Greedy Search Algorithms

- **Greedy search**
  
  A greedy search algorithm is an algorithm that uses a heuristic for making *locally optimal choices* at each stage with the hope of finding a global optimum.

  - **No backtracking!**
    
    - No reevaluating choices that the algorithm committed to earlier.
Hill Climbing Search

- Perhaps the most well known greedy search.
- Hill climbing tries to find the optimum (top of the hill) by essentially looking at the local gradient and following the curve in the direction of the steepest ascent.
- Problem: easily trapped in a local optimum (local small hill top)
Hill Climbing Algorithm

```plaintext
currentNode = startNode;
loop do
  L = NEIGHBORS(currentNode);
  nextEval = -INF;
  nextNode = NULL;
  for all x in L
    if (EVAL(x) > nextEval)
      nextNode = x;
      nextEval = EVAL(x);
    end if
  end for
  if nextEval <= EVAL(currentNode)
    //Return current node since no better neighbors exist
    return currentNode;
  end if
  currentNode = nextNode;
end do
```

Note: Solutions are very sensitive to the search starting position.

Source: http://en.wikipedia.org/wiki/Hill_climbing
Algorithm Comparison

Let’s compare UCS with Hill climbing

- We will find that UCS will use backtracking to recover from an initial wrong guess.
- We will also find that Hill Climbing will get stuck with its initial bad guess and will compute a sub-optimal solution.
UCS Algorithm

OPEN = [initial state]
while OPEN is not empty
do
0. Sort OPEN according to g(n).
1. Remove the best node from OPEN, call it n.
2. If n is the goal state, return n as the solution.
3. Create n's successors.
4. For each successor do:
   add it to OPEN.
done

g(n) = cumulative cost of path so far.
currentNode = startNode;
loop do
    L = NEIGHBORS(currentNode);
    nextCost = INF;
    nextNode = NULL;
    for all x in L
        if (HOPCOST(x) < nextCost)
            nextNode = x;
            nextCost = HOPCOST(x);
        end if
    end for
    if nextNode == targetNode
        return “computed path from startNode to nextNode”;
    end if
    currentNode = nextNode;
end do

Note: the algorithm has been slightly modified for minimum path finding in a graph.
Observations

- Greedy algorithms can save us a lot of computation (no sorting of the priority queue necessary, no exploration of other alternatives)
- But there are no guarantees of finding the (optimal) solution.
Use
1. Hill Climbing (graphs)
2. UCS
3. Best-FS

to find the cheapest Path from A to G.

Note: for h(n) in Best-FS use the number of remaining nodes in the path n