Discussion Forums

Week 1

← Week 1

Problem: Stock Charts

Alexander S. Kulikov Instructor Week 1 · 4 years ago

Please use this thread to discuss Stock Charts advanced problem (make sure to review forum rules before posting).

1 Upvote Reply Follow this discussion

SUBFORUMS

All

Assignment: Programming Assignment 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

0 Upvotes Reply

H **HS** · 4 months ago

I need help for this - I am not getting the answer when I tried Vivekanund's method.

0 Upvotes Reply

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 Reply

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:

```
[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.

2 Upvotes Reply

AR **Anton Rapetov** · 2 years ago

C++:

```
Good job! (Max time used: 0.00/2.00, max memory used:
8933376/536870912.)
```

Though another submission of the same code gave me `0.08`, fluctuations. A neat problem by the way.

0 Upvotes Reply

Andrii Shostatskyi · 2 years ago

Thanks for the problem!

Again this feeling when the problem that initially seemed to be too complex finally gets reduced to a simple one :)

In C++:

```
1   Good job! (Max time used: 0.01/2.00, max memory used: 8937472/536870912
        .)
```

0 Upvotes   Reply

Ahmad Bashar Eter · 3 years ago

Great problem :D I've liked it very much! Thanks for all.

I've used Dinic algorithm for max flow with C++ and I've solved it with 0 second LOL.

For hints search for "Minimum path cover on a directed acyclic graph" or see this blog: http://mradwan.github.io/algorithms/2014/05/02/flows-cuts-and-matchings/

Good job! (Max time used: 0.00/2.00, max memory used: 8937472/536870912.)

4 Upvotes   Hide 2 Replies

IS **Ibrahim Saad** · 2 years ago

the hint from this blog refers to minimum paths cover a DAG was really helpful in solving this problem using maximum bipartite match. thanks

0 Upvotes

Greg G. · 7 months ago

Thanks for the link! I never would have figured this out by myself but after understanding the logic behind it, it was very simple to implement the solution (given the good Java starter files and reusing solutions for the previous 2 problems).

(Also there were a lot of good test files included, so that also helped)

0 Upvotes

Reply

Reply

J. Andrew Howe, PhD · 3 years ago

Python 3:

Good job! (Max time used: 0.23/10.00, max memory used: 18391040/536870912.)

0 Upvotes   Reply

Rocky Whitely Jr · 3 years ago

Solution was very simple. I thought outside the box.

Good job! (Max time used: 0.04/10.00, max memory used: 8925184/536870912.)

0 Upvotes   Reply

Sachin Mour · 4 years ago

Guys.. help needed on this problem

0 Upvotes   Hide 13 Replies

Vivekanand Ganapathy Nagarajan · 4 years ago

This is the hint for solving this problem.

1. Create a Directed Acyclic Graph(DAG) with directed edges when stock(i) < stock(j) (all the k points of stock(i) is less than all the k points of stock(j)).
2. Now convert the DAG into a bipartite graph by having 2 nodes per stock , eg) so stock(i) can be written as stock(i1) and stock(i2) and stock(j) as stock(j1) and stock(j2). If there was edge from stock(i)->stock(j) in DAG , then there is an edge in the bipartite graph from stock(i1)->stock(j2).
3. Now create this into a max flow by adding source and sink and capacities of 1 unit.
4. After computing max flow, find the no of vertices in the min cut (no of vertices reachable from source) . This will give you the result

12 Upvotes

Sachin Mour · 4 years ago

man :) ...That was so simple... why didn't i think of it like this before...

1 Upvote

Aaron Wang · 4 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

SD

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

R

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.

0 Upvotes

Hyun Jun Jung · 4 years ago · 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 !

0 Upvotes

SB

Soumava Bera · 3 years ago

Why do we need two nodes for each stock?

0 Upvotes

VB

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-and-matchings/

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

YZ

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.

0 Upvotes

JL

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:

Good job! (Max time used: 0.15/10.00, max memory used: 8921088/536870912.)

0 Upvotes

Reply

Reply

Vivekanand Ganapathy Nagarajan · 4 years ago

Any insights on how to solve this problem . I have having a tough time figuring out how to translate this problem into a max bipartite matching problem.

0 Upvotes Hide 2 Replies

Julian Chukwu · 4 years ago

Hi vivekanad,

I need help understanding the intuition behind your approach. What I don't quite get is how "n" stocks, each with "k" points can be compressed in to a bipartite graph of 2 stocks i & j with at most k edges (for all stock(i) points less than stock(j) points) running from stock i to stock j. Were all points in stock "i" are attached to the source and all points in stock "j" are attached to sink. This is so far how am able to interpret your solution. But i am yet to understand what happens to other n-2 stocks if n> 2.

Thanks.

0 Upvotes

daniel · 2 years ago

'ur solution' is nonsense!

0 Upvotes

Reply

Reply

Sampriti Panda · 4 years ago

Is it possible to solve this using Max flow-Min Cut?

0 Upvotes Hide 2 Replies

CT **chenzhe tian** · 4 years ago

It is solved by Max flow through max-matching in a bipartite graph.

0 Upvotes

Sampriti Panda · 4 years ago

Yes, I used that solution. But is there any solution using min cut in particular (Not using D***).

0 Upvotes

Reply

Reply

CT **chenzhe tian** · 4 years ago

Just a small tip:

Think about the condition that for any pair of stocks if they do not cross each other, which one is higher than the other one, instead of the condition that whether two stocks cross each other or not.

0 Upvotes Hide 2 Replies

CT **chenzhe tian** · 4 years ago

And also test case #3 is the counterexample of an incorrect greedy algorithm.

0 Upvotes

daniel · 2 years ago

this is bullshit

0 Upvotes

Reply

Reply

Sam Handelman · 4 years ago

Just wanted to say that this problem is awesome! It really got me thinking about and researching lots of interesting aspects of graph theory. It also helped me gain some insight into algorithmically solving this game that I love to play on my phone called Flow (I think the name is no coincidence!) which I've been trying to do for a little while now. If anyone is interested, it's a very fun puzzle game and is really all about connecting k-vertex disjoint paths when you thinks about it: https://blog.bigduckgames.com/category/flow/

0 Upvotes Reply

Jan Herman · 4 years ago

Wow, this was a hard one.

I have to admit, that I "cheated" a little by googling how to solve (unnamed problem that I reduce the original task to) by finding maximum bipartite matching.

0 Upvotes Reply

1 2

Reply

Reply