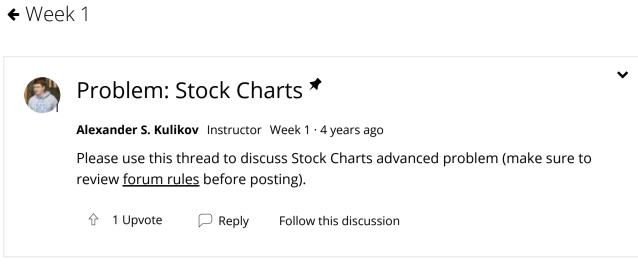
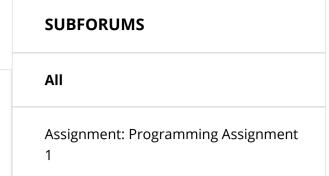
Week 1

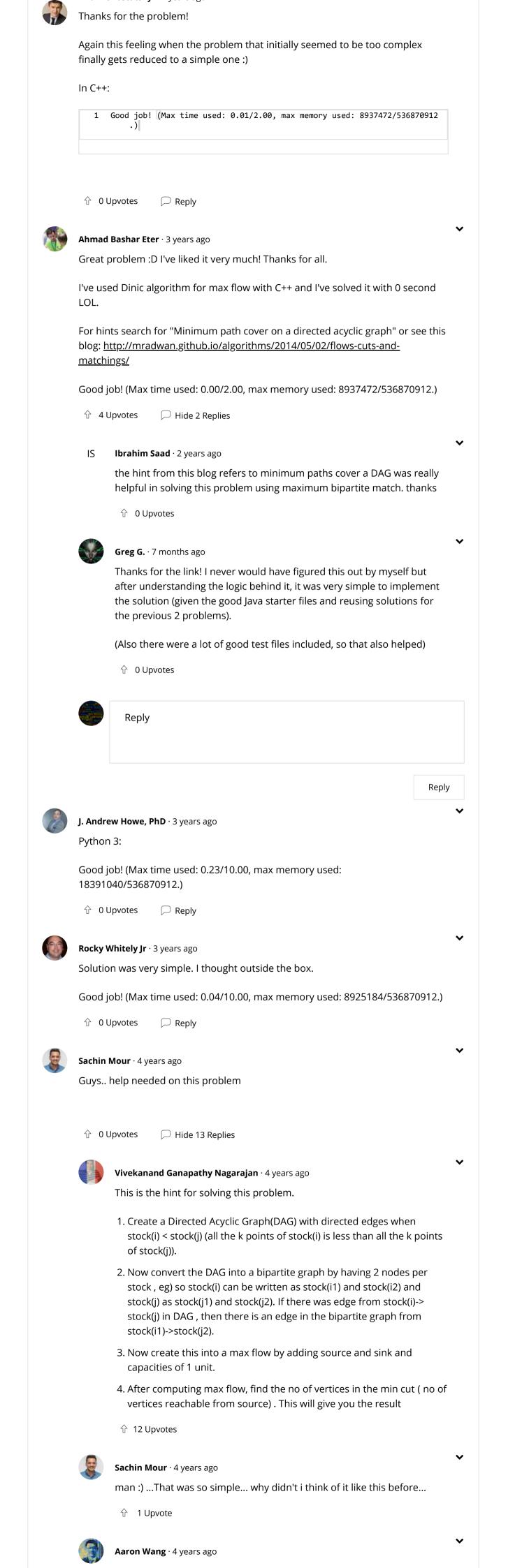


	Earliest Top Most Recent	
DP	dip patel · 3 months ago	~
	I solved a similar problem a few months back. It uses a similar idea.	
	https://www.codechef.com/problems/CHEFDAG	
	û 0 Upvotes	
Н	HS · 4 months ago	~
	I need help for this - I am not getting the answer when I tried Vivekanund's method.	
	♀ 0 Upvotes	
3	Anton Berezin · 10 months ago	~
	Ok, I found a simpler solution without bipartite graph matching:	
	1. Create a undirected graph where there is a node for each line and there is an edge between two nodes if the respective lines intersect	
	2. Color the graph	
	To color the graph use the following algorithm:	
	While there are nodes to color:	
	1. assign empty set to neighbor_set	
	2. allocate a new color	
	3. sort uncolored nodes by their number of their uncolored neighbors in a	
	reverse order (largest count first) and for each node that is not in a neighbor_set do:	
	- add all visited node's neighbors to the neighbor_set	
	- color the visited node with the allocated color	
	♀ 2 Upvotes	
ВС	↑ 2 Upvotes	~
ВС		~
ВС	Balint Cristian · 2 years ago	~
ВС	Balint Cristian · 2 years ago Maybe this will help the someone.	•
ВС	Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following.	•
ВС	Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from	•
ВС	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks 	•
BC	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: 	•
BC	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: Lets say you have n stocks. 	•
BC	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: Lets say you have n stocks. You can start with a list of sets of graph such that each set contains one graph: 	•
BC	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: Lets say you have n stocks. You can start with a list of sets of graph such that each set contains one graph: 	•
BC	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: Lets say you have n stocks. You can start with a list of sets of graph such that each set contains one graph: 1 [g1], [g2],, [gn] Your job is to join these sets such that no graphs in the same set are overlaid. 	•
BC	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: Lets say you have n stocks. You can start with a list of sets of graph such that each set contains one graph: 1 [g1], [g2],, [gn] 	
BC	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: Lets say you have n stocks. You can start with a list of sets of graph such that each set contains one graph: 1 [g1], [g2],, [gn] Your job is to join these sets such that no graphs in the same set are overlaid. Now every unit of flow corresponds to a join of two of these sets. 	•
	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: Lets say you have n stocks. You can start with a list of sets of graph such that each set contains one graph: 1 [g1], [g2],, [gn] Your job is to join these sets such that no graphs in the same set are overlaid. Now every unit of flow corresponds to a join of two of these sets. 	•
	 Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: Lets say you have n stocks. You can start with a list of sets of graph such that each set contains one graph: 1 [g1], [g2],, [gn] Your job is to join these sets such that no graphs in the same set are overlaid. Now every unit of flow corresponds to a join of two of these sets. 	•
	Balint Cristian · 2 years ago Maybe this will help the someone. To solve the problem you have to do the following. 1. Build the bipartite graph as Vivekanand mentioned before 2. Instead of step 4. in Vivekanand's post just subtract the max flow value from the number of stocks Explanation: Lets say you have n stocks. You can start with a list of sets of graph such that each set contains one graph: 1 [g1], [g2],, [gn] Your job is to join these sets such that no graphs in the same set are overlaid. Now every unit of flow corresponds to a join of two of these sets.	•

Andrii Shostatskvi · 2 years ago









What a neat solution! Thanks, Vivekanand ↑ 0 Upvotes ~ **Dat Tran Thanh** · 4 years ago I dont seem to get correct result using this way. I set up the bipartite graph as described above: connect all node (i1) to source, all (i2) to sink (i=1...stock_number), compute max flow. But I'm not sure what I should return here. ↑ 0 Upvotes ~ **Shouman Das** · 4 years ago Cool. Thanks for the idea, vivek. ⊕ 0 Upvotes ~ **Kfir Berger** · 4 years ago Thanks for the tip, Vivek! Made the solution pretty simple. And a lot of copy paste from the previous problem (Assigning Airline Crews to Flights) ↑ 1 Upvote ~ **Rishikesh** · 4 years ago · Edited Does this really work? I would expect the answer to be N - (# of matchings), where N is the number of stocks. Added: Unless Vivek means reachable in the residual graph. û Upvotes ~ **Hyun Jung** \cdot 4 years ago \cdot Edited @Rishikesh Strangely, it works as you stated ! (N - #of_matchings) I cannot prove mathematicaly, but if you play with toy examples (drawing many charts with different overlaping criteria), you would see this working! û Upvotes ~ **Soumava Bera** · 3 years ago Why do we need two nodes for each stock? û Upvotes **~** Vaibhav Bhandari · 3 years ago · Edited There is a very nice explanation of why we need to split the stock nodes into 2 and why the min number of charts = Number of Stocks - Max flow: http://mradwan.github.io/algorithms/2014/05/02/flows-cuts-andmatchings/ Implemented in Java using Ford-Fulkerson DFS algorithm: Good job! (Max time used: 0.19/3.00, max memory used: 26447872/536870912.) ↑ 0 Upvotes **Rachel Bowyer** · 3 years ago · Edited Wow, this was such a hard problem I came to the forums for some hints. Even finding a counterexample showing the greedy algorithm fails needed some thought. In my view probably the hardest problem yet in the Data Structures and Algorithms specialism. Having said that, the divide and conquer algorithm for finding the closest pair of points in n log n time runs it a close second. And what a beautiful solution! So my understanding is the problem becomes finding the minimum vertex disjoint path cover for a DAG. This problem is then transformed into a matching problem of a bipartite graph. This problem is then transformed into a maximum flow problem. The maximum flow problem is solved using Edmunds Karp algorithm, which itself uses a BFS. This leads to a polynomial time solution to the problem. BTW you don't need to find a minimum cut. The maximum flow comes directly out of Edmunds Karp. And as has been noted, the number of disjoint paths = number of charts is N - max flow. Out of curiosity, do participants in the Google Code Jam have an encyclopaedic knowledge of algorithms? Similar to how participants in the world scrabble championships memorise thousands of obscure words. Or do participants just rely on intuition alone? ↑ 0 Upvotes **Yan Zhou** · 2 years ago Thank you for good explanation, after build the matrix by compare points, I just copied code from airline crews, and it was solved. û Upvotes ~ James D Lin · a year ago Good tips. Merely needed to modify the min_charts() and flight crew code a little bit to get it working. Python3 code:

