Name: Laxmi Kandel ­­­ Date: 03/10/2020

1. Define each of the following types of algorithms:

|  |  |
| --- | --- |
| Strategy |  |
| Brute Force | It is an approach that searches exhaustively through a problem space for an answer. There are no shortcuts to improve this algorithm. It is considered as one of the easiest approaches to apply and is useful only for solving small instance problem.  Example: Travelling salesman problem  Search takes O(n!) |
| Greedy | A problem-solving technique that tries to always make the choice that seems to be the best at that moment. These types of algorithm can produce incorrect results sometimes.  Examples: Graph algorithms   * Dijkstra’s shortest path algorithm * Prims MST algorithm * Kruskal’s MST algorithm * Huffman encoding * Coin change problem |
| Divide & Conquer | This algorithm breaks up their input into several sub-problems. The solution to each problem is used to solve the original problem. |
| Decrease & Conquer | Reduces a problem instance to a smaller instance of the same problem, solves the smaller instance and then extends the solution of the smaller instance to obtain a solution to the original problem. |
| Transform & Conquer | The techniques solve a problem by a transformation to a simpler/more convenient instance of the same problem, to a different representation of the same instance (representation change) and to a different problem for which an algorithm is already available (problem reduction). |

**Brute Force Algorithms**

1. State the Travelling Salesman Problem (TSP):

|  |
| --- |
| * Generate all permutations of n cities. * Iterate over each permutation and save the minimum distance. * Return the permutation with minimum cost. |

1. Solving the TSP requires finding a **Hamiltonian** cycle, which is a cycle in the graph that visits each vertex exactly once.
2. This is similar to a **Euler** cycle, which is a cycle in the graph that includes all vertices but does not visit an edge more than once.
3. The exhaustive search solution to the TSP runs in **O(n!)** time.
4. Solve the TSP problem below by hand.

100

125

75

50

300

100

75

125

50

100

Starting from A

A->E ->75

E->D-> 50

D->B->75

B->C->50

C->A->300 (return back to A)

Total cost: 550

1. How many candidate solutions would you need to examine to prove exhaustively that your solution is correct? **all permutations of n where n is number vertices (all edges of n).**

**Greedy Algorithms**

1. We have already covered the following algorithms that show greedy behavior:

* Dijkstra’s shortest path algorithm
* Prims MST algorithm
* Kruskal’s MST algorithm
* Huffman coding
* Coin change problem

1. **Coin Changing Problem:** Given a set of coin denominations: and an amount N, what is the minimum number of coins required to make the value N?

Write the steps of a greedy algorithm (in pseudocode) that determines the minimum number of coins needed to produce the value N.

|  |
| --- |
| **Let a = N be the amount remaining**  **Let c = 0 be the coin count**  **Repeat until a = 0**  **Take the largest denomination c1 such that c1<=a**  **Increment c**  **Set a = a-c1** |

1. Give a set of coin denominations for which our greedy algorithm does not produce the correct result.

**When denominations are {1,5,11,15} the value need to create 33 will give incorrect results If greedy algorithm is used**

**Correct: 11-11-11 (Required 3 coins)**

**Incorrect: 15-15-1-1-1 (Required 5 coins)**

**Divide & Conquer Algorithms**

1. We have already covered the following divide & conquer algorithms:

* Merge sort
* Qick sort
* Strassen’s Matrix Multiplication

**Decrease & Conquer Algorithms**

1. For each of the following decrease & conquer algorithms, mark whether they decrease a problem by a constant factor, a variable factor, or by one.

|  |  |
| --- | --- |
| Algorithm | Decrease-by |
| Insertion Sort | By one |
| Binary Search | Constant factor |
| Euclid’s GCD Algorithm | a variable factor |

1. State the definition of a Topological Sort:

|  |
| --- |
| Finds a linear ordering of vertices in a graph such that for every directed edge (u, v), vertex u comes before v in the ordering. This algorithm only works with directed and acyclic graph.  Applications:   * Anything related to prerequisites * Class scheduling * Job scheduling |

1. To perform the Topological Sort on a graph, the graph must be a DAG, which is short for

**directed- acyclic.**

1. To support the Topological Sort algorithm, we maintain a heap of **Min-heap** sorted by

**In-degree**.

1. Show the steps of the topological sort algorithm for the following graph. Make sure to show the heap of vertices at each step of the algorithm and the ordering of vertices as the result.

Min heap

(Underwear,0)

(Trousers,1)

(Belt, 2)

(Shirt ,0)

(Tie,1)

(Jacket,1)

(Socks,0)

(Shoes,3)

(Watch,0)

Remove min (Underwear,0)

Remove min (Shirt,0)

Remove min (Socks,0)

Remove min (Watch,0)

Remove min (Trouser,1)

Remove min (Tie,1)

Remove min (Jacket,1)

Remove min (Belt,2)

Remove min (Shoes,3)

Results: Underwear, Shirt, Socks, Watch, Trouser, Tie, Jacket, Belt, Shoes

**Transform & Conquer Algorithms**

1. The following are all examples of Transform & Conquer algorithms:

|  |  |
| --- | --- |
| Algorithm | Description |
| Presorting | Presort an array and then use the structure of the sorted elements to solve a problem.   * Finding the smallest, largest, median or kth-largest value. * Finding element uniqueness. |
| Heaps | For implementing the Priority Queue ADT   * Transform array of elements into a complete binary tree with a heap ordering property. |
| Binary search trees | Used to store and retrieve elements with reduced search times. |
| Heap-Sort routine | Transforms an array into a heap and sorts the elements using the deleteMin() operation. |