Heuristics

Simeon Monov

What is heuristic algorithms

- A heuristic algorithm solves a problem faster than full search algorithms by sacrificing optimality, accuracy or completeness for speed.
- Heuristic algorithms mostly are used to solve NP-Complete (NP-hard) problems.
- Heuristic algorithms are used, when optimal solution is not needed and “good enough” solution is enough.

Heuristic algorithms - examples

- **Greedy Algorithms** – problem-solving heuristic of making the locally optimal choice at each stage
- **Genetic Algorithms** – search algorithms based on techniques inspired by natural selection such as inheritance, mutation, and crossover
- **Tabu Search** – local search algorithm
- **Artificial Neural Networks (ANNs)** – pattern recognition algorithms influenced how animal brain works
- **Simulated Annealing** – global search algorithm, borrowing ideas from metallurgical processes

Greedy Algorithms

- A greedy algorithm is any algorithm that follows the problem-solving heuristic of making the locally optimal choice at each stage.
- In many problems, a greedy strategy does not produce an optimal solution, but a greedy heuristic can yield locally optimal solutions that approximate a globally optimal solution in a reasonable amount of time.

Greedy solution for TSP

- The traveling salesman problem is as follows:
  - Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?
- TSP is an NP-hard problem
- Greedy solution / strategy to solve TSP: “At each step of the journey, visit the nearest unvisited city.”
- The above greedy solution does not intend to find the best solution, but to return “good enough” solution in good time.

Find minimum number of currency notes to sum an amount

- We have unlimited supply of coins of different denominations.
- Find the minimum number of coins of different denominations, that sum to a specified amount.
  - Example 1:
    - Coins set: (1, 2, 5, 10, 20, 50, 100)
    - 345 = 100 + 100 + 100 + 20 + 20 + 5
  - Example 2:
    - Coins set: (1, 2, 5, 7)
    - 10 = 5 + 5

Find minimum number of currency notes to sum an amount - greedy

- Let's explore a greedy solution of the problem:
  1) Start with **sum = initialSum**
  2) Find a maximum value coin (**coinVal**), which is smaller or equal to **sum**
  3) Add **coinVal** to solution
  4) sum = sum – coinVal
  5) Repeat step 2 – 4 until **sum > 0**
- The above solution will not always give optimal answer. It will always give optimal answer only when the value every next coin is greater or equal to the double of the previous coin. For example (1,2,5,10,50,100) will always give optimal solution: for example 345 = 100 + 100 + 100 + 20 + 20 + 5
- In cases when some of the coins' value is smaller of the double value of the previous coin, the algorithm will not always give correct answer. For example (1, 2, 5, 7) and amount 10, the algorithm will give result 10=7+2+1, but the best solution is 10=5+5

Knight tour greedy solution

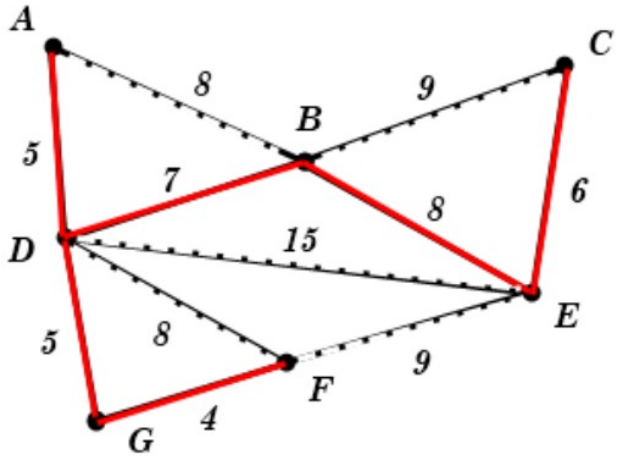
- Warnsdorff's algorithm (described by H. C. von Warnsdorff in 1823) for Knight's tour problem: "Always move to an adjacent, unvisited position on the board with minimal degree (minimum number of unvisited adjacent)".
- The algorithm:
  1) Start from startX, startY
  2) Mark current position as visited
  3) If we already visited all positions, exit with solution
  4) Visit next possible position, which has minimal degree (check all possible positions for each new position and choose the one with minimal count)
  5) If visiting next, minimal degree solution returned true, return true otherwise return false

Minimal spanning tree

- We have a set of cities which are not connected. It is known if two cities can be connected and what is the distance between them. Find paths between the cities in such a way, that all cities are connected and the sum of the path lengths are minimal.

Prim's algorithm

- Prim's algorithm for finding minimal spanning tree:
  - Create undirected graph with edges equals to the path lengths.
  - Find the shortest edge in the graph.
  - Add it to the result tree.
  - On every step find the shortest edge, which do not belong to the result tree.
  - Repeat above step until all nodes of the graph are added to the tree



Always optimal greedy algorithms

- Dijkstra – find all minimal paths from one vertex to all other vertices in a graph
- Prim’s algorithm