|  |  |  |
| --- | --- | --- |
| Depth first    recursively explore one part before going back to the other parts not yet explored  But depth-first can use less space in finding a path  –If longest path in the graph is p  and highest out-degree is d then DFS stack never has more than  d\*p elements  –But a queue for BFS may hold O(|V|) nodes | Breadth first  explore areas closer to the start node first  Breadth-first always finds shortest paths, i.e., “optimal solutions”  –Better for “what is the shortest path from x to y” | Graph representations (ADTs) |
| How to solve: Kruskal's algorithm and corresponding complexities      READ THE NUMBERS OF EDGES      How to solve: **Prims Spanning tree**    Algorithm    **Sort**  An algorithm that solves this computational problem is called a Comparison Sort.  Merge Sort  Values should be sorted first  Quick Sort    Heap sort – array and put in tree then compare root with child. If child is larger then becomes root. Keep repeating.  **Previous Tests** | How to solve: Shortest path with Dijkstra's algorithm      How to solve: spanning trees  Approach 1    Skipped some Second Approach  How to solve: Topological sort  Uses – Figuring out how to finish your degree, computing the order in which to recompute cells in a spreadsheet. Determine the order to compile files using a makefile. Using dependency graph to find an order of execution  Jagannadha Chidella      Bucket Sort  Radix Sort | **Density/Sparsity** s  **Adjacency Matrix**    **Adjacency List**  **NP-Completeness (poly. vs exp.)**  For some problems, we don’t know if any efficient non exponential growth solution to problems exists  Any problem that runs on a non-deterministic machine in polynomial time is in class NP.    **NP-Completeness: the hardest problems in NP**  **REDUCTION**  The general belief is that there is no efficient alogrithm for any NP-complete problem, but no proof of that belief is known.    **Undirected Graphs**  **Directed Graphs**s  **Weighted Graphs**  **Paths and Cycles**  A cycle is a path that begins and ends at the same node(Vo == Vn)    **More Graph**  **RANDOM INFORMATION**  **Sorting massive** data – needs sorting algorithms that minimize disk/tape access time:  **Quick and heapsort** both jump all over the array, leading to expensive random disk accesses.  **Merge Sorts** scans linearly through arrays, leading to relatively efficient sequential disk access  **In-place** – sorted items occupy the same space as the original items.  **Stable** – Items in input with same value end up in the same order as when they began |

|  |  |  |
| --- | --- | --- |
| 4)Does Prim’s algorithm work when there are negative weights in the graph? Prove why or why not. Either use a counter example to prove it doesn’t work. Or explain why it works. It does because it always picks smallest weighted edge until it creates an MST  5) describe algorithm to find minimum number of edges. Run a DFS based algorithm to number the vertices in the order in which they are visited during DFS. Use this numbering to detect back edges and keep count of them. The total number of back edges is the minimum number of edges. |  |  |