1

Yes, uniform cost search could be a special case of A-star search. A-star search is a modified breadth first search that picks the next node to explore based on its cost from the beginning and its expected cost to the goal based on a heuristic. A-star simply adds these two costs together to choose the most optimal node to explore. Uniform cost search is also a modified breadth first search which only picks the best node on its cost from the initial state. Where this differs from A-star is its lack for accounting for the estimated cost to the end. This means that if we used A-star with a heuristic that estimates its cost to goal as 0 every time, the sum of the initial cost and estimated cost will equal the initial cost only, just like uniform cost search.

2

Fagaras

Cost: 0

1. Fagaras ≠ Drobeta
2. Neighbors
    - I. Sibiu
        - A. f = 99
        - B. g = 173
        - C. h = 272
    - II. Bucharest
        - A. f = 211
        - B. g = 274
        - C. h = 485
3. Visited List
    - I. Fagaras
    - II. Sibiu
    - III. Bucharest
4. Current Priority Queue
    - I. 272: Sibiu
    - II. 485: Bucharest

Sibiu

Cost: 99

1. Sibiu ≠ Drobeta
2. Neighbors
    - I. Oradea
        - A. f = 151 + 99 = 250

B. g = 273

C. h = 523

II. Arad

A. f = 140 + 99 = 239

B. g = 202

C. h = 441

III. Rimnicu Village

A. f = 80 + 99 = 179

B. g = 145

C. h = 324

IV. Fagaras

A. In visited list

3. Visited List

I. Fagaras

II. Sibiu

III. Bucharest

IV. Oradea

V. Arad

VI. Rimnieu Village

VII. Fagaras

4. Current Priority Queue

I. 321: Rimnicu Village

II. 441: Arad

III. 485: Bucharest

IV. 523: Oradea

Rimnicu Village

Cost: 179

1. Rimnieu Village ≠ Drobeta
2. Neighbors

I. Sibiu

A. In visited list

II. Craiova

A. f = 146 + 179 = 325

B. g = 96

C. h = 421

III. Pitesti

A. f = 97 + 179 = 276

B. g = 177

C. h = 453

3. Visited List

I. Fagaras
II. Sibiu
III. Bucharest
IV. Oradea
V. Arad
VI. Rimnicu Village
VII. Fagaras
VIII. Craiova
IX. Pitesti

4. Current Priority Queue

I. 421: Craiova
II. 441: Arad
III. 453: Pitesti
IV. 485: Bucharest
V. 523: Oradea

Craiova

Cost: 325

1. Craiova ≠ Drobeta
2. Neighbors

I. Drobeta
  A. f = 120 + 325 = 445
  B. g = 0
  C. h = 445
II. Rimnicu Village
  A. In visited list
III. Pitesti
  A. In visited list

3. Visited List

I. Fagaras
II. Sibiu
III. Bucharest
IV. Oradea
V. Arad
VI. Rimnicu Village
VII. Fagaras
VIII. Craiova
IX. Pitesti
X. Drobeta

4. Current Priority Queue

I. 441: Arad

    II. 445: Drobeta
    III. 453: Pitesti
    IV. 485: Bucharest
    V. 523: Oradea

Arad

Cost: 239

1. Arad ≠ Drobeta
2. Neighbors
    I. Zerind
        A. f = 75 + 239 = 314
        B. g = 239
        C. h = 553
    II. Timisoara
        A. f = 118 + 239 = 357
        B. g = 164
        C. h = 521
    III. Sibiu
        A. In visited list
3. Visited List
    I. Fagaras
    II. Sibiu
    III. Bucharest
    IV. Oradea
    V. Arad
    VI. Rimnicu Village
    VII. Fagaras
    VIII. Craiova
    IX. Pitesti
    X. Drobeta
    XI. Zerind
    XII. Timisoara
4. Current Priority Queue
    I. 445: Drobeta
    II. 453: Pitesti
    III. 485: Bucharest
    IV. 521: Timisoara
    V. 523: Oradea
    VI. 553: Zerind

Drobet

Cost: 445

1. Drobet = Drobet
2. Path:
    I. Drobet
    II. Craiova
    III. Rimnicu Village
    IV. Sibiu
    V. Fagaras

3

a

Depth first search, it only knows about the current state and does not keep any competitors in memory

b

A-star search, the way nodes are picked are the same, and without pruning the choices, its just A-star

c

Gradient Descent, without any temperature, there are no random movements, so it just uses gradient descent

d

Random walk, it just randomly looks at a value and hopes its better

4

A-star uses huge amounts of memory compared to simulated annealing. A-star is not feasible for large data sets, whilst simulated annealing performs very well. Simulated annealing does not always provide the best solution, but it is pretty fast at producing a well optimized solution quickly, A-star takes much longer to find a solution.

5

Similar to simulated annealing, have a temperature value that slowly decreases over time or iterations. Implement a similar technique to random walk to improve the exploration of gradient ascent, where larger temperatures are more explorative and more likely to select unfavorable trades, while lower temperatures are safer and only take favorable trades. Once the temperature reaches 0, simply default back to gradient ascent to find that area's local maxima. The gradient could be considered a heuristic as it would give a rough estimate as to how close to the top of a local maxima is assuming the maxima are rounded at their peaks. I do not see much use for implementing the gradient as a heuristic for simulated annealing though.

6

a

In minimax, MAX always assumes that MIN will play optimally, so the algorithm does not account for MIN's play-style or deficiencies. So whether MAX plays an optimal MIN or a subobtimal MIN, it will still see every game tree the same regardless.

b

Say MAX and MIN are playing rock-paper-scissors. Presuppose we know that MIN likes to play scissors abnormally more than the other actions.

A regular MAX would not care about which action he chooses as they all have theoretically the same chance of winning, but using the information that MIN likes to play scissors, we would bias our choice to rock in order to beat MIN more.

7

Because we know that that variable must have a solution within the set. It it does not, then the whole set is unsolvable. If we select the variable with the smallest state space or greatest constraints, then we eliminate many possibilities without a chance of removing the solution combination. When selecting values, selecting a value that reduces the space the most will most likely lead to no solution first, as when we select values we are not guaranteed to still have the solution within our search set, thus we are more likely to hit the solution with fewer backtracks if we pick a non-limiting value.

8

a

I would likely choose a local search for CSPs, as they are fast and efficient at solving to find a solution to a problem. The constraints would be that words must fill the length of the cavity, and one box can only contain a single letter. A heuristic would be the number of letter collisions and the number of empty spaces. Local search should aim to minimize this heuristic.

b

I would choose to have the variables be words as they significantly reduce the state space.