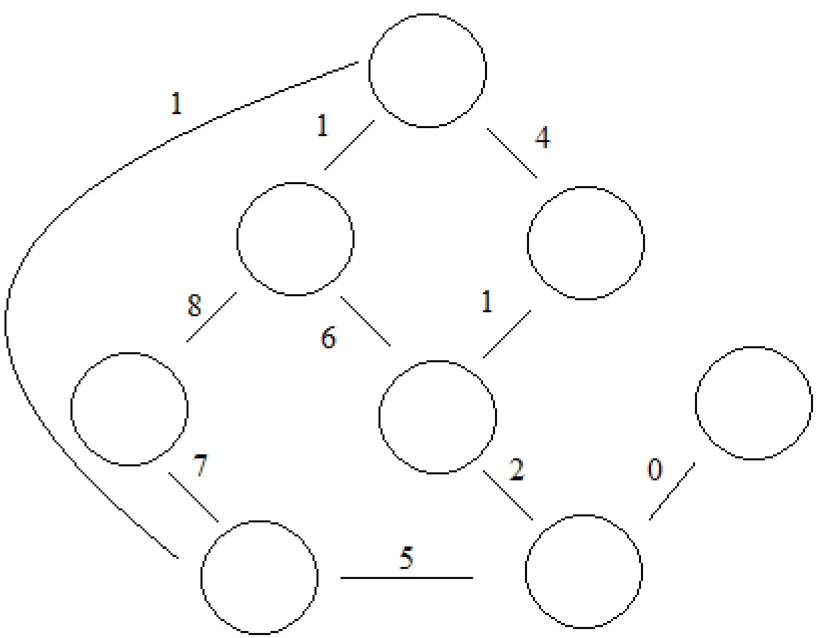


What we know

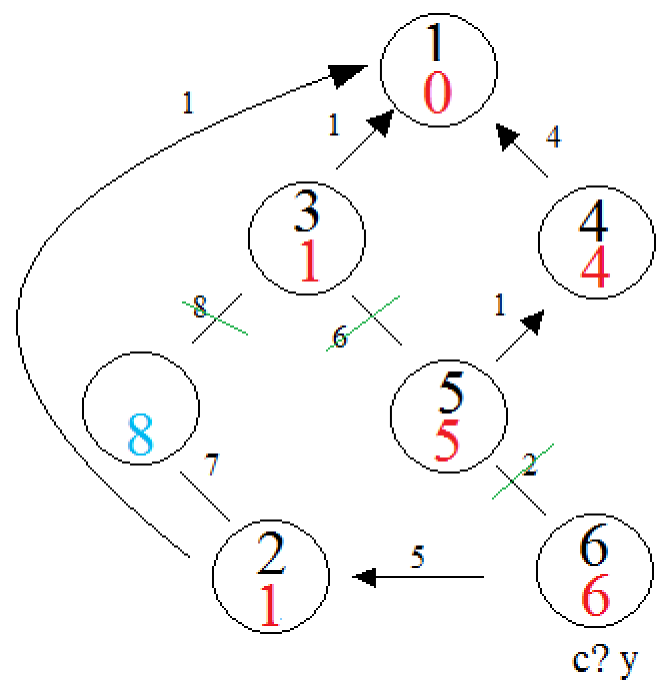


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

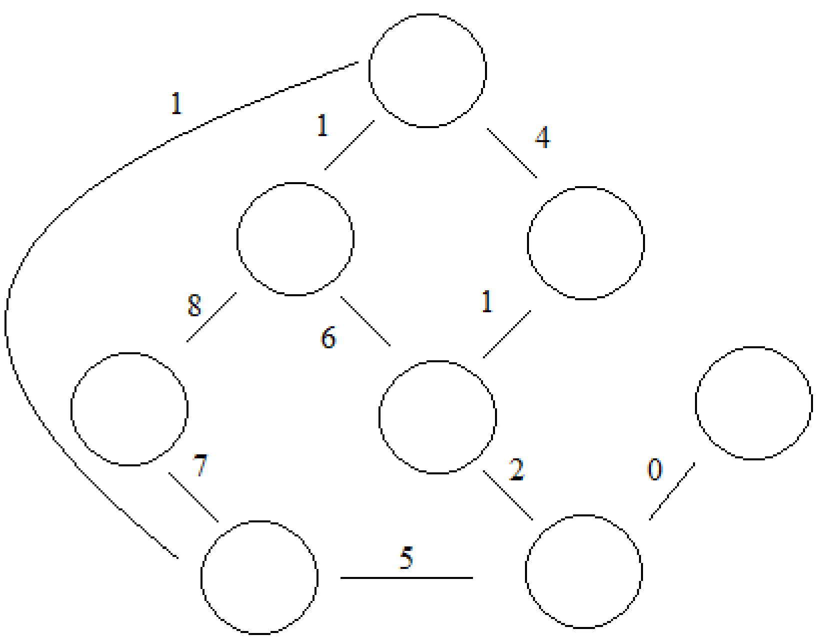


What we know

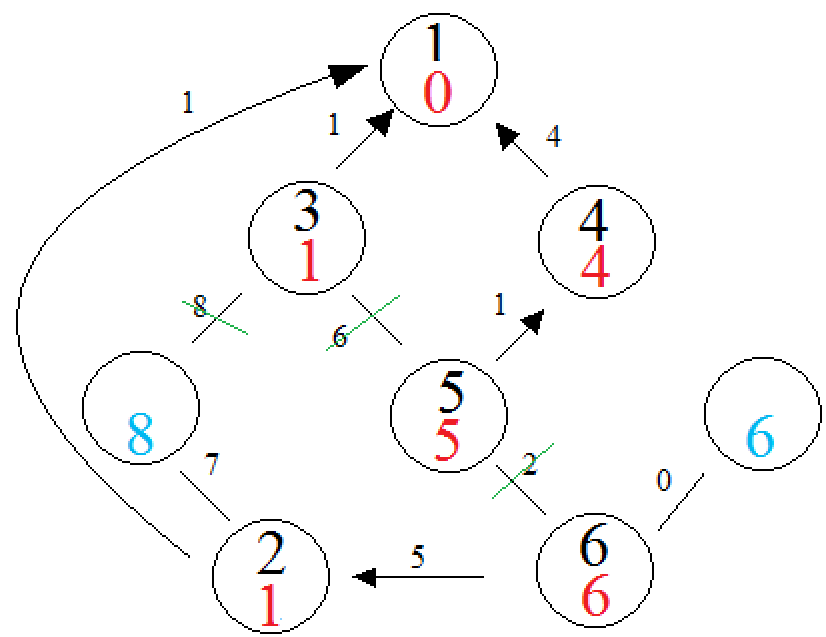


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

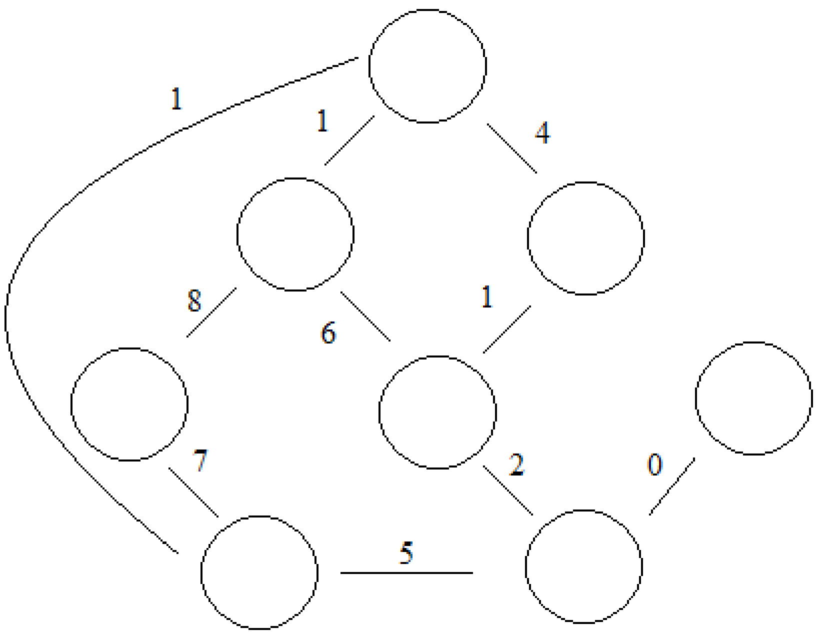


What we know

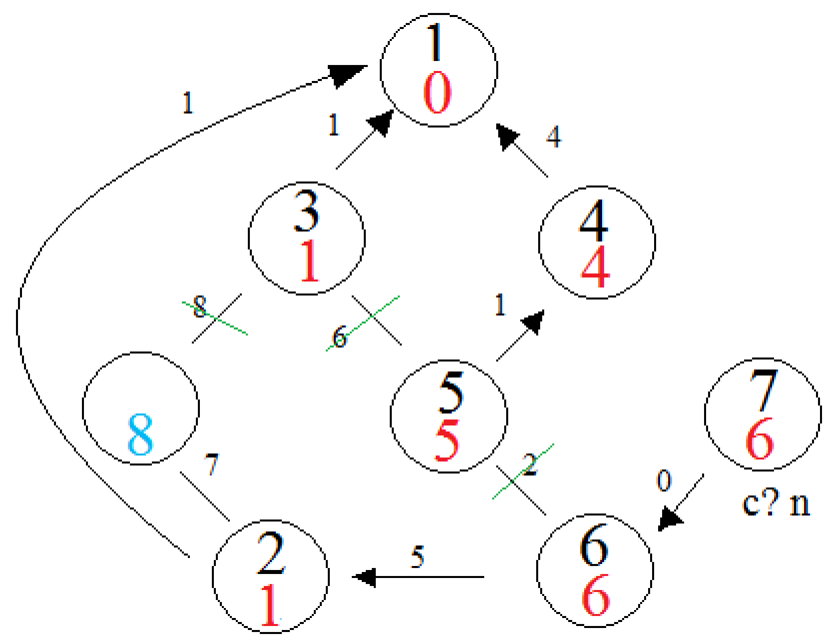


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

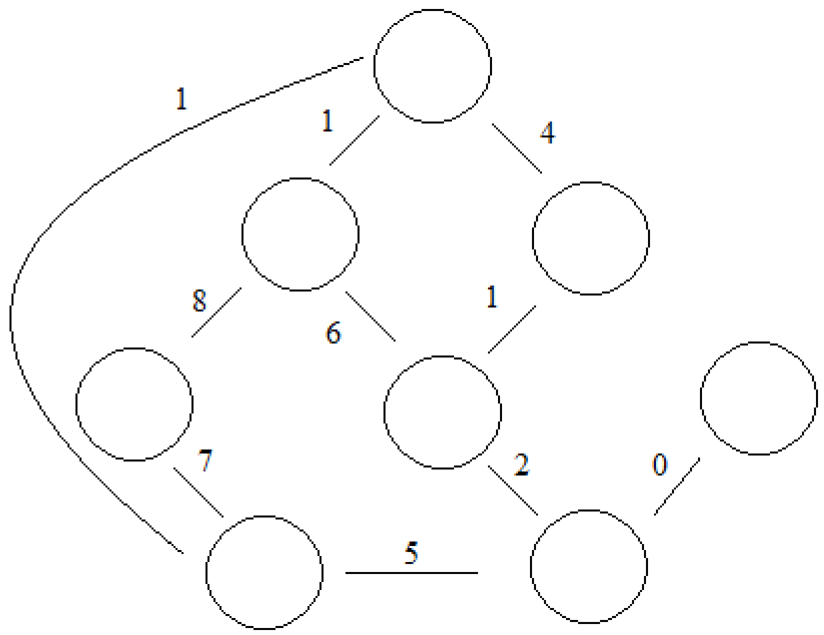


What we know

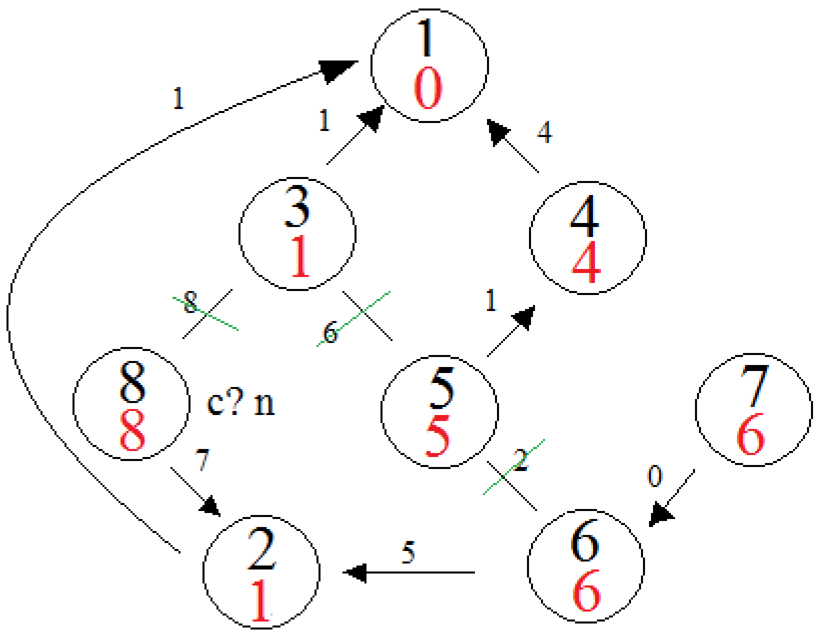


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

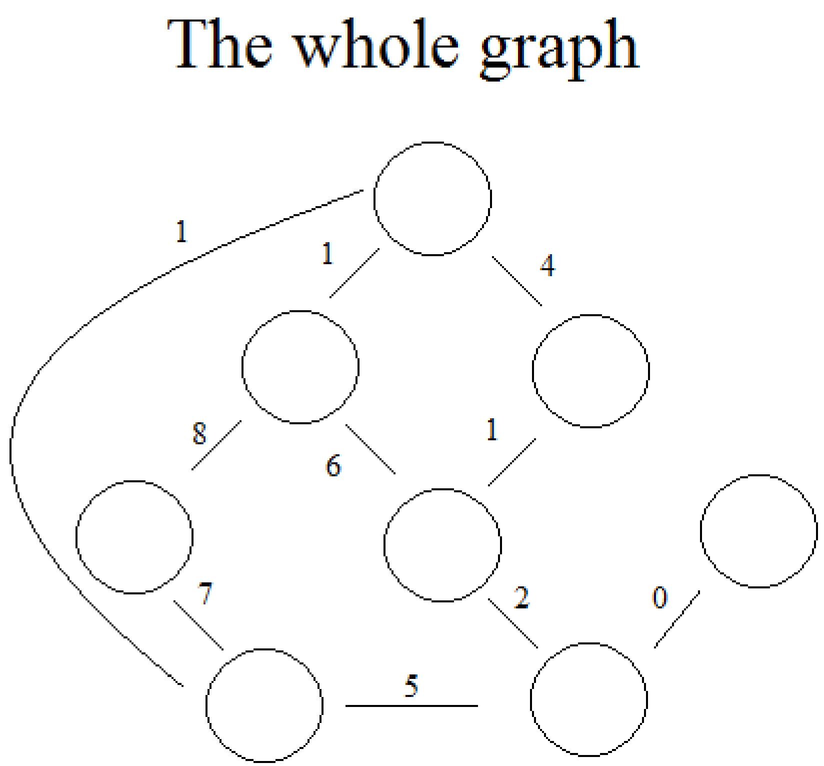
The whole graph

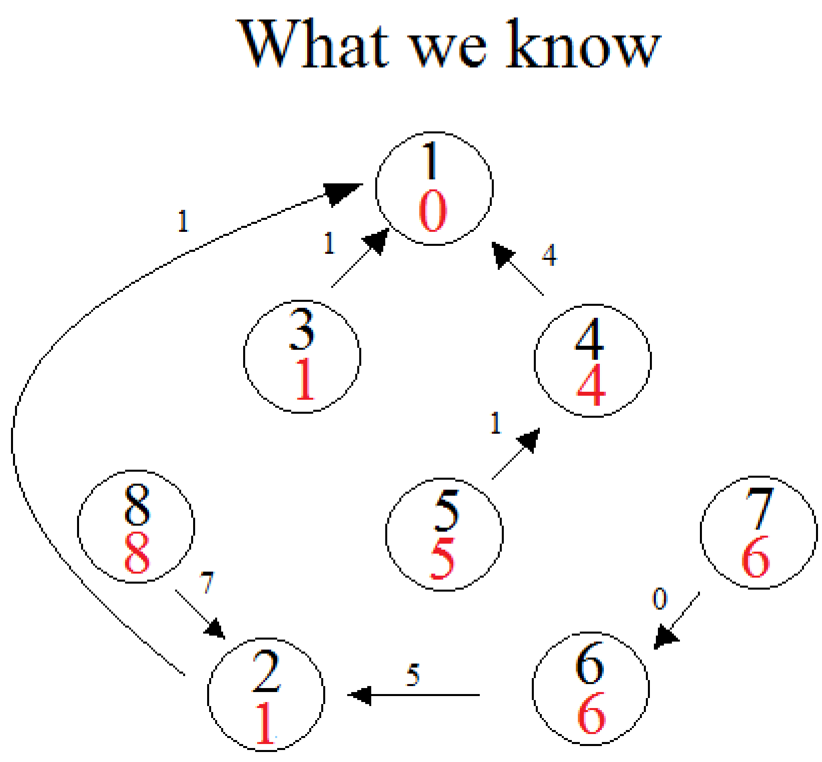


What we know

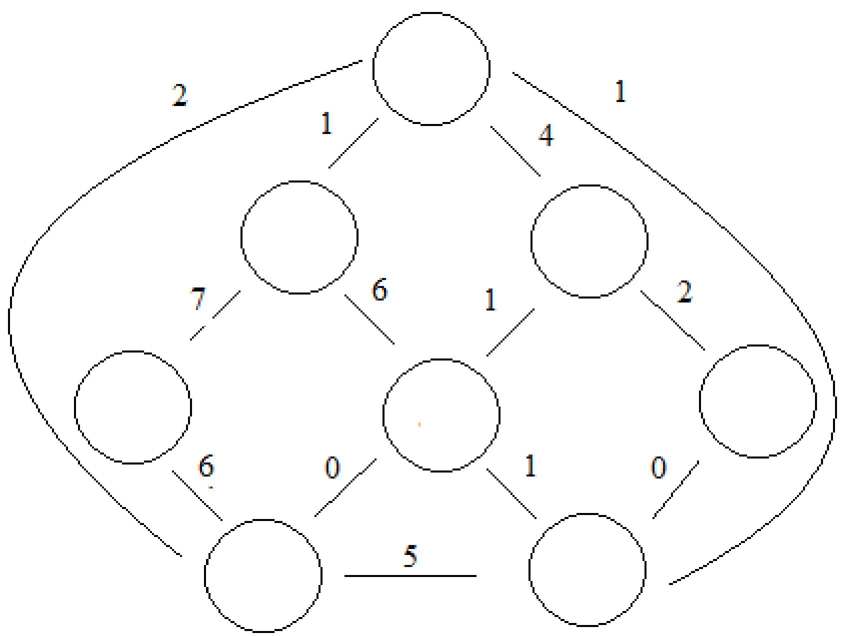


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search



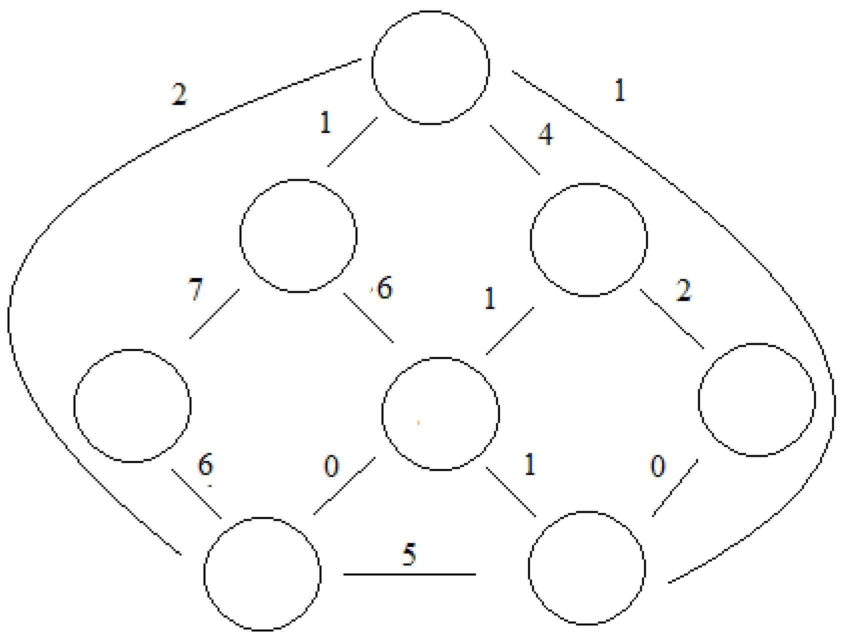


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

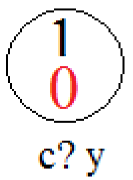


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

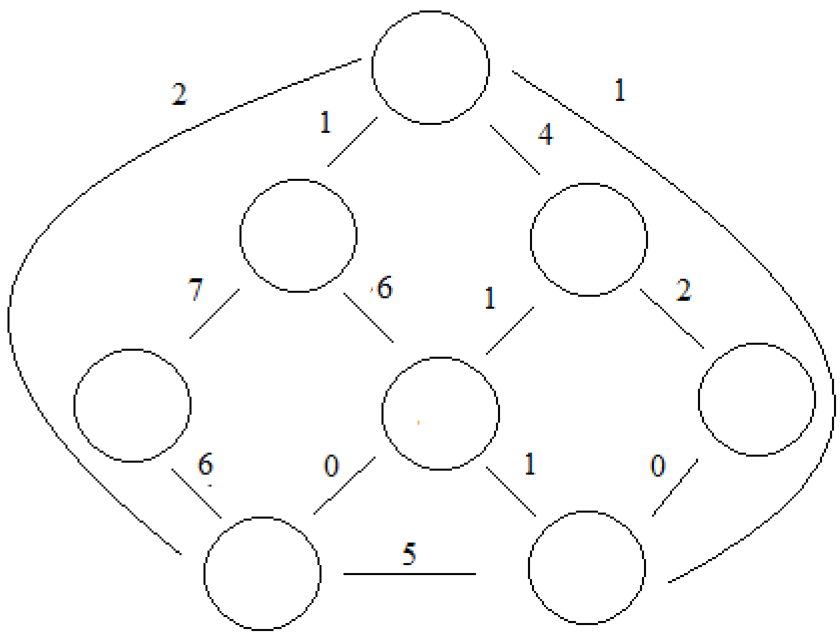


What we know

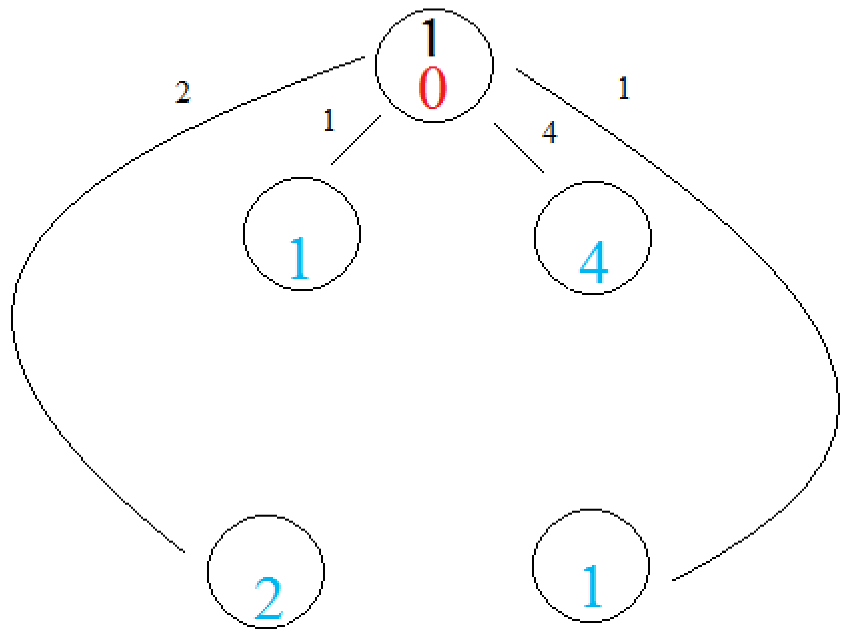


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

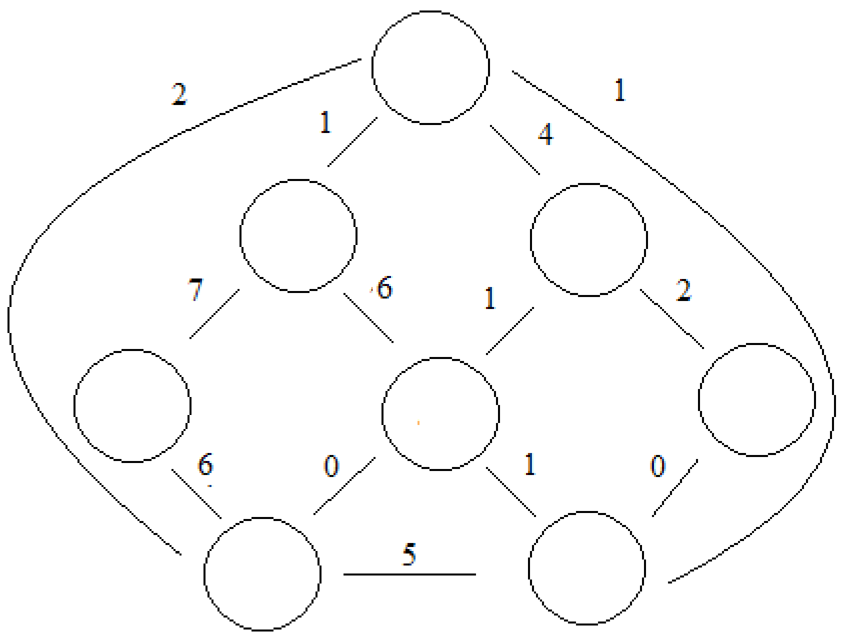


What we know

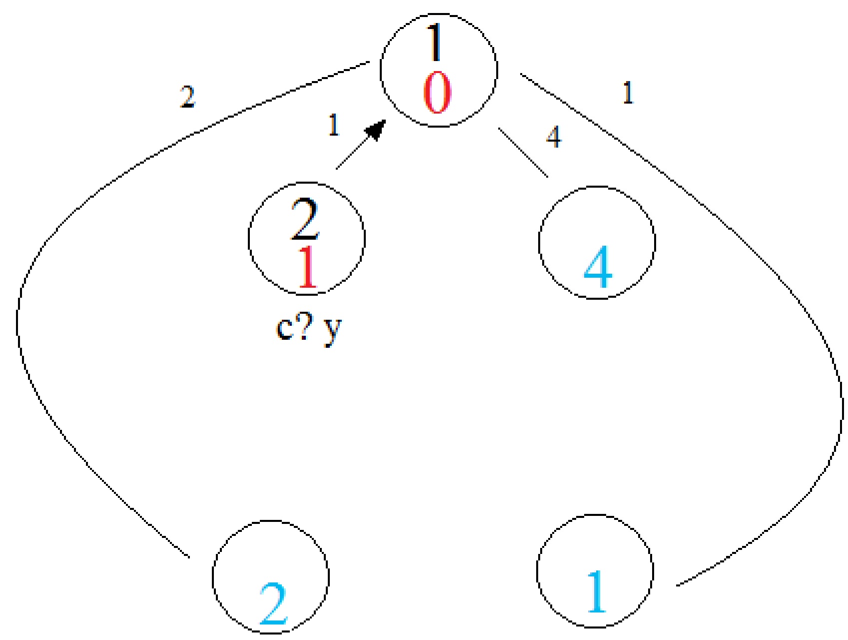


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

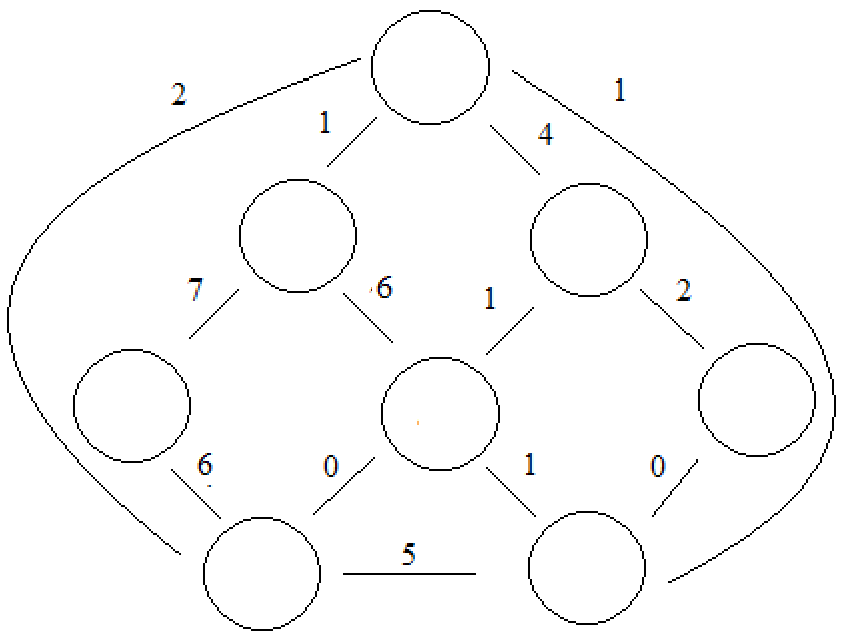


What we know

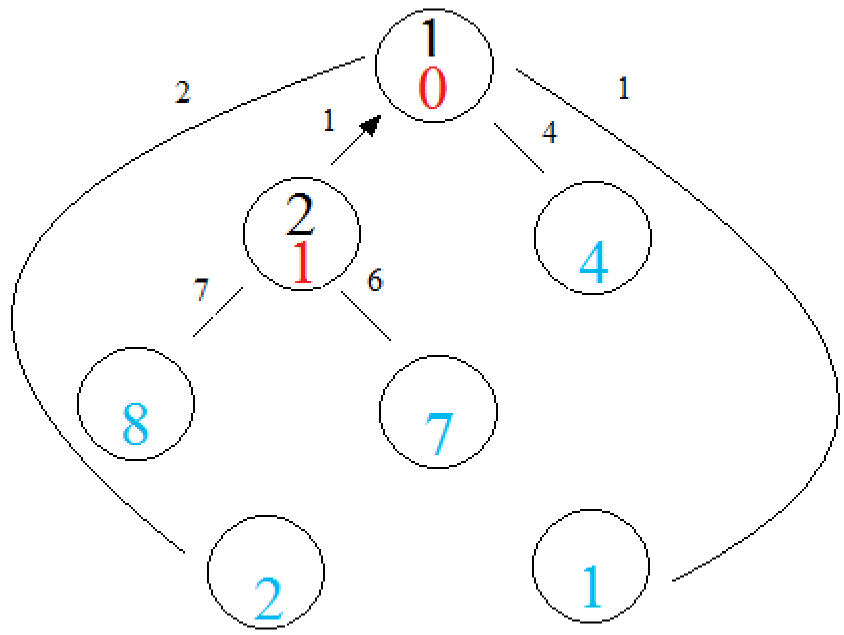


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

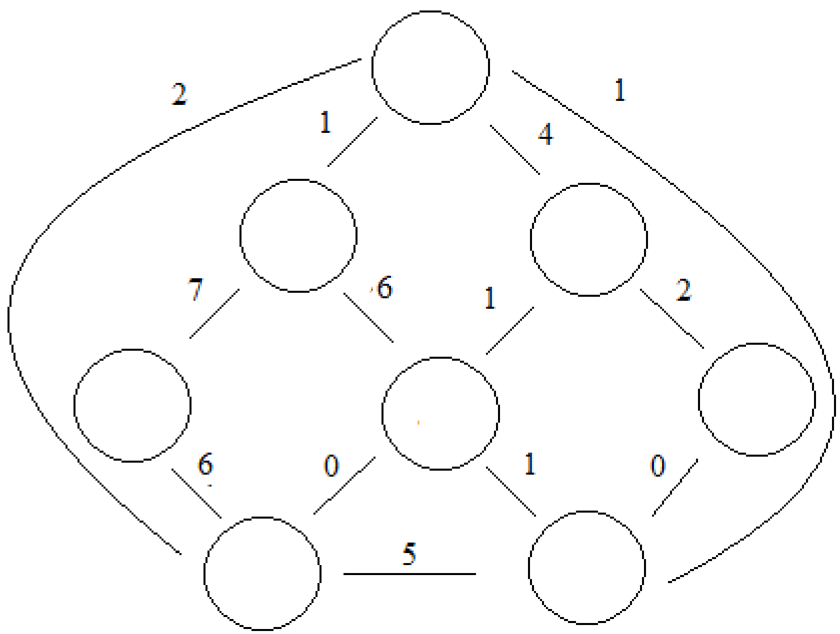


What we know

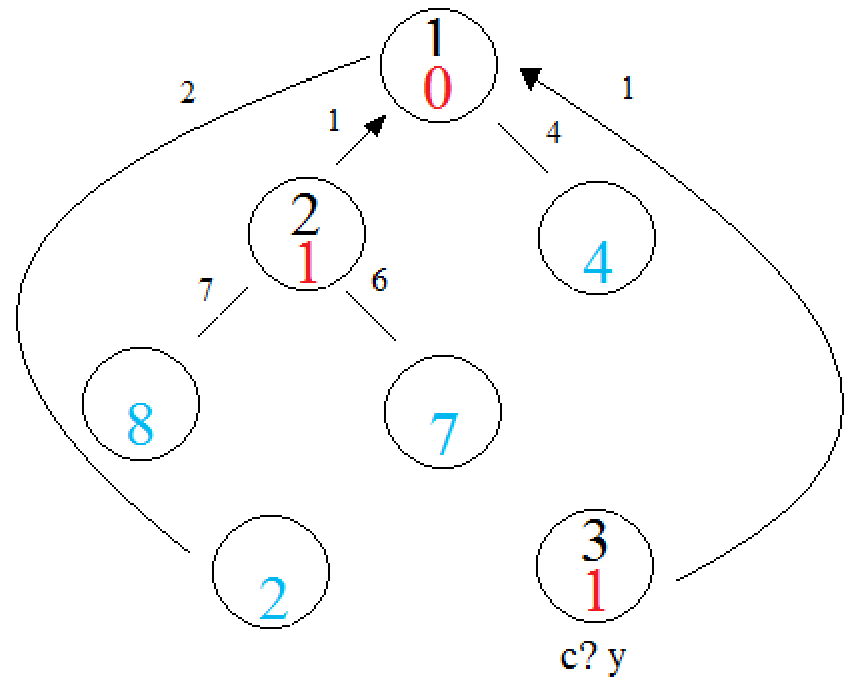


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

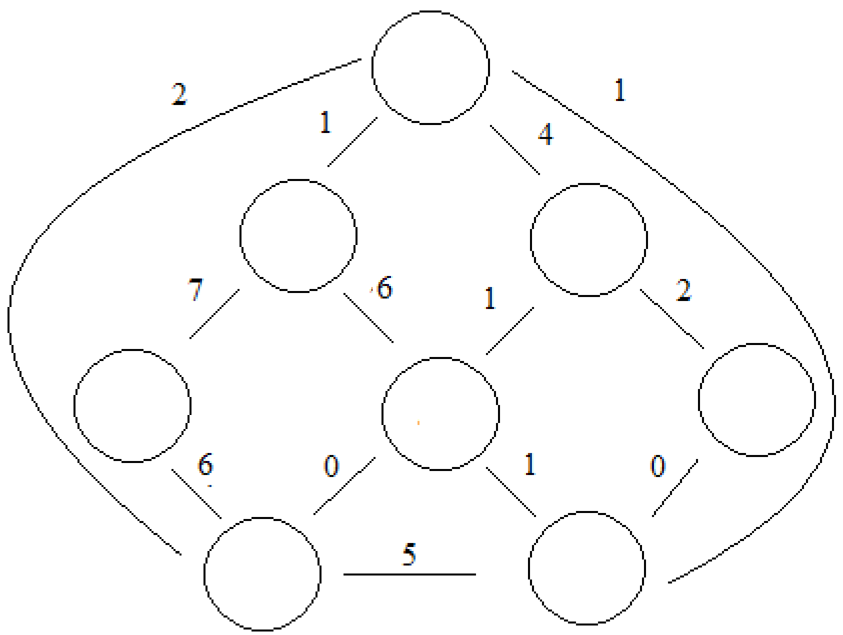


What we know

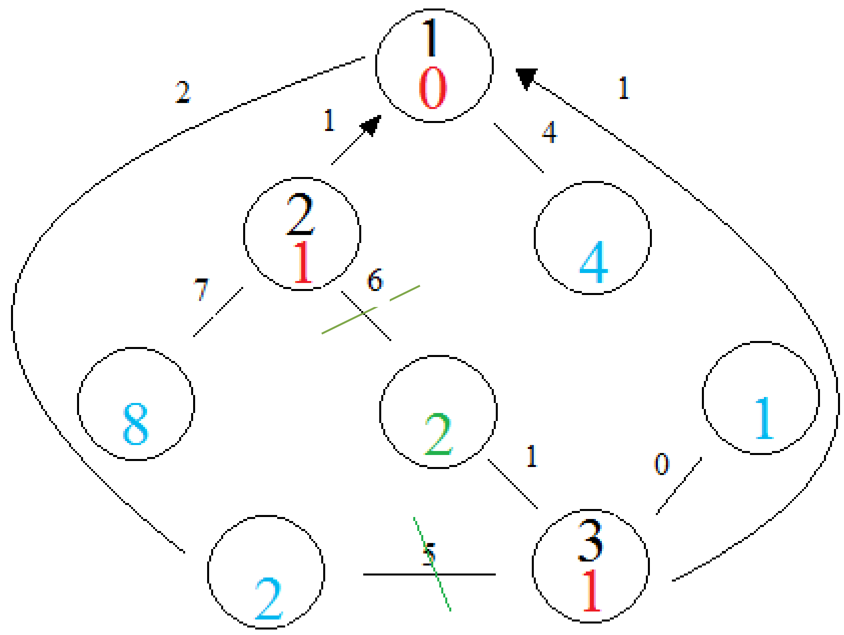


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

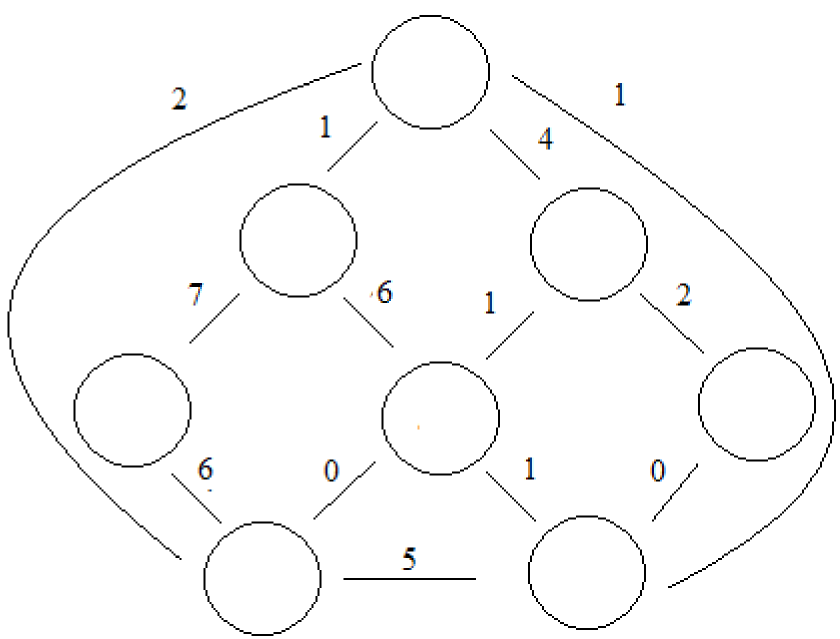


What we know

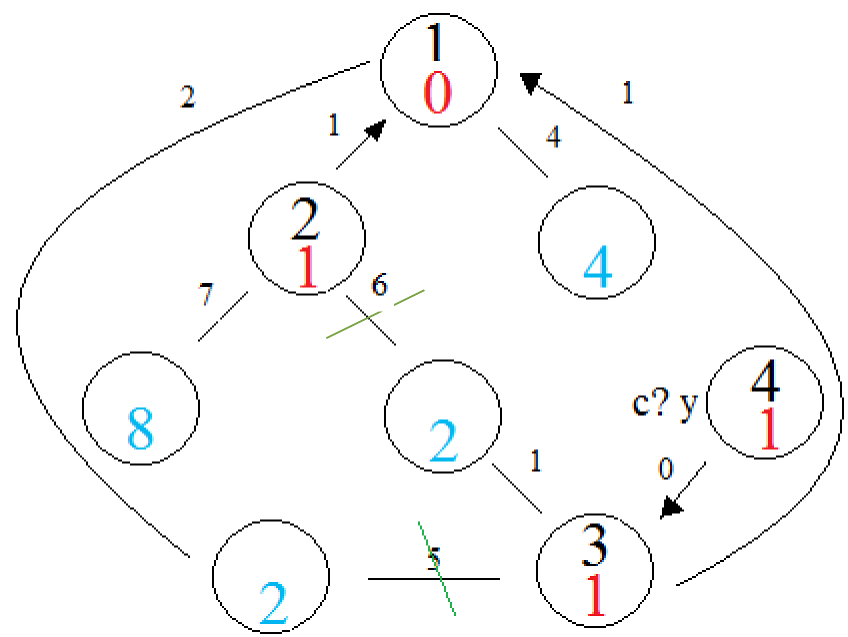


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

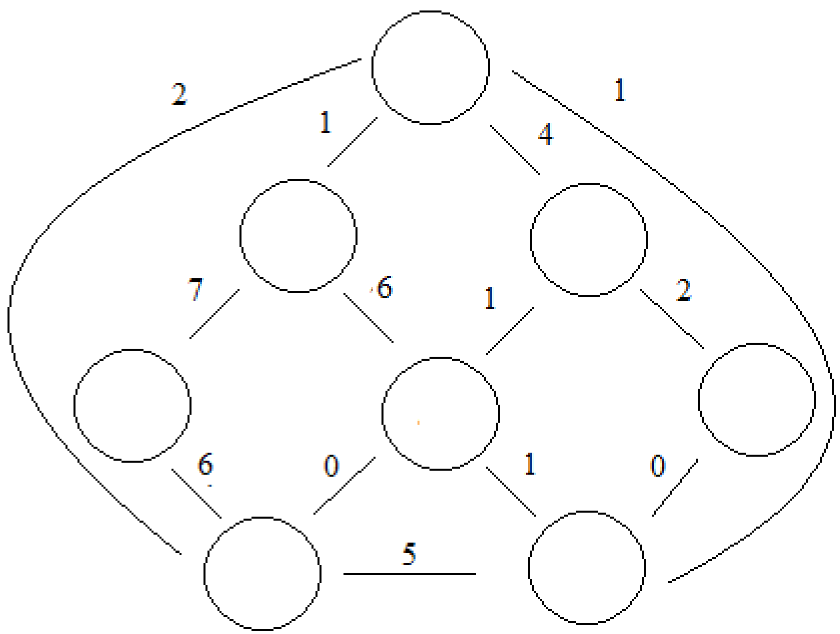


What we know

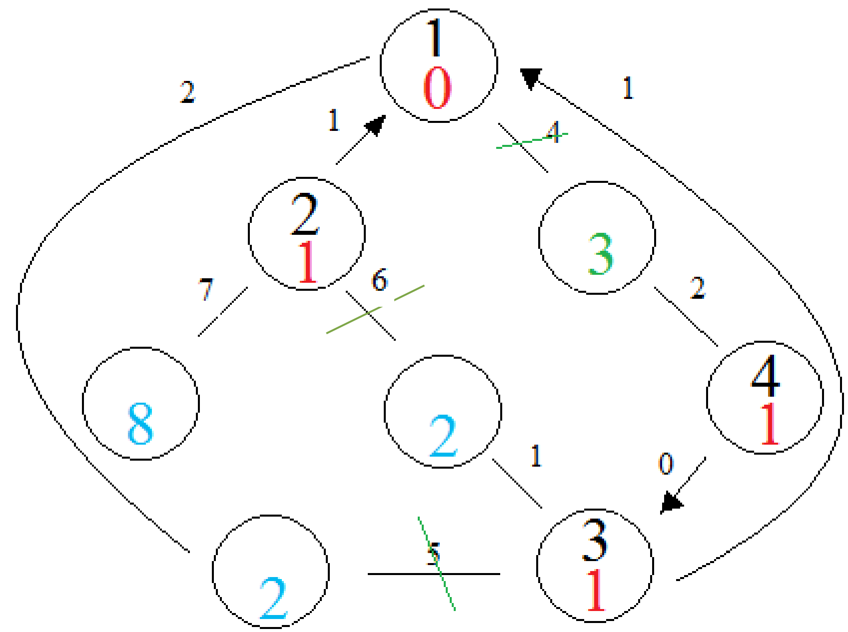


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

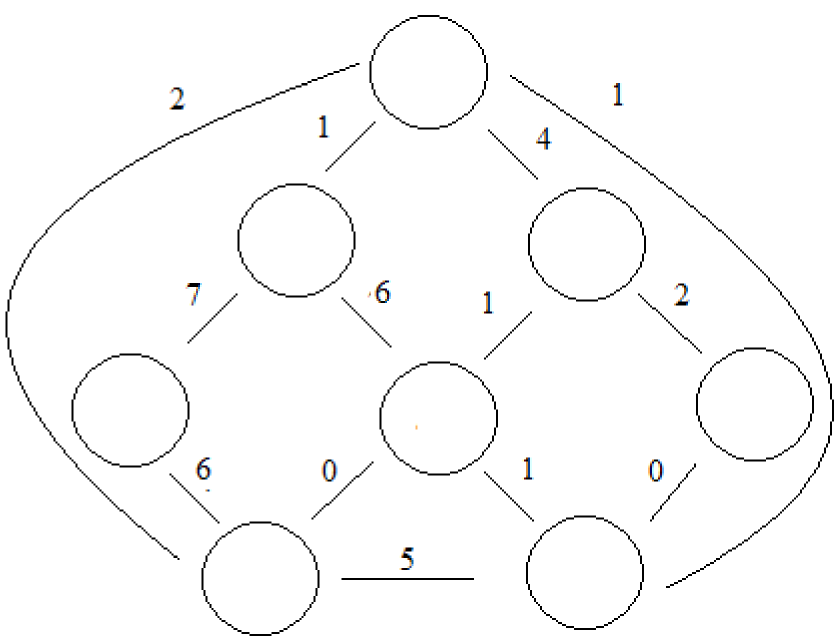


What we know

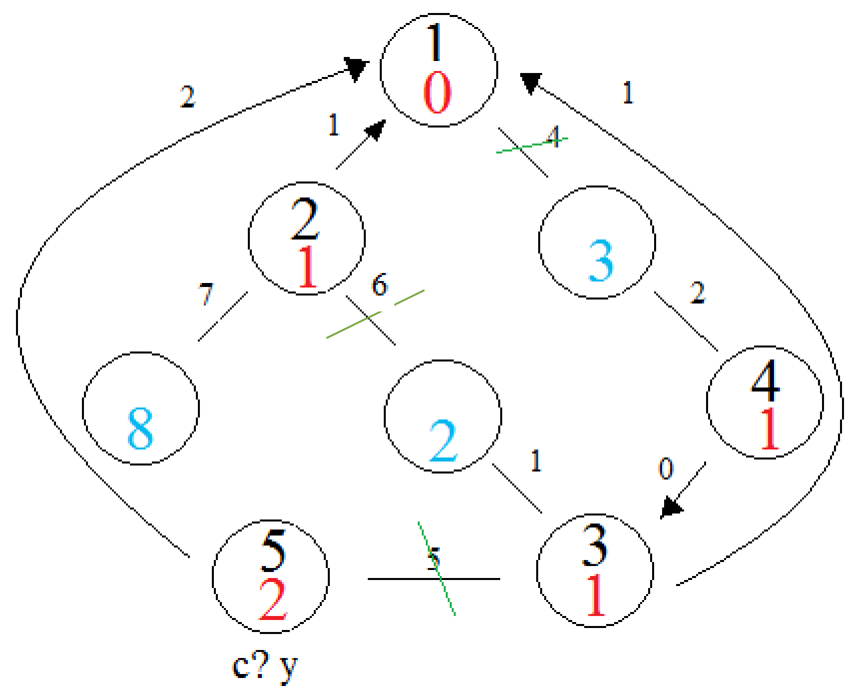


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

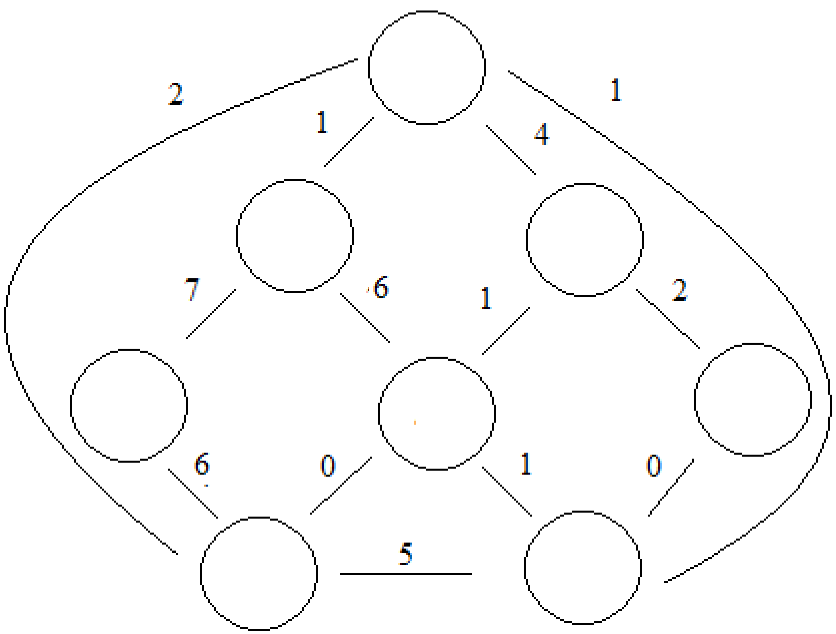


What we know

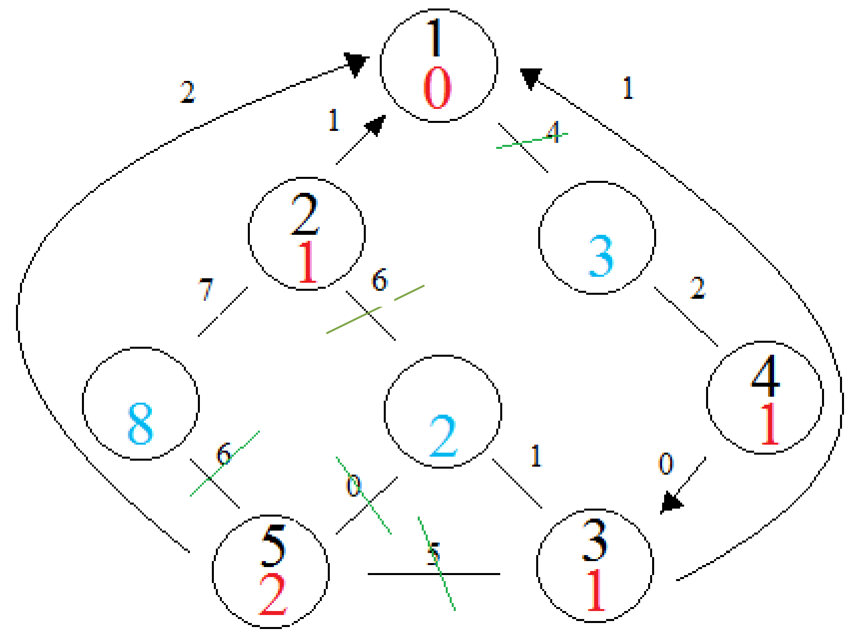


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

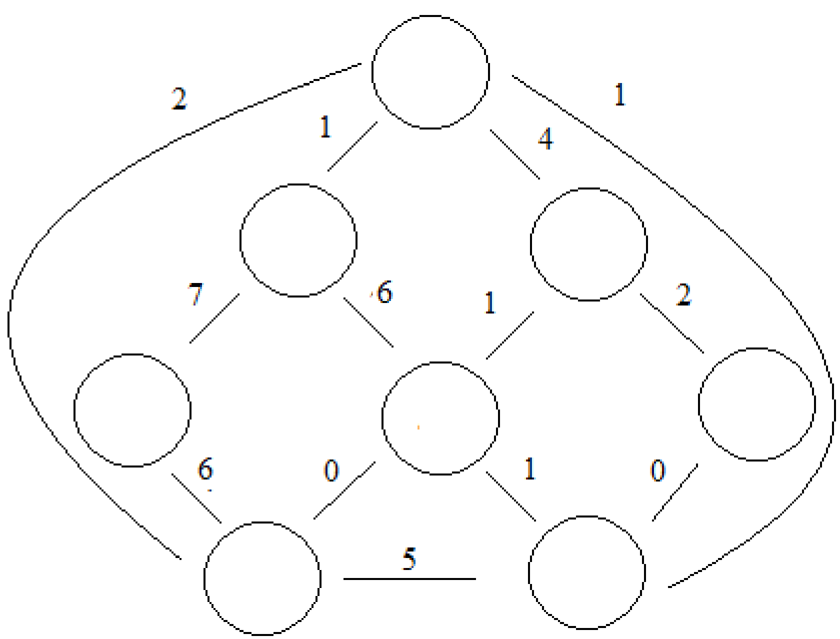


What we know

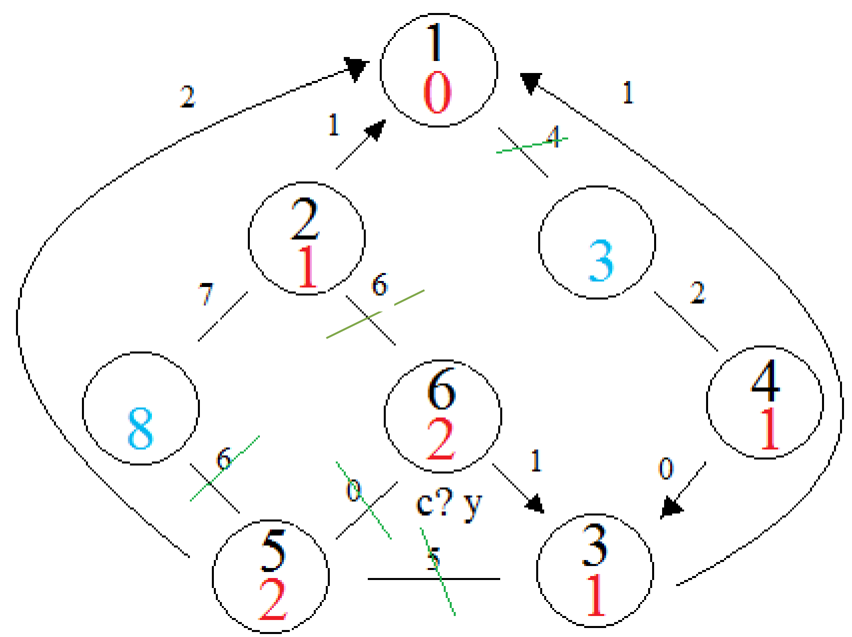


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

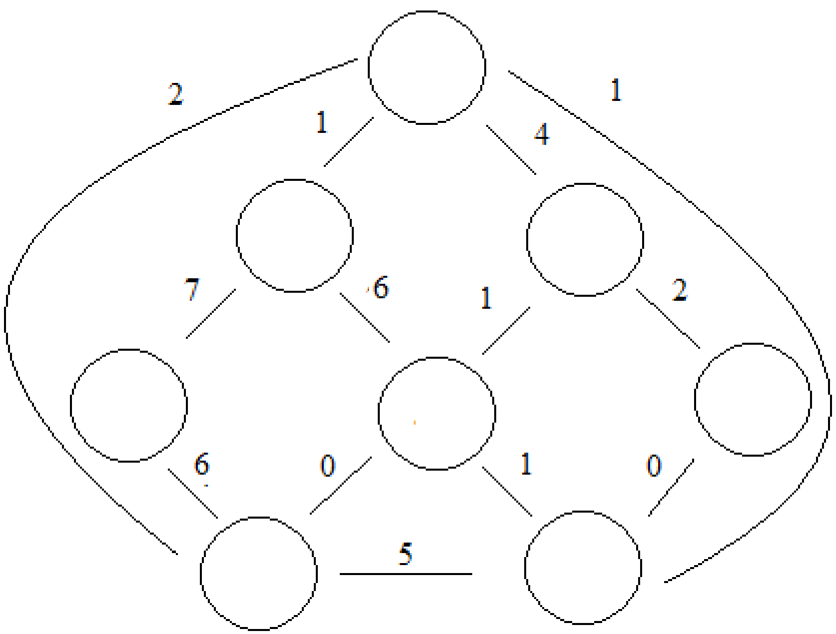


What we know

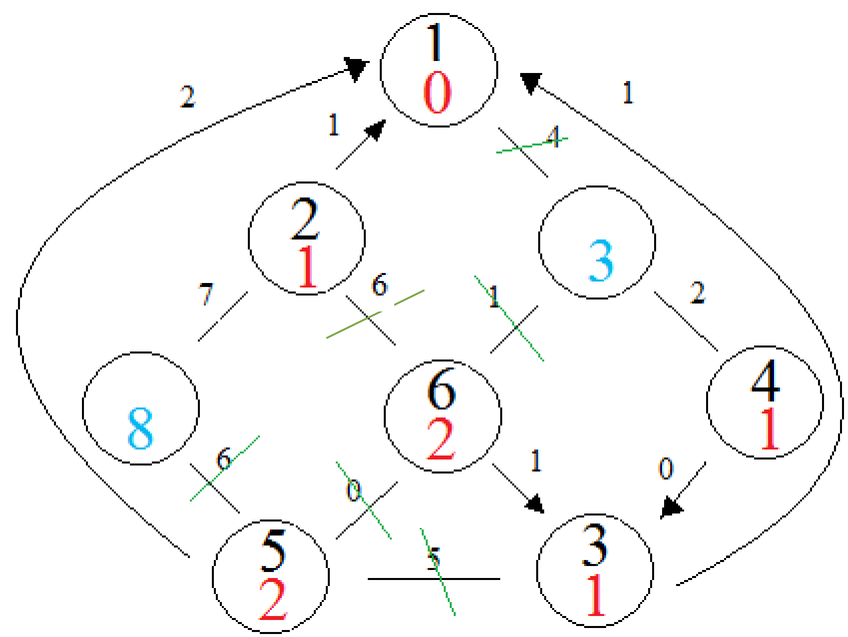


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

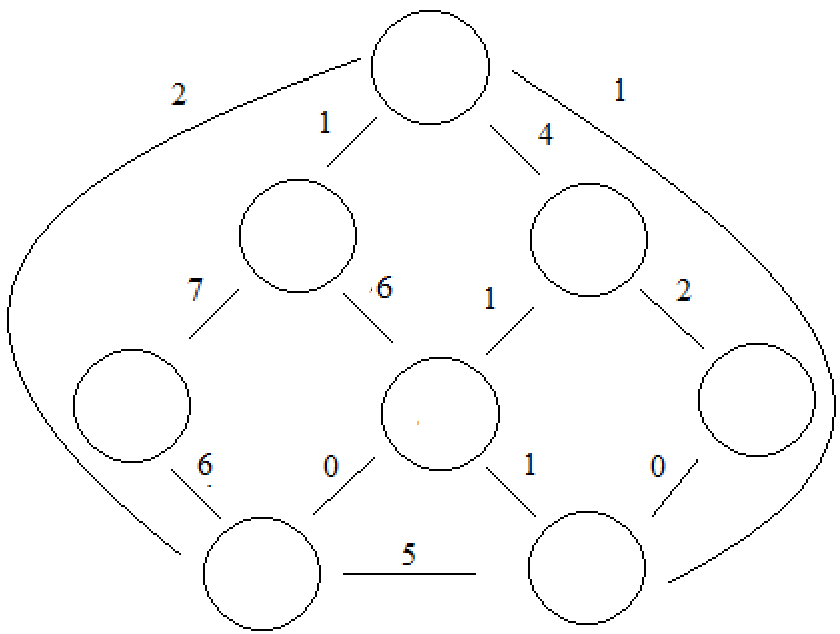


What we know

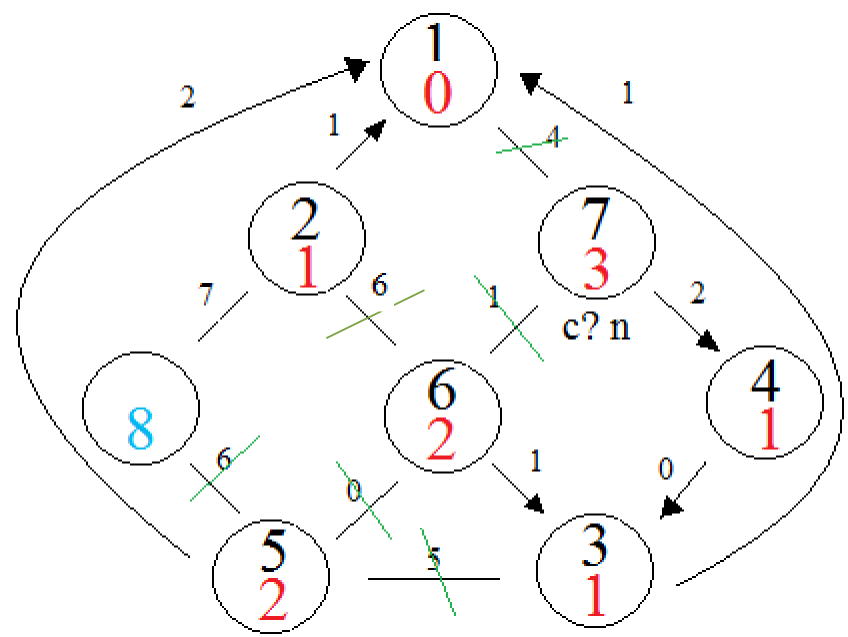


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

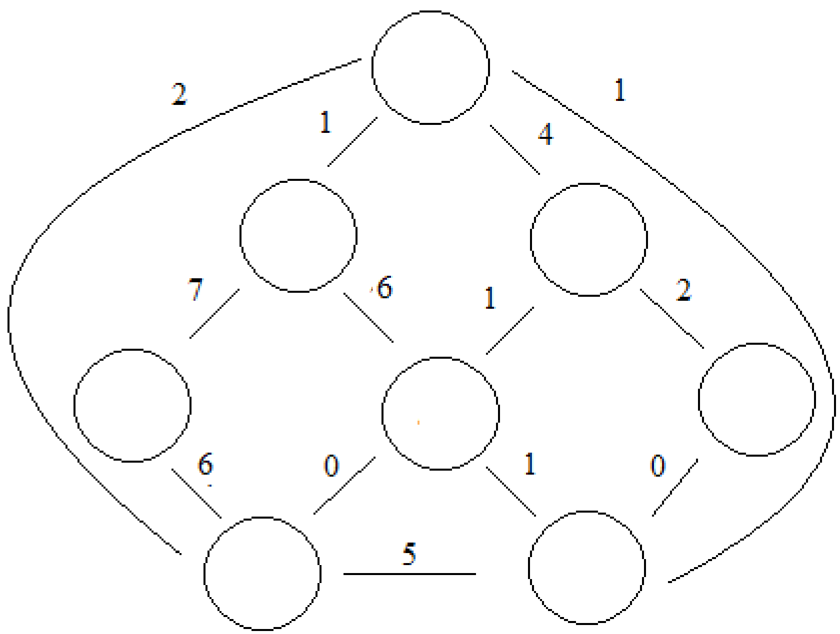


What we know

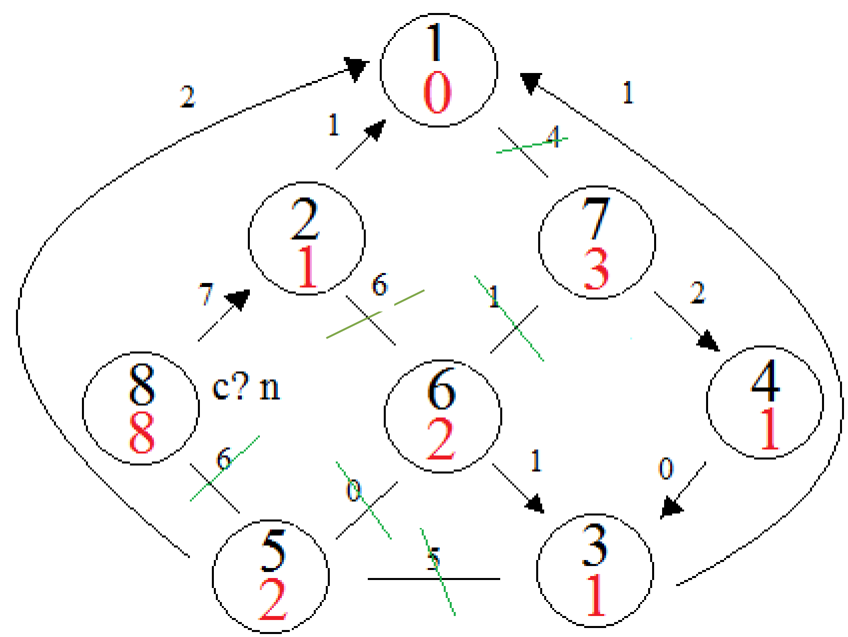


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph

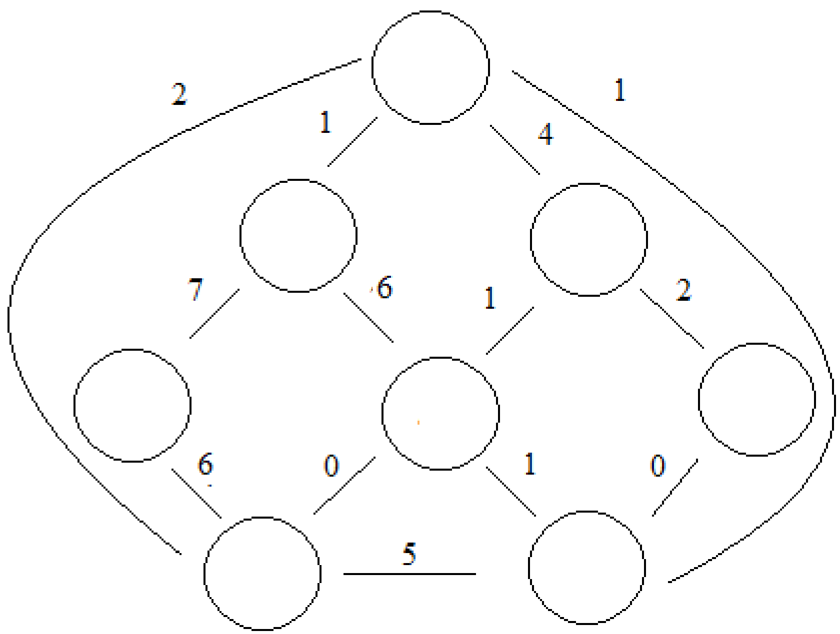


What we know

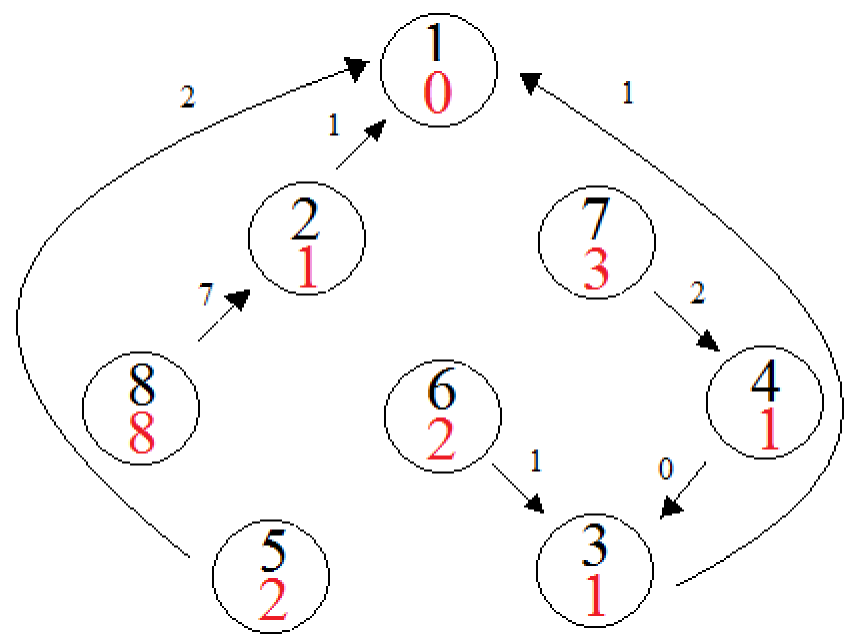


Uninformed Search Revisited: Dijkstra's (least-cost-first) Search

The whole graph



What we know



Informed Search
Greedy Best-First Search

- Greedy best-first search: expands the node on the fringe that is closest to the goal.
- In 2D or 3D Cartesian space, if cost = distance
  - Expand the node that is closest to the goal first.
  - $f(n) = h(n)$
  - Trip navigation example on the board.

Informed Search
A* Search

- You are going to implement this as part 3 of your project.
- Idea: minimize the total estimated solution cost.
- Expand Nodes that have a low value of:

$$f(n) = g(n) + h(n)$$

- $g(n)$: actual cost from the start to a node n
- $h(n)$: estimated cost from node n to the goal
  - If $h(n) = 0$ this reduces to Dijkstra's

Informed Search
A* Search

- (This is part 3 of your project).
- Idea: minimize the total estimated solution cost.
- Expand the nodes that has the lowest value of:

$$f(n) = g(n) + h(n)$$

- $g(n)$: actual cost from the start to a node n
- $h(n)$: estimated cost from node n to the goal
  - If $h(n)$ is an *admissible* and *consistent* then A* is optimal.
  - That is, if $h(n)$ never overestimates the cost to the goal, and maintains the triangle inequality, then A* will always find the best solution.
  - How fast this happens depends on how good your heuristic is.

Informed Search
A* Search

- How to apply this to navigation in:
  - real environments (robotics)
  - simulated environments (computer games)
- Step 1: model the world as a 4 or 8-connected graph (i.e. a chessboard).
  - Graph locations are defied by row and column.
  - Neighbours are row ±1 and column ±1
  - Let each node contain cost information about the world—now the graph is called an occupancy grid.
- Step 2: think of a good heuristic
  - Any ideas...
- Step 3: implement A* with your heuristic.

Informed Search
A* Search

- Step 4: make some code that does this:
  - Sense or input information about your environment
  - figure out your start and goal locations
  - run A* from start to goal on your occupancy grid
  - Move along the resulting path a little bit
  - Repeat
- This is already implemented on the LAGR robot
  - All you have to do is implement A* with your own heuristic.
- Example on board...
  - Cat chasing a mouse in an environment that has cost given by: dirt = 1, grass 3, and water = 10

Informed Search
A* Search

- A* pseudo-code (found on Wikipedia):
  - **function** A*(start,goal)
  - **var** closed := *the empty set*
  - **var** q := make_queue(path(start))
  - **while** q *is not empty*
  - **var** p := remove_first(q)
  - **var** x := *the last node of p*
  - **if** x **in** closed
  - **continue**
  - **if** x = goal
  - **return** p
  - *add x to closed*
  - **foreach** y **in** successors(x)
  - enqueue(q, p, y)
  - **return** failure

Informed Search
A* Search

- Grading for part 3
  - 50 % Does it work off-line on maps that I give it?
  - 25 % Explanation of why you chose a particular heuristic.
  - 25 % Does it run in real-time on the LAGR robot and make good decisions?
    - You'll need to have it running at about 1 frame per second on a map of 100 X 100 grids