Developing Algorithms

Developing Algorithms

Strategies –

- realize your problem is another well-known problem in disguise
  - it is searching or sorting
  - there's a data structure for that
  - it is a graph problem
- develop a new algorithm
  - divide-and-conquer
  - iterative
  - series of choices – greedy, recursive backtracking, dynamic programming

Modeling With Graphs

Things to decide –

- what the vertices represent
  - vertices represent things
- what the edges represent
  - edges represent connections (relationships) between pairs of things
    - does the relationship simply exist or not exist? – indicated by presence or absence of edge
    - does the relationship have a strength? – modeled by edge weights
    - is the relationship one-way? – indicated by directed or undirected edges
- what a solution to the original problem looks like in the graph
  - ideally it is a concept for which there is a known algorithm
    - e.g. reachability, shortest path, minimum spanning tree, cut vertices

Modeling With Graphs

As you think about graph algorithms, identify –

- the flavor of graph
  - connected or not connected
  - simple or containing self loops and/or multiedges
  - sparse vs dense
  - cyclic vs acyclic
  - weighted vs unweighted
    - if weighted, are there negative weights? negative weight cycles?
  - embedded vs topological
    - is there a particular ordering of incident edges for each vertex?
  - explicit vs implicit
    - does a graph data structure need to be built?

these properties are consequences of the modeling choices rather than independent decisions

they are important to recognize because they can affect the algorithms used and the choice of implementation for the graph structure (and other data structures)

e.g. representing an embedded graph would require an adjacency list implementation
e.g. (recursive) DFS doesn't require explicit graph structure, but BFS and Dijkstra's do (at least for vertices)