02-713 Review
Midterm 1

* I took some images from Prof Kingsford’s slides
• Disclaimer: I have not seen the midterm you will take.

• But we would like to recap some important points and try our best to answer questions you may have.
Anything is fair game, but ...

- I always like using slides as a starting point for what could be asked.
- See what theorems could be handy w/ each algorithm: they are useful tools in the design of algorithms.
- Branch out to the book for sup info.

For the test ...

- On anything that’s more open-ended, describe your primary strategy.
- Pseudocode is good for being more specific.
• Basic data structures
  (Arrays, Lists, Trees, Graphs, Heaps, Union-Find)
• Analyze asymptotic run-time of an algorithm (Big Oh)
• 2 sorting algorithms
  (HeapSort, MergeSort)
• 3 Optimal Algorithms for MST
  (Prim’s, Reverse-Delete, Kruskal’s)
• 5 Algorithms based on Tree Growing
  (Prim’s, BFS, DFS, Dijkstra’s, A*)
• Preview of dynamic programming
  (Bellman-Ford)
• Intro to Divide and Conquer
  (Counting Inversions, Closest Points)
Heaps

- Finding the minimum item takes constant time.
- Tree property: children are always greater in value than the parent.
- Insertion and deletion take $\log(n)$ time and preserve this property via “sift ups” and “sift downs”.
- Heapsort: Keep grabbing the minimum element from the top of the heap and delete it.
Union-Find

- Given: a partition of a set (collection of sets are disjoint)
- Goal: efficiently find which set an element belongs to and efficiently merge them together.
- During a union, updating the array that says which set an element is a member of can loop through entire array.
- However, \( k \) union operations is bounded by \( k \log(k) \) (set sizes “double”)
MST Key Points (1)

- Find the tree in a graph with the minimum total edge cost.
- **Prim’s algorithm**: tree grow by adding the lowest cost edge to the tree.
- Cut property can be used to prove their correctnesses.

**Theorem (MST Cut Property).** Let $S$ be a subset of nodes, with $|S| \geq 1$ and $|S| < n$. Every MST contains the edge $e = \{v, w\}$ with $v \in S$ and $w \in V - S$ that has minimum weight.
MST Key Points (2)

- **Reverse-Delete algorithm**: Remove edges and avoid disconnecting the graph.

- **Kruskal’s algorithm**: Add edges avoiding cycles.

- Cycle property can be used to prove the correctness of both of these.

**Theorem (Cycle Property)**. Let $C$ be a cycle in $G$. Let $e = (u, v)$ be the edge with maximum weight on $C$. Then $e$ is not in any MST of $G$. 
Big Oh

- Big Oh used often in practice to describe runtime of algorithms
- Ideally find the “best” Oh (i.e. the lowest upper bound)
- When proving statements in asymptotic analysis, start with the definition and make sure you specify constants

**Definition (O).** A runtime $T(n)$ is $O(f(n))$ if there exist constants $n_0 \geq 0$ and $c > 0$ such that:

$$T(n) \leq cf(n) \quad \text{for all } n \geq n_0$$
BFS

• Traverse the graph in ‘layers’
• Add visited nodes to queue (FIFO, TreeGrowing process *earliest* discovered node)

Non-tree edge property:

**Theorem.** Choose \( x \in L_i \) and \( y \in L_j \) such that \( \{x, y\} \) is an edge in undirected graph \( G \). Then \( i \) and \( j \) differ by at most 1.
DFS

- Keep walking down a path until you are forced to backtrack.
- Add edges to a stack
  (LIFO, TreeGrowing: process *most recently* discovered node)

Non-tree edge property:

**Theorem.** Let $x$ and $y$ be nodes in the DFS tree $T_G$ such that $\{x, y\}$ is an edge in undirected graph $G$. Then one of $x$ or $y$ is an ancestor of the other in $T_G$. 
DAG

- Key property: Directed, no cycles!
- You can ‘topologically sort a DAG’ so that there are no ‘backwards’ edges.

**Theorem.** Every DAG contains a vertex with no incoming edges.
Dijkstra

- Tree Growing: instead of choosing min-weight edge like Prim’s, choose the minimum of $d(u) + \text{length}(u,v)$
- Only positive weights
Bellman-Ford

- Handles negative weights (assume no negative cycles)
- 1 pass applies the Ford rule to all edges
- Can make at most \( n-1 \) passes
- Not ‘greedy’ like Dijkstra’s algorithm

**Ford step.** Find an edge \((u, v)\) such that

\[
dist_s(u) + d(u, v) \leq dist_s(v)
\]

and set \( dist_s(v) = dist_s(u) + d(u, v) \).
A*

- Another Tree Growing, but with a goal node and a heuristic to help get us there
- Heuristic guarantees we will find optimal solution
- Heuristic must be less than true distance
- Ideally it is closer to the real distance
Divide and Conquer

- Try to divide into ‘equal’ parts to get $O(n \log n)$
- Assume the problem is solved for each part and the trick is to design an efficient merge step.
- Need a base case for the ‘leaf’ of the recursion tree.