Recitation 18: Quiz 2 Review

Scope

- Quiz 1 material fair game but explicitly **not emphasized**
- 5 lectures on graphs, 2 lectures on dynamic programming
- 2.75 problem sets on graphs, 0.25 problem sets on dynamic programming

Graph Problems

- Graph exploration, count connected components
- Topological sort
- Cycle detection, negative weight cycle detection
- Single Source Shortest Paths (SSSP), Relaxation framework

Restrictions		SSSP Algorithm		
Graph	Weights	Name	Running Time $O(\cdot)$	How it works
DAG	Any	DAG Relaxation	$\lvert V\rvert + \lvert E\rvert$	Relax in topological order
General	Unweighted	BFS	$\lvert V\rvert + \lvert E\rvert$	Relax level by level
General	Nonnegative	Dijkstra	$\lvert V\rvert \log \lvert V\rvert + \lvert E\rvert$	Relax in priority order
General	Any	Bellman-Ford	$\lvert V\rvert\lvert E\rvert$	Relax in $\lvert V\rvert$ rounds

- All Pairs Shortest Paths (APSP)
  - Run a SSSP algorithm $|V|$ times
  - Johnson's solves APSP with negative weights in $O(|V|^2 \log |V| + |V||E|)$ time

Graph Problem Strategies

- Explicitly describe a graph in terms of problem parameters
- Convert problem into finding a shortest path, cycle, topo. sort, conn. comps., etc.
- May help to duplicate graph vertices to encode additional information
- May help to add auxiliary vertices/edges to graph

Recursive Paradigms

Class	Subproblem Dependency Graph
Brute Force	Star
Decrease & Conquer	Chain
Divide & Conquer	Tree
Dynamic Programming	DAG (Overlapping subproblems)

Dynamic Programming Steps (SR. BST)

1. Define **Subproblems** subproblem $x \in X$
   - Describe the meaning of a subproblem **in words**, in terms of parameters
   - Often subsets of input: prefixes, suffixes, contiguous subsequences
   - Often record partial state: add subproblems by incrementing some auxiliary variables
2. **Relate** Subproblems $x(i) = f(x(j), \ldots)$ for one or more $j < i$
   - State topological order to argue relations are acyclic and form a DAG
3. Identify **Base** Cases
   - State solutions for all reachable independent subproblems
4. Compute **Solution** from Subproblems
   - Compute subproblems via top-down memoized recursion or bottom-up
   - State how to compute solution from subproblems (possibly via parent pointers)
5. Analyze Running **Time**
   - $\sum_{x \in X} \text{work}(x)$, or if $\text{work}(x) = W$ for all $x \in X$, then $|X| \times W$