course_content

Data Structures and Algorithms

Greedy Algorithms

Week 12

top

W12: Greedy Algorithms

Time:
10 min

Quiz Time

bit.ly/DSA1920Quiz10

Deadline: Friday 24th April (11.59pm)

Time:
10 min

Greedy Algorithms

- An algorithm is a set of steps that help achieve a desired outcome
- A greedy algorithm is one where in each step, one takes the option that looks obviously the best. It makes ‘greedy choices’ at each step
- REMEMBER: Greedy algorithms don’t always work
- Animals often live using the greedy approach - what do I need to do write now?
- Humans (mostly) try to think in the long term so will often take steps that are not immediately optimal (e.g. you are studying right now to help you in the long term)
- However, it can sometimes be useful to try a greedy approach because it may

Time:
10 min

Greedy Algorithms - Simple Example

- You have n tasks to do and a sorted array containing time taken for each task.
- You have a limited amount of time but you want to do as many as possible.
- The ‘greedy approach’ is to start with the task taking the least amount of time and then to repeatedly take the shortest time task until time runs out. This maximises the number of tasks you are able to complete so this is a correct algorithm.
- However, if you were given an unsorted array, then repeatedly finding the smallest element in your array would take O(n) time and you would potentially have to do this many times so it would be inefficient and it would be more

Time:
10 min

KnapSack Problem

- In this problem - we have a bag (sack) that we want to fill.. We are given a list of possible items that we can fill it up with that take up a certain size and are worth a certain amount. We want to optimise value for limited total size E.g. 25.
- E.g {A:{size:9,price:9},B:{size: 7,price: 6},C:{size 22: price: 12},D:{size:10,price:9}}
- One greedy approach is to sort the items by highest price which allows us to take C
- Another is to sort the items by lowest size which allows us to take B & A.
- The total value of these is 12 and 15 but neither of these is optimal.
- **BEFORE MOVING ON**, try to find the optimal solution and suggest an alternative approach that gets us to the optimal solution. Discuss in pairs to find a way. Hint: Recursion/Dynamic Programming may be helpful here.

Time:
10 min

Fractional KnapSack Problem

- In the fractional knapsack problem, we can also take ‘fractions’ of an object. For example ½ of object C will be size 11 and worth 6.
- Here, instead of ordering by size or by price, we create a new metric call ‘value’ which will be price divided by size. This is really what we want to sort by because it means that we optimise the price that each unit of space is worth.
- Ordering it in this way and then applying the greedy approach, we get the solution of A, D and then 6/7 of B which has a total price of 23 1/7 which is optimal.
- It also helps us get the solution of the 0-1 KnapSack Problem (where we have to take full items rather than fractional values) which is A and then D though this doesn’t always work. The 0-1 version is much harder than the fractional version.

Time:
10 min

Huffman Encoding Algorithm

- Huffman Encoding- ‘an efficient method of compressing data without losing information’
- Information is stored in strings of bits 0 or 1.
- Uses variable length strings rather than fixed lengths trings
- There is no ambiguity - there are no two strings which have the same meaning
- It is efficient - uses as few bits as possible to encode the information
- Go through this activity on Brilliant explaining how it works.
- See if you can understand where the ‘Greedy Approach’ comes into this application
- Watch this video for another explanation

Time:
10 min

Djikstra's Algorithm (Graphs)

- Dijkstra's Algorithm helps us find the shortest path from a home node to any other node. The same algorithm actually finds a 'tree' of shortest paths to every other node (minimal spanning tree). It works on weighted graphs (directed and undirected).

1. Initialise at your home node with distance 0 and all others as distance infinity.
2. Create a set S of all reached nodes which initially has just your home node.
3. In each step, find all nodes not in S reachable directly (via 1 edge) from your set S and add the one with the minimum distance (from home) to your set S. For example if your home is A, B is also in S with distance 3 and there are 2 reachable edges: A->C length 5 and B->D with length 3, you should add C because the total distance is 5 while B->D is 3+3=6.
4. Do this repeatedly until you either reach the node you want or until the set of nodes not

Time:
10 min

Dijkstra's Algorithm (Graphs)

- The algorithm is greedy because at each step, it just looks for the one with the next minimal distance.
- It is equivalent to slowly increasing the ‘radius of reach’ from your home nodes until you’ve found what you want but it is faster because if your distances are high (e.g. 1000), you jump straight from 0 to 1000 if there are no distances in between that you can reach.
- Variants of Dijkstra’s algorithm are used in all of your mapping software and many other networks.
- Read more here, here or here
- Watch videos here,

Time:
15 min

Extra Videos and Readings

- Greedy Algorithms - Hacker Earth
- Greedy Algorithms - Free Code Camp (with Lecture Scheduling Problem)
- Activity Selection Problem - Geeks for Geeks
- Prims Minimum Spanning Tree- Geeks for Greeks
- Prims Minimum Spanning Tree - RuneStone Course Book

Time:
1 min

Questions?....Office Hours!