

Day1 - OCT 1

01 October 2020 20:15

Today I searched for the graph algorithms which will help us to build a story upon that. As we are going to have a detective type of character he has to find a person who is going along streets and the detective will think of these algorithms in real life and chases the person in this we can apply Dijkstra and some other efficient algos to find the person as real life challenges are very tough we can't stick to one algo, we have to explore on various graph algos.

Day2 –OCT 5

05 October 2020 16:30

After I searched for a story that uses Merge sort Algo. I got a story in which there is a villain and hero like sherlock . In which the villain setting the bombs in different places and controlling them from a remote place. The people who belong to the villain are setting the bombs and the people who are setting the bombs have no idea about how much intensity should they keep in that bomb (like how much amount of RDX should keep in the bomb) . These instructions has to give by villain. He has a touchscreen computer on that he has the places as boxes and intensity of bombs. The villain actually set the number of swapping (like say there are different boxes and balls if we take the ball from box 1 and keep in box 2 and taking the ball in box2 keep in box1 is considered as one swap here the boxes are places he want to destroy and balls are the intensity of a bomb and here the places arranged in a descending order of crowd at that place). At first the computer there is no arrangement of intensities and villain will assign the intensities directly proportional to crowd at that place. By arranging this intensities there are only (let no of places is n) n swapping left. Our hero will find the place where the villain is operating the bombs and he will go to that place. By that time the hero arrives the villain sets the intensities he arrests villain but Hero does not how to stop the bombs and this place is so far to rescue the bombs and people and the villain also sets timer. Only way to rescue is to vacate the places . But in a crowd place they cannot vacate easier so. Our hero came up with an Idea that he will arrange the intensities inversely proportional to crowd so that in crowd places the affect bomb is less and in that lesser crowd place also the harm is less because of less people. So he wants to arrange them. As the places are arrange in descending order of crowd this problem becomes sorting problem. Hero came up with brute force idea I.e. normal sorting . But he realised that the arrangement would take more than n steps, so he can't do that sorting, later he came up with merge sort algorithm idea in which the algorithm has n Comparisions I.e. corresponds to n swaps (As we know the no steps taken in merge sort algorithm is $n \log n$ we know that $\log n$ is for splitting the array. Here they is no need of splitting the only thing comparison so only at most n steps may require to be in correct arrangement. So he use the merge sort algo and arrange the intensities and saves the lot of people.

Day3-OCT 22

22 October 2020 00:51

Today I helped arjun in explanation of algos to put in appendix as he is doing illustrations. And I made the appendix for sorting algos I attached that file here.



Appendxf
orepisode...

Comparisons with selection, insertion, bubble sorts with Merge sort:

Comparison between selection sort and merge sort:

As in selection sort we will take a part of sub array which is sorted, and other part is unsorted every time we are taking the minimum in the unsorted sub array and keep it in the sorted array so for this process the time for finding minimum in i th iteration is $O(i)$ so for n iterations is $O(n^2)$ and in merge sort for dividing array takes $\log n$ and it combines n elements so $O(n)$ is needed so the total time is $O(n \log n)$ which is better than selection sort.

Comparison between insertion and merge sort:

As in insertion sort we are iterating through the whole array and while iterating we will compare the current element with the preceding element if preceding is greater, then swap like that we must continue this swap up to the preceding value is lesser as for each iteration we are doing this swap so at i th iteration at worst case the swapping will be done for $i-1$ times so as summing up this algo will take $O(n^2)$ and we see that merge sort will take $O(n \log n)$. So, merge sort is better than selection sort.

Comparison between bubble and merge sort:

As in bubble sort we are iterating n times and, in each pass, (iteration) we will go through the whole array and swap the adjacent elements if required so for doing this swapping across the whole array takes $O(n)$ time at worst and for n iterations the time is $O(n^2)$ as merge sort takes $O(n \log n)$ so merge is better than bubble sort.

Discussing story

08 November 2020 23:21

I and Trinadh are discussing stories based on algos, At first we thought about A villain will create a virus and he will inject it into a water canal which is used by the people in city . The thing is he will choose the canal to be injected in a way the water has max flow to the city , Here we thought use a maxflow algorithm (Graphs) But this algo did not work for this type of situation, we changed the story that the villain will inject the virus into a dog, after some time the virus spreads, and now hero will enter into scene here we made two choices in story one is there is main underground water crossing now hero wants to reach that place in a less time and have inject antidote in to water so it can spread hole over the city , In this we can use the shortest path algorithm to reach this place from the police control , And the other is there are n numbered water plants in the city they are the main resources of water to the city , The thing is hero has to go to the water plants and gives antidotes to all water plants, Here the hero should do this in a less time so he has to cover all the water plants in shortest time , I.e. there are many orders of the water plants to be visited he should pick a best order in a way he goes to all water plants in short time to save people. Here we have to further research about this situation to get suitable algo for this, And also we discussed another type of the same story , Here villain has less concentration of virus in order effect the people he will release in to water canals but the concentration will not be enough to kill people so he though to dehydrate some water from canals used some dehydrating agents , here we want to discuss further about this situation that where should we keep dehydrating agents or should the dehydrating agents releases as a concentration thing and at each division of water canal the concentration will divide like that , And we will come up with a algo for doing this or the above story.

Discussing Episode 2

10 November 2020 17:58

I and Trinadh discussed about how to implement travelling sales men problem into episode 2 As the naïve of doing tsp takes more time so By Dynamic programming we can use the algorithm with less time. We thought implementation of Dynamic programming in real life is somewhat difficult, And upto 7 nodes naïves gives less time to compute, After 7 nodes Dp solution takes less time, I saw the proof of TSP and the code using Dp

I have listened to Coursera lectures about TSP and how did they came about the dp and why this dp is best. In that they explained how to take the dp array and what are the advantages of taking it like that, The naïve solution of TSP has $O(n!)$ complexity, when we use dp the complexity decreases to $O(n^2 \cdot 2^n)$.

The dp uses the a set of points which are covered only once in shortest path from source to j represented as $L_{s,j} = \min\{L_{s-\{j\},k} + C_{k,j}\}$ where s is the set of points and j is the destination.

In story we also have a n nodes and have to do TSP by the hero, At first we thought hero will assume some permutations and will find path, But thus takes time, So we proposed Dp solution, The thing is visualisation of DP using recursion is difficult for hero, Like we cannot have a physical significance to explain by DP. It becomes difficult to explain, So we came with a solution i.e. mind palace, This is used by sherlock in movies, this like virtual world and he will do anything there like imagination, So we thought to give the TSP thing So he will think about it easily because, He can visualise dp stuff and soon, so as it is hypothetical thing so he will solve problem by visualising Dp in mind palace and follow that min cost path.

This is the link to TSP Coursera video:

<https://www.coursera.org/lecture/algorithms-npcomplete/the-traveling-salesman-problem%5C-49MkW>

Code for Episode2

11 November 2020 23:32

As we thought about mind palace, I thought it would be easy to show that dp stuff in his mind and finds min cost path, Here we have 11 nodes and their respective edge weights, we know that the dp complexity of TSP is $O(n^2 \cdot 2^n)$. That if we have to visualize dp solution in his mind, We have to draw a lot of diagrams to show the approach because there are 11 nodes and may we have to draw more 500 graphs approx, So like showing visualization on paper has to draw so many, so Arjun suggested to code the TSP which prints min cost and optimal path too, I coded it with help from references and explanation in coursera and made code for TSP, which gives optimal path and cost of it, In code the edge values are hardcoded because we are using only one test case here so I hard coded it, And I have taken the nodes as cities in USA, And edge weights are random, difficult to find all exact edge weights, So gave random weights for then. And the code is on my github repo, And I made a illustration of the graph that initially we have with all nodes and edges, and I made a illustration according to optimal path we got by running TSP code.

The link to git hub is :

<https://github.com/sreeharsha2002/TSPcode>

I will paste the Readme here:



README

TSPcode

- Code for Tsp
- The minn.cpp is the code for Travelling sailsmen problem
- In the code I have recorded the optimal path of the Sailsmen problem and minimum cost of the optimal path.
- In the code I have fixed the node numbers to the city names in USA

```
1-> Amsterdam
2->New York
3->Hudson
4->Middletown
5->Mechanicville
6->Rochester
7->Kingston
8->Norwich
9->Niagara Falls
10->Jamestown
11->Cortland
```

- At first compile the code "g++ minn.cpp"
- And Run the code "./a.out"
- the output u will get is the minimum cost of the optimal path and path will be printed in both numbers and the names in the path.
- And I hard coded the edge weights between each nodes. and I have matrix for that in code.
- The matrix is

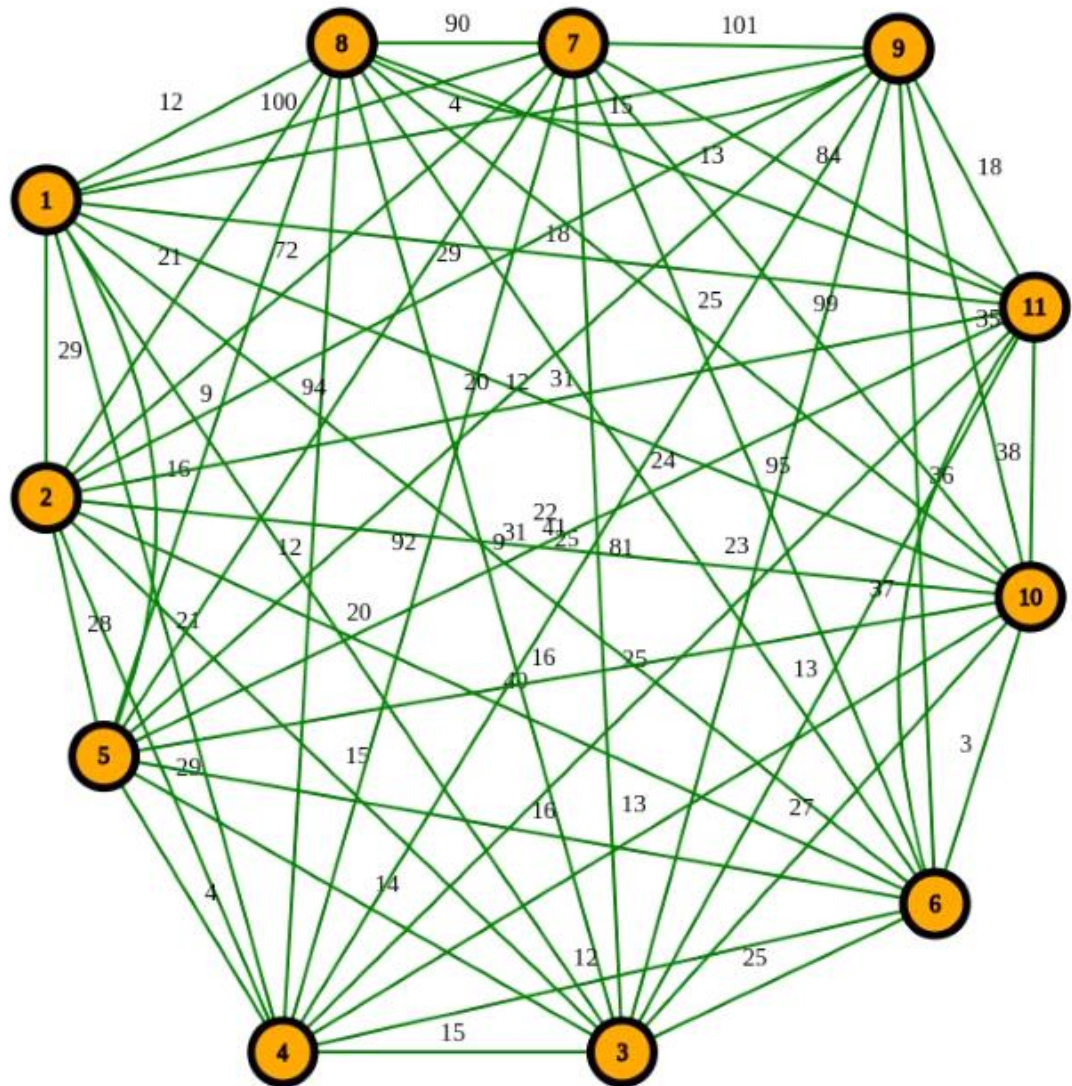
numbers and the names in the path.

- And I hard coded the edge weights between each nodes. and I have matrix for that in code.
- The matrix is

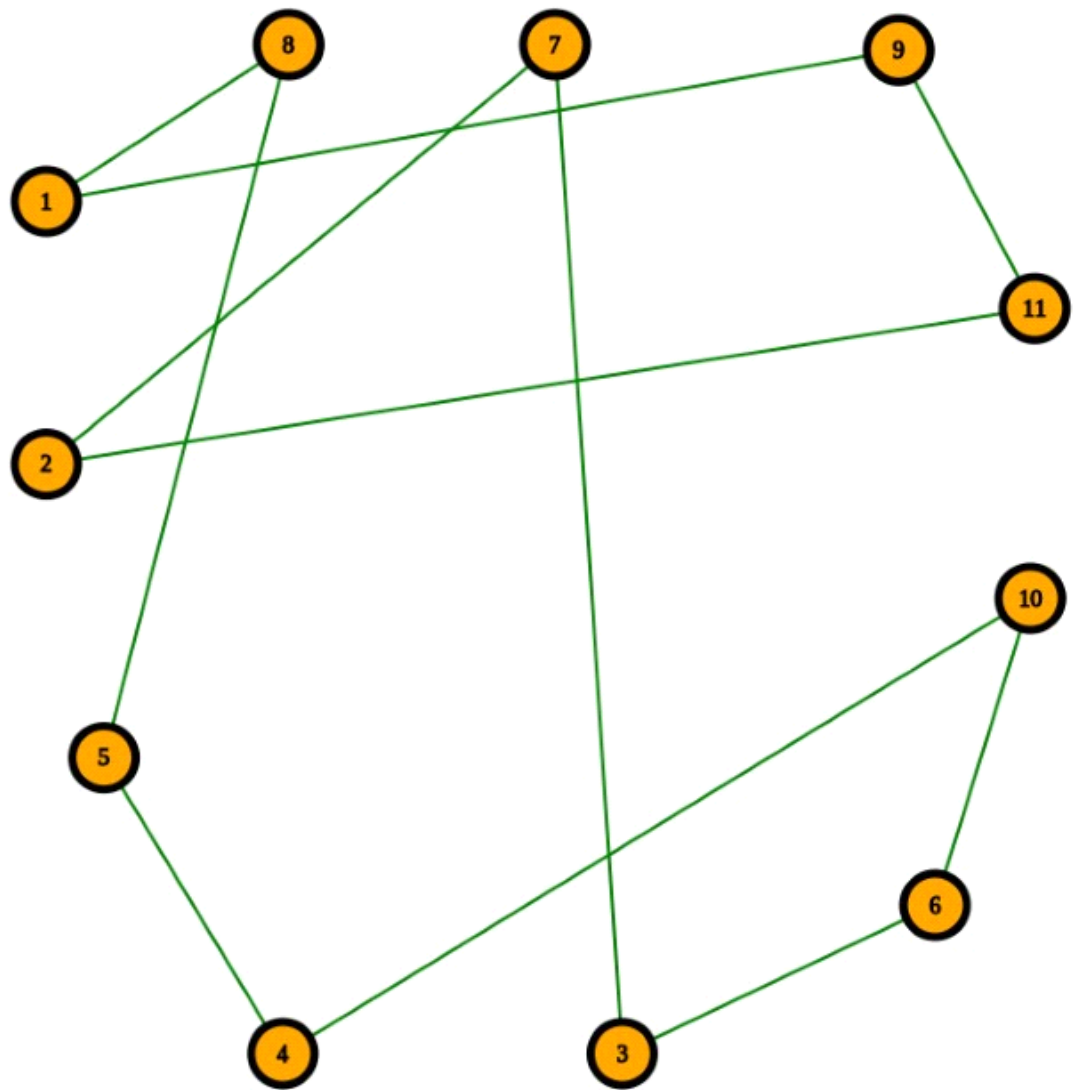
		1	2	3	4	5	6	7	8	9	10	11				
1	INF	29		20		21		16		31		100		12		4
31	18															
2	29	INF	15			29		28		40		72		21		29
41	12															
3	20		15	INF	15			14		25		81		9		23
27	13															
4	21		29		15	INF	4			12		92		12		25
13	25															
5	16		28		14		4	INF		16		94		9		20
16	22															
6	31		40		25		12		16	INF		95		24		36
3	37															
7	100		72		81		92		94		95	INF		90		101
99	84															
8	12		21		9		12		9		24		90	INF		15
25	13															
9	4		29		23		25		20		36		101		15	INF
35	18															
10	31		41		27		13		16		3		99		25	35

INF	38								
11	18	12	13	25	22	37	84	13	18
38	INF								

- INF is the maximum number in int
- Actual we should give 0 for (1,1) but as here we are computing minimum we should not have a self loop so we take it as INF.
- The actual graph is this:



- After running the code the optimal path graph is this:



Reading Algo for Episode 3 on knapsack Heuristic -1

12 November 2020 18:11

As I watched a Coursera video last time to understand the travelling salesmen problem with proofs and techniques, So I continued the next week in that course, I saw knapsack algos, which were new to me, And I am very excited to learn those

We know that the knapsack problem is to maximise the value and the sum of weights to be \leq max capacity of knapsack. Let v_i be the value of an object and w_i is the weight of the object and W is the max capacity of the knapsack.

Greedy Knapsack Heuristic:

As the name is greedy at first, we sort the objects by its values in a decreasing order, so we can pick the maximum value and add to knapsack, we can clearly say that this will fail for most testcases. So as this fails, we will go for another algo, In this,

Step1: sorting the objects by the ratio of values and weights (v_i/w_i) in a decreasing order for their ratios, this will ensure value per unit weight.

Step2: Taking the maximum ratio in the sort and adding it to the knapsack, say at k objects are filled in knapsack, and for $k+1$ th object there is no space here.

This algo also will fail for some testcases like $(v,w) = (20,10), (1000,1000)$ and the W is 1000, If we use this then we will get the total value in knapsack as 20 because the ratio is better for that. So again, step3 is proposed.

Step3: Here we will take the maximum of step2 solution or the maximum valued object, whichever is better, this clears the above example.

We can say that the value in step3 is between $\frac{1}{2}$ of optimal solution to optimal solution,

Proof:

We will take the greedy fraction algorithm in this the bag will be fully filled i.e the last part may be filled partially so this algo has always maximum value that other non-fractional algos.

Value in step3 \geq total value of first k objects in sorted thing by ratios,

Value in step3 \geq value of $k+1$ object in that sorted thing

$2 \times (\text{Value in step3}) \geq \text{total value of first } k+1 \text{ object (from adding above two)}$

As in greedy fraction algo the $k+1$ is partial i.e., total value of first $k+1 >$ value in greedy fraction algo

So, $2 \times (\text{value in step3}) \geq \text{value in greedy fraction algo}$

And greedy fraction algo is better than optimal algo

$2 \times (\text{value in step3}) \geq \text{optimal value};$

i.e., value from this algo is $\geq \frac{1}{2}$ of optimal value and maximum of this algo is optimum value.

So, we can say that this algo is at least 50% as good as optimal.

We can give more performance guarantees by making assumptions in knapsack instances. Let every object has $w_i < 10\%$ of the W , then when we apply above algo we get in step2 that the $k+1$ th object cannot be fitted into bag, as we put constraint i.e., $w_{k+1} < 10\%$ of W , so That means there is $< 10\%$ of W available in the bag this says the bag is filled $> 90\%$ of W , here we achieved 90% of accuracy, If we change the assumption to 1% then the accuracy will be 99%, So we can control the accuracy by controlling the constraint.

This is Heuristic-1.

Video link here :

<https://www.coursera.org/learn/algorithms-npcomplete/lecture/EAWJa/a-greedy-knapsack-heuristic>

Reading Algo for Episode 3 on knapsack Heuristic -2

12 November 2020 23:39

I have watched a Coursera video for this Heuristic 2 algo , It is a approximation algo which has a polynomial complexity as we know knapsack is NP complete problem, and this algo will have to compromise on accuracy or on complexity.

Heuristic -2:

This algo solves the problem using dynamic programming, as we know that for normal values of value and weights of objects, we will use the dp solution to solve them with a time complexity $O(n*W)$, n is no of objects and W is knapsack capacity, as we will iterate through the weight and number of objects in dp so we will do that. And the value dp is the maximum value of the knapsack with that weight

In case of weight has values bigger, then we have to do reverse thing, as above we did the value of dp is value and dp iterates with weight and number of objects, so here we will say the value of dp is minimum weight for a given value and number of objects, this algo has a time complexity of $O(n^2 * V_{max})$, And this will work if only if values of objects are smaller.

If the values and weights are bigger then we have to do an approximated algo, that is we have to divide the values of objects with some m to get values into smaller , at this time after approximating we can use the above algo as here the values got smaller , Here we will take the sorting and all stuff with the approximated values, as here after getting the solution we have to multiply it by m , then the resultant solution is not accurate because we approximated when we are dividing with m so, the accuracy depends on m , and m will depend on the E value E is used to make weight assumptions that $w_i < E*W$, and the time complexity for this algo is approx. $O(n^3 / E)$, So here to get more accuracy we have to decrease the values of m , to decrease the values of m we have to decrease E , If we decrease E the time complexity will be increased, So we have to compromise on accuracy or on complexity in this algorithm.

The video link is here:

<https://www.coursera.org/learn/algorithms-npcomplete/lecture/gXaGS/a-dynamic-programming-heuristic-for-knapsack>

Discussed the story of Episode3

13 November 2020 10:19

We three trinadh, me, sriram have meet, to discuss the story of episode 3 , As we finally taken the knapsack algorithm but not sure about taking heuristic-1 or heuristic-2 , These algo will depend the test cases in those situations. We first thought to use a heuristic-2 algo because it's a new algo for us and we spent lot of time on those algos so we think to use heuristic-2 , We know that this algo needs big values of value and weight of the object. So at first we thought about mining , industry, cargo thing, As they have more weight and cost that fits for doings this algo I.e. the weight is normal weight and the value is the cost of the object. In this way we can implement knapsack algorithm.

But we thought to make a good story , Like it should not be visible as knapsack algo , So for this we suggested a situation that there will be a road which was about broken and there care cars on others side, so now we thought the cars should cross the road, but here all cars cannot come, it may broke the road so here the weight is weight of the car and value is the person's life, like that we though as this is not seen in a way of knapsack thing but we can use knapsack here.

From this we made a story that has a lab and the here will blast the lab with a timed bomb, with time T , and he takes the medicine and want to go out , now he soon realised that there are profs there, so he want help them to go out, and profs donno that the bomb is there, here the time taken to take the prof from his room to the place where he left the medicine is t_i , as there only T time so , he may not take all members from here, So he has to choose to take, This situation does not look like a trivial knapsack thing but we can use the algo as , weight as t_i and the value as the person's life or no of years researched something like that, In this way we can use the knapsack algo for this type of situation, and we can say that the timer is in milli seconds precision or some more, so here the weight has a big value and , the value is some life like that we can consider it also a big numbers , Here we can use the heuristic 2 algo, and other part of the story like computing Knapsack , is also discussed in discussion as trinadh mentioned in his diary.

Discussion of algo for Episode –4

17 November 2020 23:36

We had a teams call for taking the algo for episode-4, We found a pdf that has important and famous algos , We are taking some of them and searching them.

The pdf link is here:

<http://www.koutschan.de/misc/algorithms.php>

This has so many algos.

At first I went for Union find , but it is known and some what easier then previous episode , so I left that and searched another, And I gone for Maximum flow algo But I dint get a situation connecting to our episode , So I went for LLL algo,

LLL algorithm

The Lenstra-Lenstra-Lovasz lattice reduction (LLL) algorithm is an algorithm which, given a lattice basis as input, outputs a basis with short, nearly orthogonal vectors. The LLL algorithm has found numerous applications in cryptanalysis of public-key encryption schemes: knapsack cryptosystems, RSA with particular settings, and so forth.

But I saw this as a cryptography and It was so complex to understand.

Then I went for Viterbi algo:

It is interesting So I went to read it on Wikipedia.

In Wikipedia:

Link:https://en.wikipedia.org/wiki/Viterbi_algorithm

There they have another new thing called Markov information Sources and hidden Markov models(HMM).

And I searched about it they are high level stuff,

This algo is also for cryptography.

But there is explained example in Wikipedia, We tried to understand that but no use. And I searched for the algo in Coursera but there are also used complex stuff. So we leaved this algo, and search for another. I went to listen Coursera videos for max cut and min cut and local search and Trinadh gave me some other Coursera videos to read . They are advanced algos, In between He decided to take string algos.

Code for Episode-3

18 November 2020 11:47

I have written the code for knapsack with big integers of weights and small integers of values, So , This code will give u the complexity of $O(n^2 V_{max})$, In that there is a dp it calculates the minimum weight for the corresponding value and no of items, After calculating that , I am looping through the $dp[n][0,1, n \cdot v_{max}]$ n is the no of vertices , In loop I compared the weights of them with Knapsack capacity W, And we have to take the maximum value of weight which is less than W, After that I am doing reversing to find which objects are taken in Knapsack.

But the code is not complete, I did only for small values and bigger weights , But I have do for Bigger values.

Debugging code for Episode-3

20 November 2020 19:00

I am getting wrong answers for the code , Trinadh and me debugged the code and corrected it for only small values , After This I have done the approximate algorithm , Here the values are bigger so that we can't put them as index too, So what the algo is we divide the values by m , then the values become smaller, and we use that $O(n^2 V_{max})$ algo, After attaining the knapsack value we should multiply it by m Here comes the approximation , when we divide with m , the non-divisible things got approximated, So we should choose m wisely to get good performance , Taking the value of m is written in appendix, That is the reason we didn't divide the weights by m , for more to understand the code is commented.

The GitHub link is here: <https://github.com/sreeharsha2002/Episode3-code>

Reading and coding KMP String algorithm

22 November 2020 09:52

We had a discussion of taking string algos, We listened to the Coursera videos of string algorithm, In that they said about the significance of string algorithm, This is very useful in genome sequencing.

That Coursera course link is here:

<https://www.coursera.org/learn/algorithms-on-strings/home/welcome>

In That I that I saw that the implementation with suffix tree is efficient, So I started working on the code for suffix tree. But the Implementation is very big, And We saw that KMP algorithm is more efficient, In terms of code, and the algo is very tricky, They said KMP algo with prefix function. So I wrote the code for KMP Algo, And A test case generator for the code the string length is 10^8 , In that big strings these algos will do efficiently with complexity $O(\text{text} + \text{pattern})$, In that I testcase generator They are only six Jecyll strings and other are random alphabets, And use rand() function to place in different places. And after that the test will be given to KMP code, it gives the starting index of strings, And we have to calculate the password according to the story, So I have coded that part too and it returns directly password. The codes are in GitHub

The GitHub link is here:

<https://github.com/sreeharsha2002/Episode4-code>

I will attach the readme to how to run the codes.



README

Episode4-code

KMP Code

Generation of a test case

1. ./tkmp > testkmp1.txt
2. ./tkmp > testkmp2.txt
3. ./tkmp > testkmp3.txt

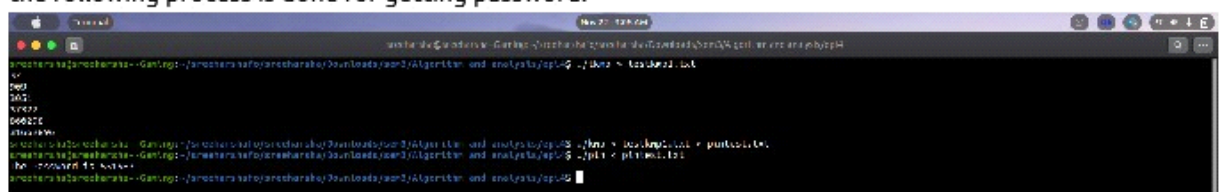
Running the testcase to find matches of word jecyll

- ./kmp < testkmp3.txt > pintest.txt
- the output will redirect to pintest.txt printed the output in stderr too. to know the values,

Finding the password

- ./pin < pintest.txt

the following process is done for getting password.



```
Terminal
sreeharsha@veera:~/Downloads/word3/Algorithm and analysis$ ./tkmp > testkmp3.txt
ve
pqr
SOS
00020
00020
00020
sreeharsha@veera:~/Downloads/word3/Algorithm and analysis$ ./kmp < testkmp3.txt > pintest.txt
sreeharsha@veera:~/Downloads/word3/Algorithm and analysis$ ./pin < pintest.txt
the password is KMP
```

[illegible]