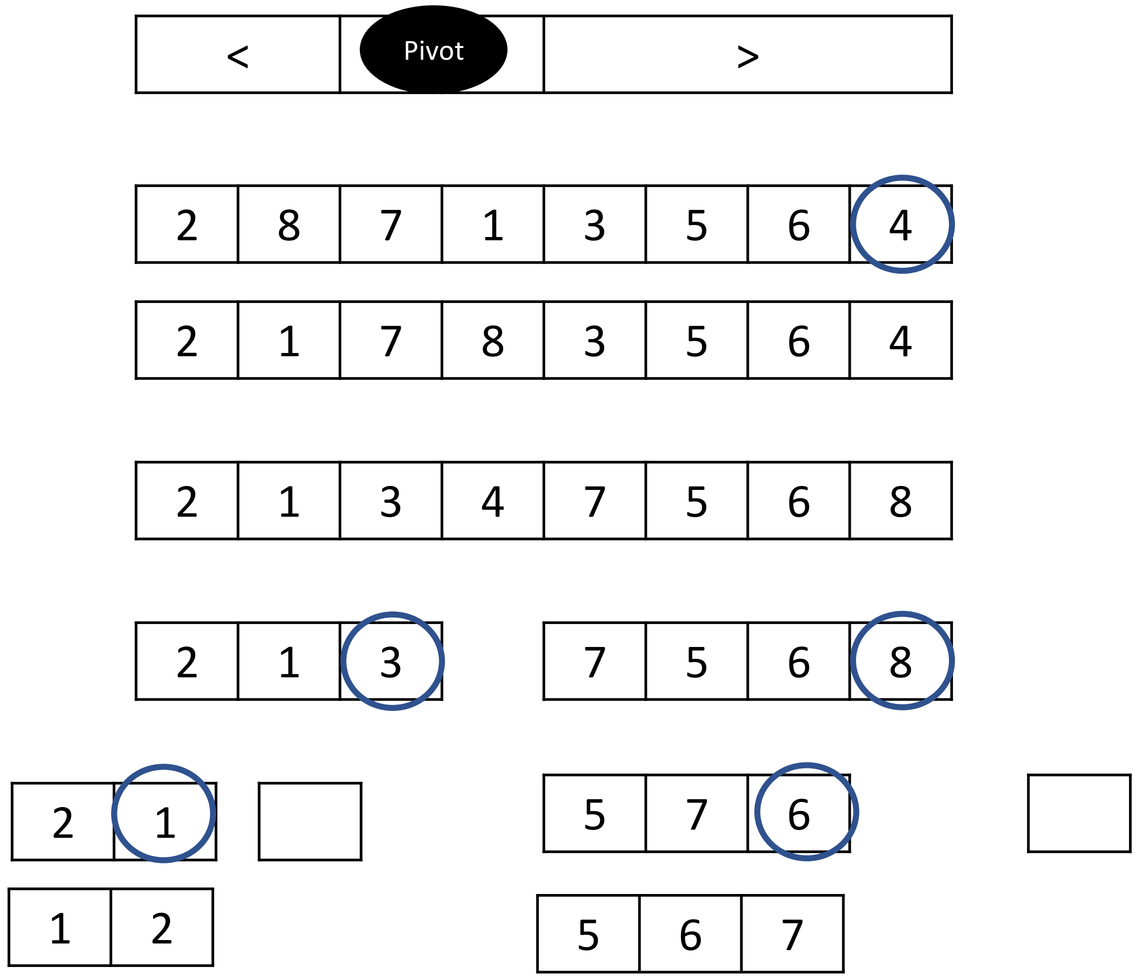
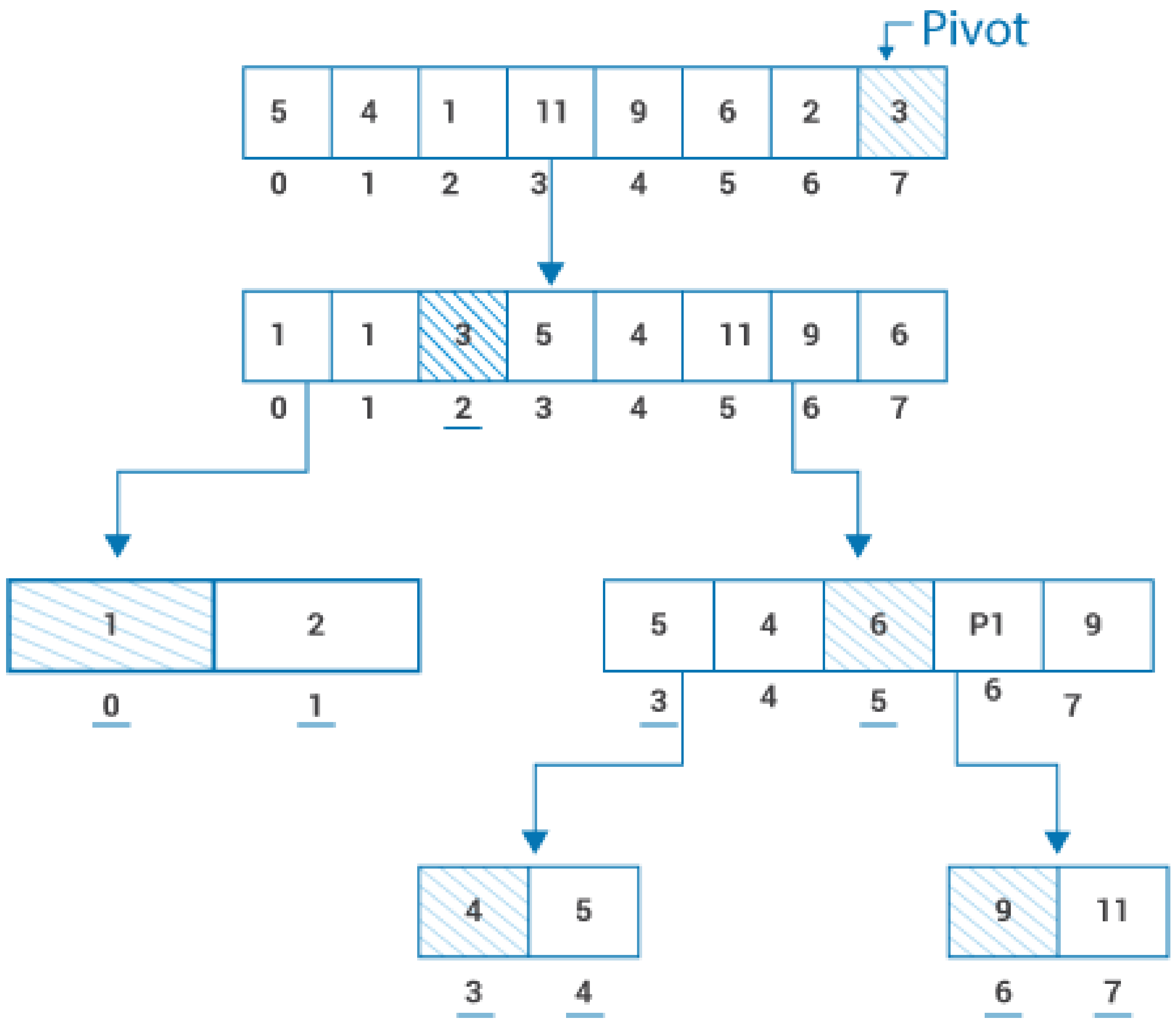
Randomized Algorithm

- Quick Sort visualize | Algorithms | HackerEarth

Quick Sort



Quick Sort



Pivot
5 4 1 11 9 6 2 3
0 1 2 3 4 5 6 7
1 1 3 5 4 11 9 6
0 1 2 3 4 5 6 7
1 2
0 1
5 4 6 P1 9
3 4 5 6 7
4 5
3 4
9 11
6 7

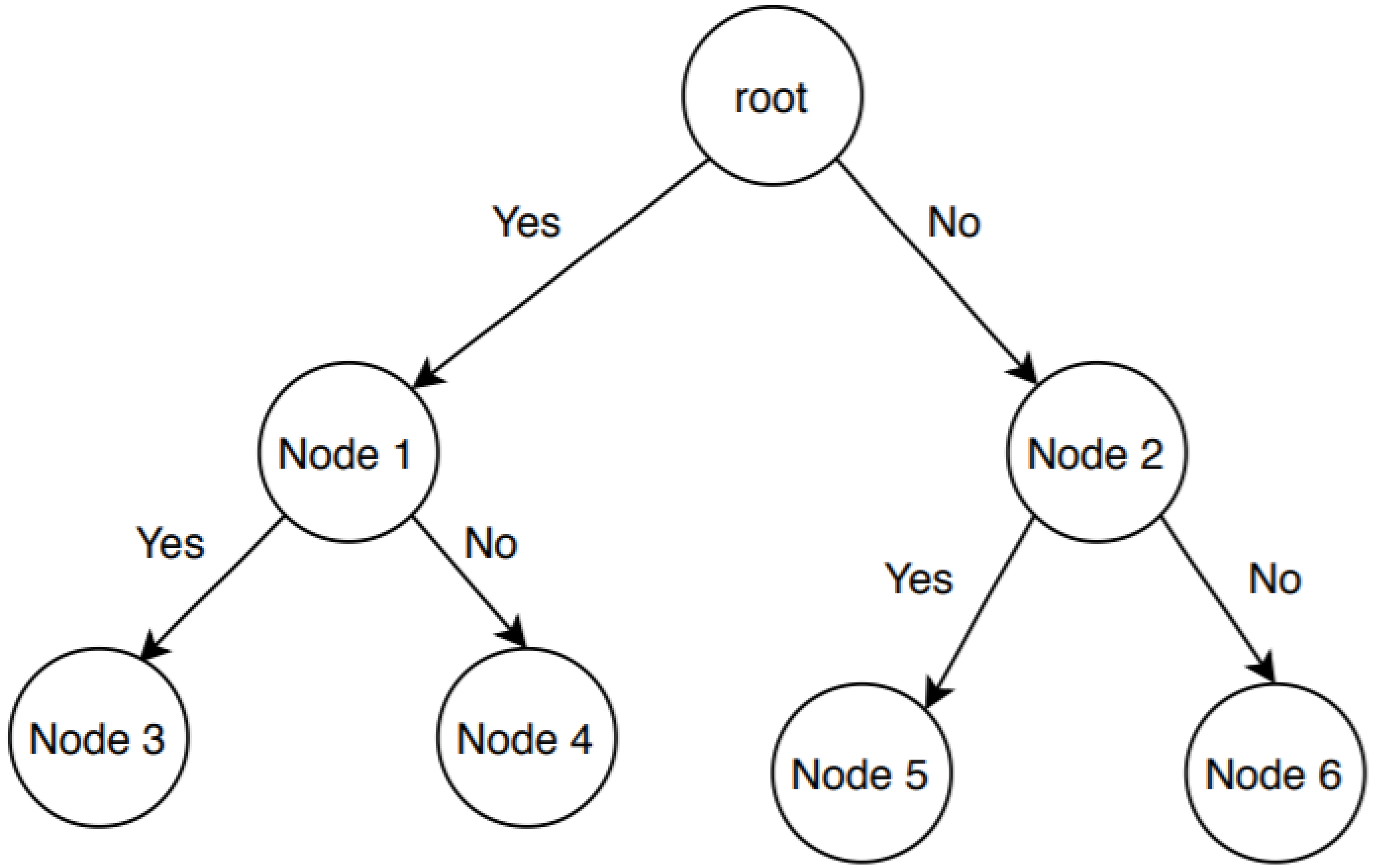
Quick Sort

- QuickSort(A):
  - **If** len(A) <= 1:
    - **return**
  - Pick some x = A[i] at random. Call this the **pivot**.
  - PARTITION the rest of A into:
    - L (less than x) and
    - R (greater than x)
  - Replace A with [L, x, R] (that is, rearrange A in this order)
  - QuickSort(L)
  - QuickSort(R)

Branch and bound (B&B)

Introduction

- a branch and bound algorithm explores the entire search space of possible solutions and provides an optimal solution.



Branch and bound (B&B)

- Before constructing the rooted decision tree, we set an upper and lower bound for a given problem based on the optimal solution. At each level, we need to make a decision about which node to include in the solution set. At each level, we explore the node with the best bound. In this way, we can find the best and optimal solution fast.

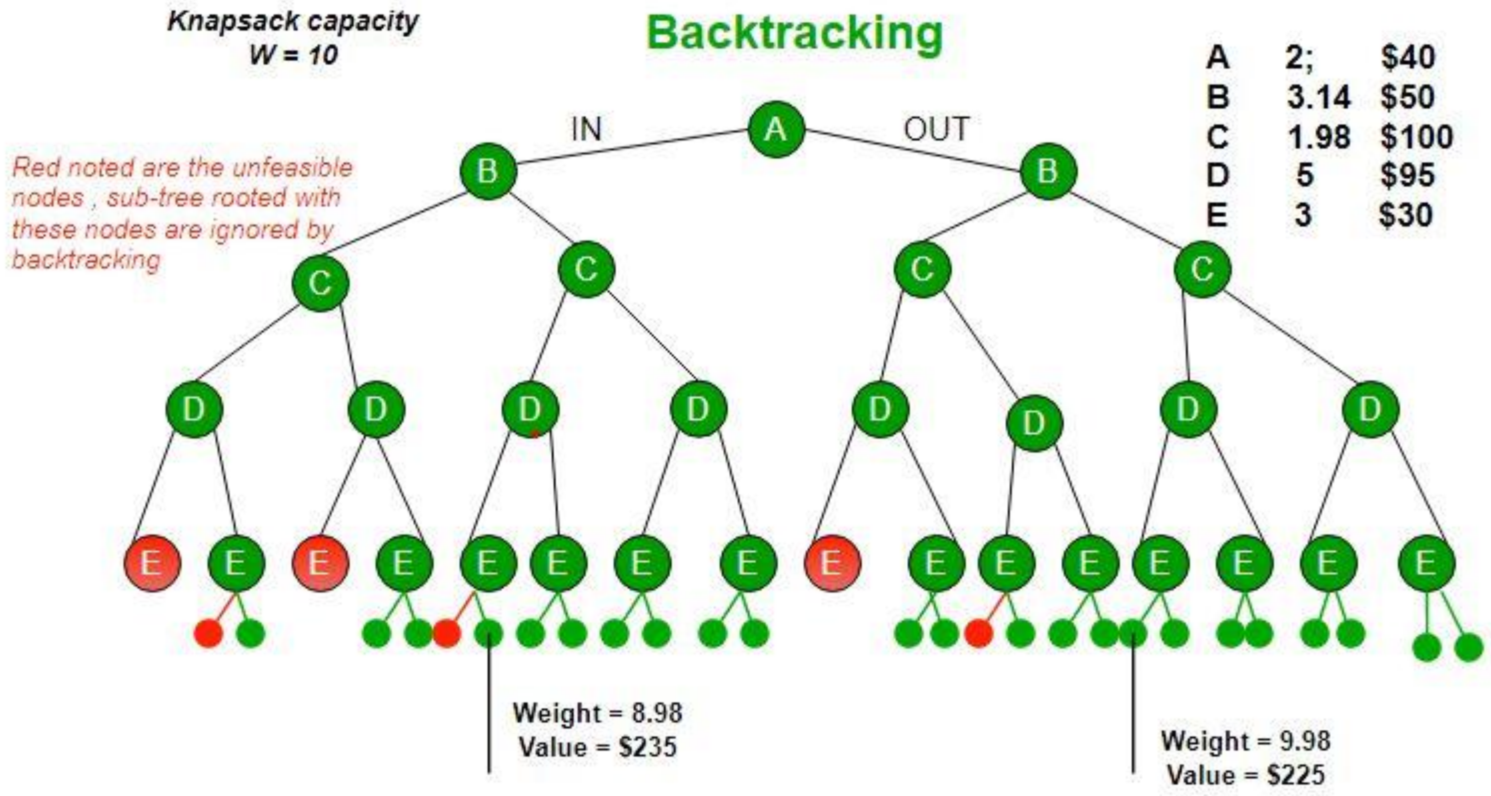
0/1 Knapsack using Branch and Bound

- A Greedy approach is to pick the items in decreasing order of value per unit weight. The Greedy approach works only for fractional knapsack problem and may not produce correct result for 0/1 knapsack.
- We can use Dynamic Programming (DP) for 0/1 Knapsack problem. In DP, we use a 2D table of size n x W. The DP Solution doesn’t work if item weights are not integers.
- Brute Force. With n items, there are 2n solutions to be generated, check each to see if they satisfy the constraint, save maximum solution that satisfies constraint. This solution can be expressed as tree.

0/1 Knapsack using Branch and Bound - GeeksforGeeks

0/1 Knapsack using Branch and Bound

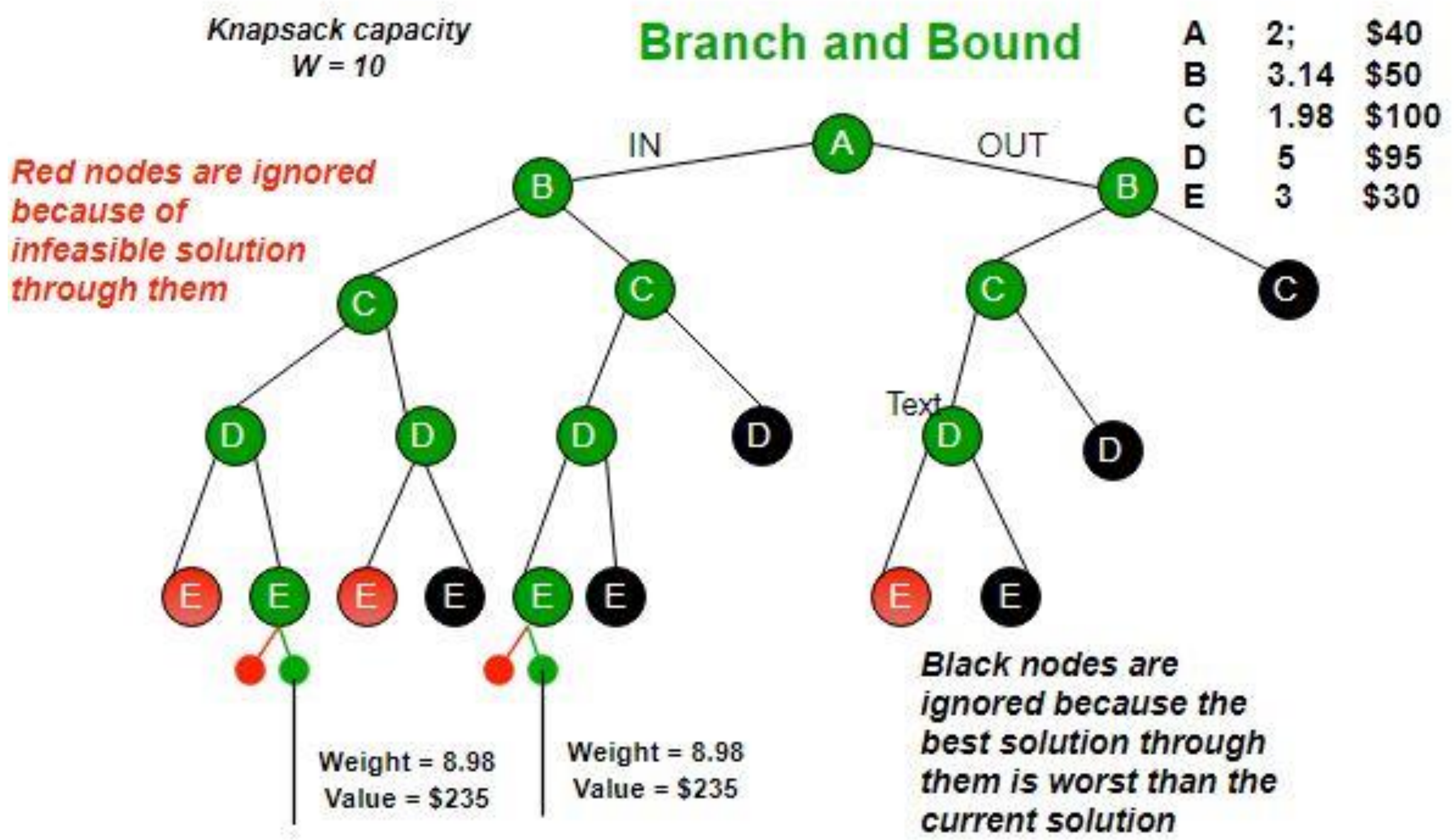
- We can use **Backtracking** to optimize the Brute Force solution. In the tree representation, we can do DFS of tree. If we reach a point where a solution no longer is feasible, there is no need to continue exploring.



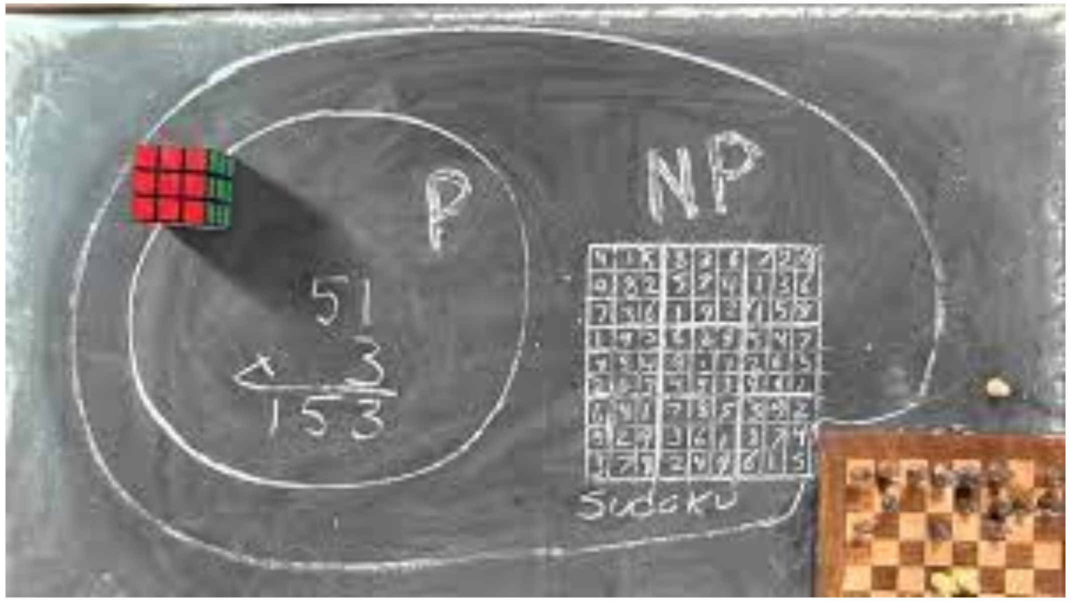
Knapsack capacity W = 10
Backtracking
A 2; $40
B 3.14 $50
C 1.98 $100
D 5 $95
E 3 $30
IN
OUT
A
B
C
D
E
Red noted are the unfeasible nodes , sub-tree rooted with these nodes are ignored by backtracking
Weight = 8.98 Value = $235
Weight = 9.98 Value = $225

[0/1 Knapsack using Branch and Bound - GeeksforGeeks]

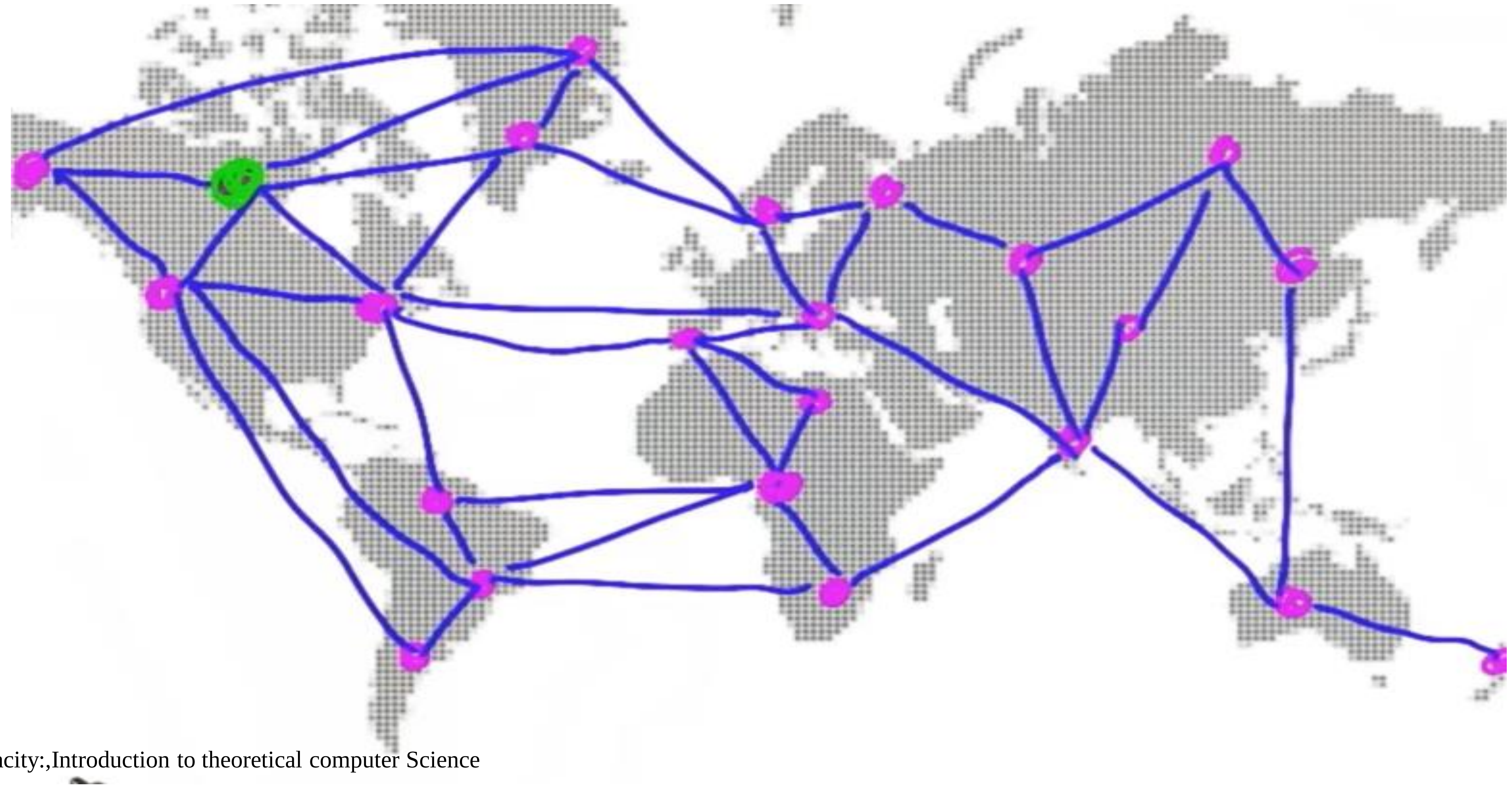
0/1 Knapsack using Branch and Bound


Knapsack capacity
W = 10
Branch and Bound
A 2; $40
B 3.14 $50
C 1.98 $100
D 5 $95
E 3 $30
IN
OUT
A
B
C
D
E
Text
Red nodes are ignored because of infeasible solution through them
Black nodes are ignored because the best solution through them is worst than the current solution
Weight = 8.98
Value = $235
Weight = 8.98
Value = $235

P vs NP

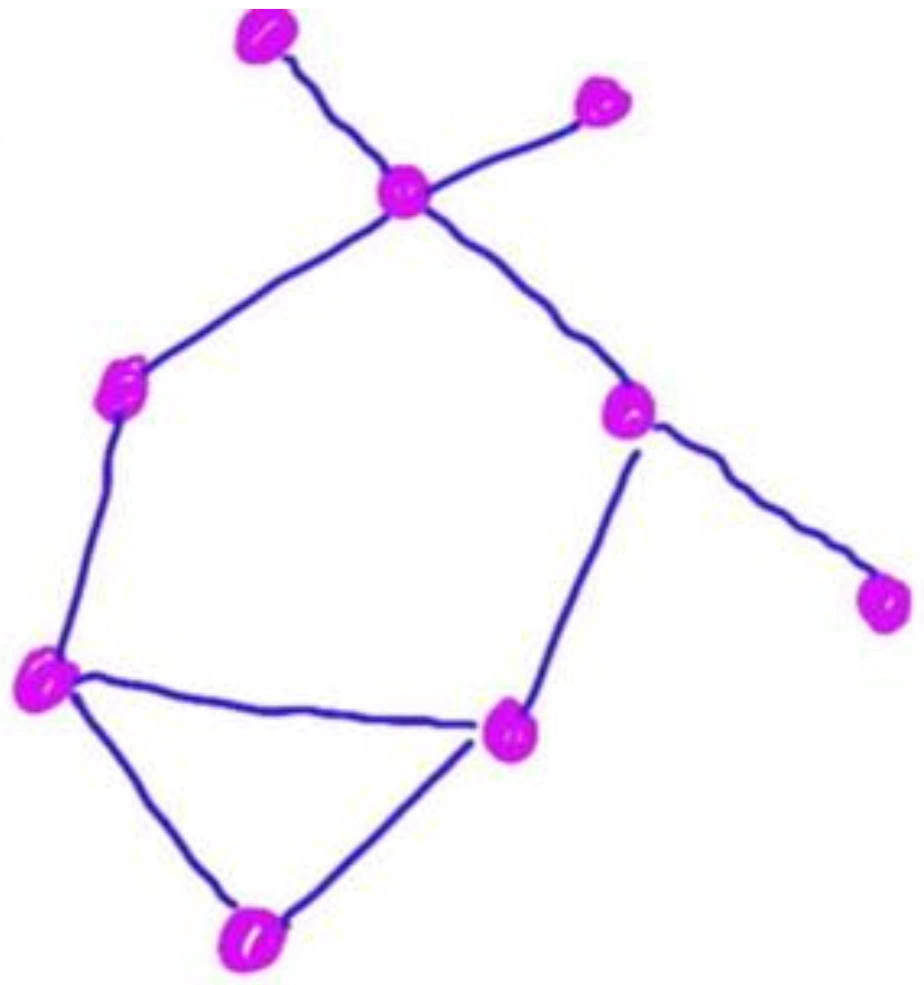


Installing Monitoring Devices(1)

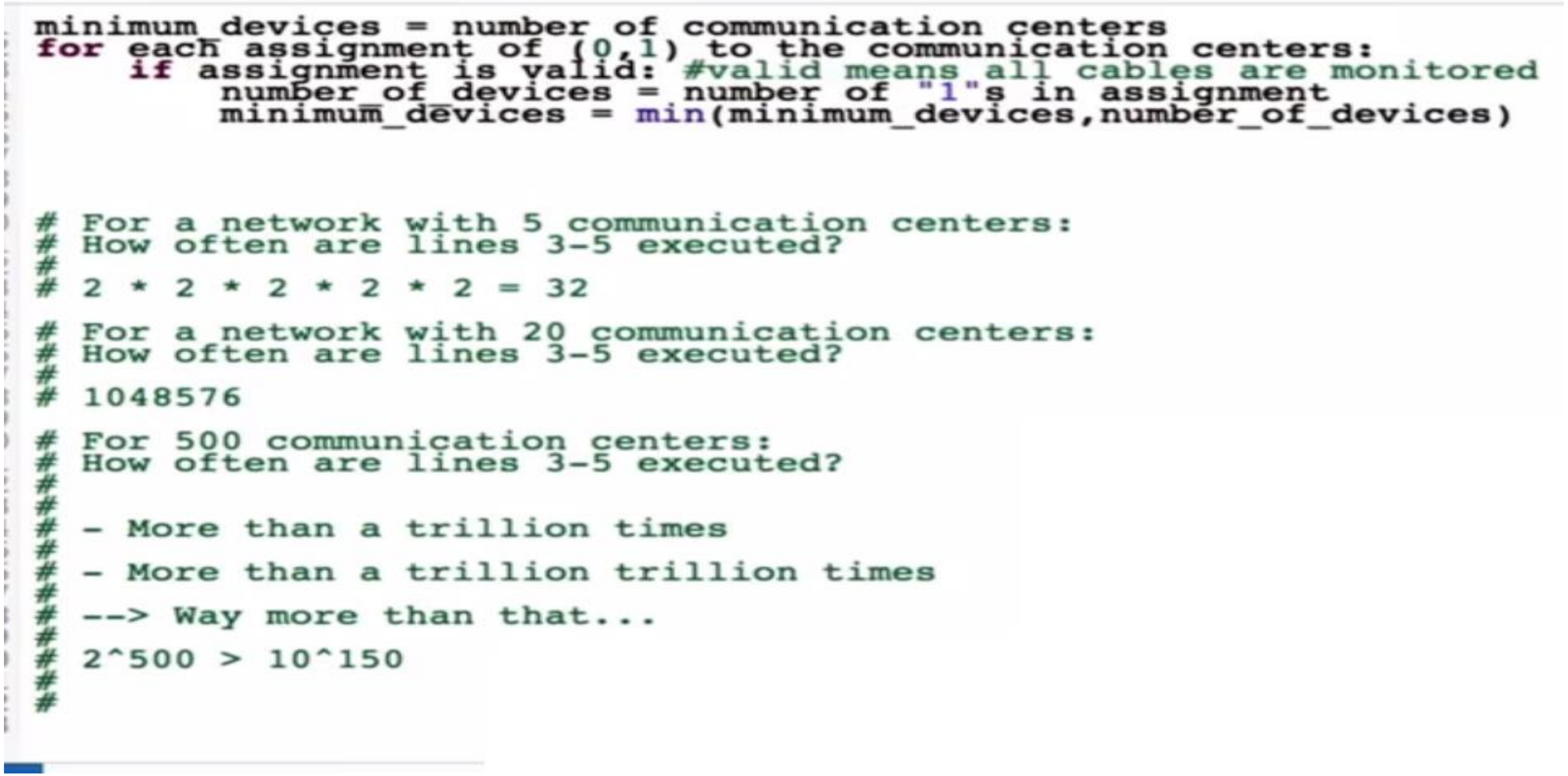


Udacity:,Introduction to theoretical computer Science

Installing Monitoring Devices(2)



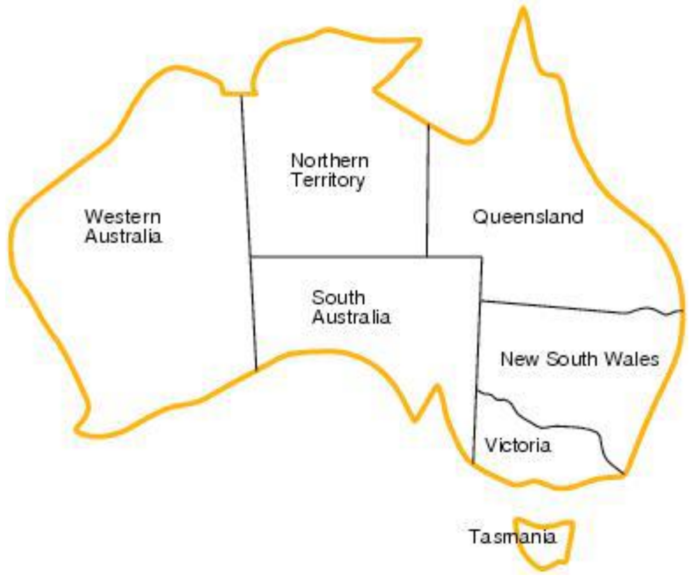
Installing Monitoring Devices(3)

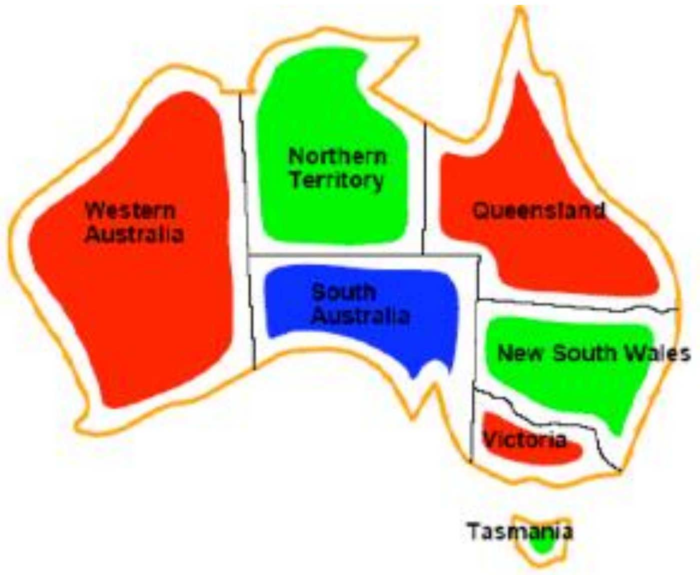


```
minimum_devices = number of communication centers
for each assignment of (0,1) to the communication centers:
    if assignment is valid: #valid means all cables are monitored
        number_of_devices = number of "1"s in assignment
        minimum_devices = min(minimum_devices,number_of_devices)



# For a network with 5 communication centers:
# How often are lines 3-5 executed?
#
# 2 * 2 * 2 * 2 * 2 = 32

# For a network with 20 communication centers:
# How often are lines 3-5 executed?
#
# 1048576

# For 500 communication centers:
# How often are lines 3-5 executed?
#
#
# - More than a trillion times
#
# - More than a trillion trillion times
#
# --> Way more than that...
#
# 2^500 > 10^150
#
#
```

Map Coloring

- Given a map of Australia, color it using the minimum number of colors such that no neighboring territories have the same color.





Diophantine Equations

- DIOPHANTINE EQUATIONS: Given a polynomial equation, does it have an integer solution? e.g.:
  - $\boldsymbol{x^2}+\mathbf{2}xy - \boldsymbol{y^3}-\mathbf{13} = 0$
  - $\mathbf{9}\boldsymbol{x^2}+\mathbf{12}\,\boldsymbol{x^3} = \mathbf{1000}$
- These types of equations are named after the ancient Greek mathematician Diophantus. A linear Diophantine equation is a first-degree equation of this type
- Hilbert's tenth problem is the tenth on the list of mathematical problems that the German mathematician David Hilbert posed in 1900. It is the challenge to provide a general algorithm which, for any given Diophantine equation (a polynomial equation with integer coefficients and a finite number of unknowns) can decide whether the equation has a solution with all unknowns taking integer values.

Example

- Algorithm → O(2^n) on a machine execute $10^8\ op/sec$
- N=50
  - $t = \frac{2^{50}}{10^8}$ =0.317 years
- N=100
  - $t = \frac{2^{100}}{10^8}$ =10^{15} $*$0.317 years

Subset Sum Problem

- Given a set of integers S and a target integer T. Determine if S has a subset that sum to T Exactly.
- Example :
  - S={2,3,4,8}
  - T=7→ yes {3,4}
  - T=9→yes {2,3,4}
  - T=11→yes {3,8}
  - T=16→no
- In general, we need every combination (subset). For a set of size n,2^nsubsets

Tractable vs Intractable

Tractable	log n
	n
	nlgn
	n^2
	n^3
	:
	n^k
Intractable	2^n
	3^n
	:
	:
	:
	k^n
	$n!$
	n^{2n}

Revisit 0-1 Knapsack(DP)

- Polynomial time means that the running time is bounded by a polynomial in the length of the input.
- The running time of 0-1 knapsack is bounded by nW.
  - n, the number of items <= length of the input.
  - But W, the Capacity, is a number that appears in the input, in binary. In ℓ bits, you can write a number up to 2^lso W is potentially exponential in the length of the input, not polynomial.
- In practice, it works well for most instances of the problems

Traveling salesman problem(TSP)

- Given a complete weighted undirected graph. Find a simple cycle that goes through all (TSP tour) that has the lowest cost➔ Optimization Problem

- Given a complete weight graph and a target cost T. Determine if this graph has a TSP tour and has accost <=T.→ Decision Problem