CS F364
Design & Analysis of Algorithms

ALGORITHMS – DESIGN TECHNIQUES

- Exact Solutions
  - Search
    - Approach
    - Modeling as a Graph problem

Searching for solutions

- One “algorithmic” approach for solving hard problems is to “search” the solution space:
  1. Characterize “feasible” solutions:
     1. Characterize “feasible” solutions for sub- problems
  2. Construct “feasible” solutions out of “feasible” solutions for sub-problems
  3. Test such “feasible” solutions for “validity” or “acceptability”.

Search – How Does It Work?

- Given that it is difficult (or *it is not known how*) to "compute" a solution directly:
  - the algorithm is intended to search the space of solutions by ***brute-force***:
    - test all possible solutions one by one;
    - construct larger solutions from smaller solutions by exploring all possible incremental choices

Search Modeled As A Graph Problem

- Solution space can be viewed as a *directed graph*:
  - Vertices are "feasible, partial solutions" (i.e. for sub-problems)
  - An edge leads from one partial solution to another (more constrained solution)
    - Number of edges in a vertex correspond to the number of "choices" available for constructing a feasible solution
- Searching the space for a solution is to

  traverse the graph until a "valid" solution is found

Search Modeled As A Graph Problem [2]

- Note that although the solution space can be defined it is not a "given"
  - i.e. the solution space (i.e. the solution graph) has to be generated on-the-fly:
    - Start with an initial partial solution (i.e. root vertex)
    - Identify choices that can be made and for each choice generate the next partial solution (i.e. adjacent vertex)

Search Modeled As A Graph Problem - Representation

- Observation:
  - If each edge is monotonic (i.e. from a "less constrained" partial solution to a "more constrained" partial solution)
    - then the graph won't have cycles.
- Question:
  - Should the graph be represented as a *tree* or a *DAG*?
- Hints:
  - Do you keep track of visited vertices (i.e. previously seen / examined solutions) ?
  - Do you have sufficient memory to keep track of all visited vertices?

Search Modeled As Graph Traversal

- If searching the solution space is modeled as graph traversal:
  - what should be the traversal technique used?
- Questions:
  - When is Breadth-First search suitable?
  - When is Depth-First search suitable?
- Graph Traversal requires a "to be expanded" data structure:
  - A queue in the case of BFS
  - A stack in the case of DFS
- If solutions can be weighted (for potential "to lead to good solutions"), then
  - one can choose Best-First-Search i.e.
    - expand the vertex with maximum potential
- Example Problems??