! This class has been made inactive. No posts will be allowed until an instructor reactivates the class.

note @1034 692 views

Project Postmortem

The project is officially over! We hope you enjoyed it.

We are shooting to get project grades out to you all before the deadline to change your grading option this Friday. Until then, we won't be able to productively answer questions about grading, so please hold tight.

In the meantime, feel free to share your approaches with your classmates below. Staff may share their ideas here as well.

#pin

project

~ An instructor (Rishi Veerapaneni) thinks this is a good note ~

Updated 2 months ago by David Vendrow

followup discussions *for lingering questions and comments* 14 endorsed followup comments

● Resolved ○ Unresolved

Anonymous Atom 2 months ago What were your guys' approaches to doing well on medium and large graphs? We had some seriously bad outputs on large graphs.

helpful! 1

Anonymous Beaker 2 months ago I'm not that smart, but some of the really large, dense graphs often just had 2-3 vertices/edges, so I just brute-forced using a n^3 time algorithm starting from the highest degree vertex. Most of the time, it got near or exactly the optimal solution but you only need a couple vertices to be connected to every vertex. Any large graph with >1500 edges seemed to behave like this and any medium graph with >450 edges.

~ An instructor (Ajay Raj) thinks this is a good comment ~

helpful! 3

Anonymous Beaker 2 months ago 2-3 vertices/edges as in as an optimal(ish) output for the network

helpful! 0

● Resolved ○ Unresolved

Anonymous Comp 2 months ago
Does my team name need to appear on the leaderboard to get graded? It was there yesterday night when we submitted, but when I checked just now it was gone.

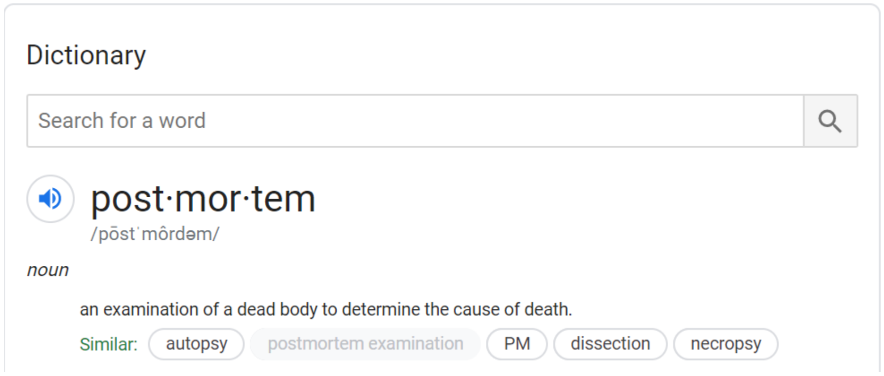
helpful! 0

David Vendrow 2 months ago Are you sure it disappeared? This should not happen. In any case as long as your outputs are submitted to Gradescope you are fine.

helpful! 1

[✓ Resolved] Unresolved

Frank Liu 2 months ago


Dictionary

Search for a word

post·mor·tem
/pōstˈmôrdəm/

noun

an examination of a dead body to determine the cause of death.

Similar: autopsy postmortem examination PM dissection necropsy

Nice

~ An instructor (Varun Jhunjhunwalla) thinks this is a good comment ~

undo helpful | 37

[✓ Resolved] Unresolved

Charlie Snell 2 months ago

all_my_homies_hate_dino_nuggets used a genetic algorithm, basically we just get a large set of base cases (Dijkstra's from each vertex, prim's from each vertex, a couple other custom approches). We then run a couple greedy algorithms to remove some of the removable vertices from the tree. And then we create a probability distribution from their scores using the Softmax/Boltzman function with some temperature, and then sample 2 graphs randomly from that distribution and then get the intersection of those graphs. We then create a sort of "meta graph" with this intersection graph, basically treating the connected components as a single node, and then run the same set of algorithms on this meta graph to create a new "generation" of approximate graphs, we then repeat this process on the new generation of graph and do this again and again. The idea is to simulate evolution, pick the best graphs, combine them, and then in theory as generations progress you will find better and better graphs. We also induce some random "mutations" to help us explore the space a bit better, we remove some edges at random and then randomly rebuild the graph with like 2% chance. It ended up working pretty well overall, I spent a lot of time tuning the parameters of the algorithm to get it just right though.

Here is our GitHub if you want to check it out. https://github.com/Sea-Snell/170project the code is super undocumented and a bit of a mess admittedly

~ An instructor (Emaan Hariri) thinks this is a good comment ~

undo helpful | 27

Charlie Snell 2 months ago We also scrape the leaderboard, so that we can just run longer on the inputs we are preforming worst on, so the code might not work if the leaderboard database is down when you run it, there are instructions in the readme on how to get around this

helpful! | 3

Anonymous Helix 2 months ago This is amazing! Our group went through the first couple steps (getting multiple starting spanning trees, greedily finding the best arrangement of edges, removing leaves etc.), but definitely didn't think of your remaining procedure. Congrats!

helpful! | 0

Anonymous Helix 2 months ago How long on average did your small, medium, and large inputs take?

helpful! | 0

Charlie Snell 2 months ago I have it programmed to stop early if it hits the top score on the leaderboard, so for small inputs it would do that frequently in the first generation, so it just took a minute or so. For medium id say it took 3 generation on abverge so maybe 5 minutes. And then the

large ones usually ran for 10 generations before stopping so maybe 10-15 minutes. I just ran everything on my laptop though
helpful! | 2

Anonymous Gear 2 months ago this is massive brainz
helpful! | 1

Anonymous Mouse 2 months ago You're crazy bruh. Mad respect.
helpful! | 0

Annamira O'Toole 2 months ago I also implemented the same metagraph idea you described to run approximations over graphs simplified by their connected components as a rough divide and conquer approach, where different connected components can be solved with different subroutines like MST or set cover style algs. But... I never finished this and my team focused on implementing well several straightforward approaches. I had some trouble with networkx while implementing my mini "metagraph library". Now that I've thought about this for so long I'd love to see how you implemented it because I'm sure there are some design decisions that I could have improved on, thanks for the link Charlie
helpful! | 0

Annamira O'Toole 2 months ago did you use multiple levels of metagraph or only one level?
helpful! | 0

Charlie Snell 2 months ago Technically it kinda does 2 levels, cause there's this feature where we break the graph into highly connected components, cluster it into those components, solve each component then solve the meta graph of all the components together, and this can be preformed on the meta graph formed by the intersection. But I don't think going any deeper with the meta graph would help cause like you lose information every time you reduce a set of nodes into a single vertex.
helpful! | 0

Max Litster 2 months ago This is a pro gamer move
helpful! | 0

[✓ Resolved] Unresolved

Jaren Mendelsohn 2 months ago
We (jaarn) used a fairly basic simulated annealing method: we started the algorithm off with some initial solution (like try to make an MST and stop when we have a dominating set), and then iteratively try to improve it by randomly mutating the graph (removing nodes, adding nodes, switching edges around) to get another valid solution. Mutations with lower cost are accepted automatically, but sometimes mutations with higher costs are accepted too in order to allow the algorithm to escape local minimas. We didn't really use anything complicated apart from tuning a few hyperparams (number of iterations, energy scaling, etc.).

Like dino nuggets, we also optimized directly against the leaderboard: we scraped the leaderboard data used a priority queue by rank where higher-rank problems (things that we were relatively worse at) were run first, and we essentially just ran the program on one of our computers for a few days to whittle down the ranks more.

~ An instructor (Varun Jhunjhunwalla) thinks this is a good comment ~

undo helpful | 16

Aaron Opell 2 months ago Here's our GitHub repo if you want to check it out, but it's not particularly well-organized or well-documented: https://github.com/aopell/cs170-project
helpful! | 1

Anonymous Scale 2 months ago While introducing random mutations in the simulated annealing procedure, what steps did you take to ensure that the resulting network at every iteration was still valid (i.e. both a tree and a dominating set)?
helpful! | 0

Jaren Mendelsohn 2 months ago There were three mutations we used, each with a certain probability of happening:
- Add a vertex if possible
- Remove a leaf vertex if its removal would still leave a dominating set
- Disconnect an edge randomly and reconnect the connected components on either side of that edge with a random possible edge between those components

I'm sure there are other ways to implement mutations which would've been faster or led to better behavior, but these are what we ended up using.
helpful! | 1

[✓ Resolved] Unresolved

Haoyuan Liu 2 months ago
HopesAndDreamsGon generates a partial SPT and MST from each vertex and run a series of greedy local optimization including edge-swapping / pruning / expanding. Different heuristics (such as dividing edge cost by degree of vertices when pushing vertices onto the heap) are selectively used in tree generation for graphs where performance is poor (yes, I also scraped the leaderboard). Code and more detailed descriptions can be found in the github repo.

~ An instructor (Varun Jhunjhunwalla) thinks this is a good comment ~
undo helpful | 4

[✓ Resolved] Unresolved

Chase Norman 2 months ago

I'm MeanPairwiseSocialDistance. We used branch and bound which prioritizes branches according to a heuristic cost. HMU for more info or just to chat c_#9141 @cut.to.the_chase :)

our code can be found here: https://github.com/chasenorman/HorizonWireless

Also great job everyone! If you're Anthony_Fauci, jaarn, all_my_homies, etc. contact me! It was great fun competing with y'all.

~ An instructor (Ajay Raj) thinks this is a good comment ~
undo helpful | 10

Charlie Snell 2 months ago good shit Chase ! Every time I submitted some improved outputs you would match them a couple hours later, that was hella fun to watch
helpful! | 1

Jaren Mendelsohn 2 months ago yeah great job! you just flew up the leaderboard towards the end there lol
helpful! | 0

徐一末 2 months ago

Chase aced this again

helpful! | 0

Chase Norman 2 months ago Were you in summer 61A?

helpful! | 0

徐一末 2 months ago

yeah

helpful! | 0

Chase Norman 2 months ago yo that's awesome! Contact me!!

helpful! | 0

徐一末 2 months ago

gotta follow your github lol

helpful! | 0

[✓ Resolved] Unresolved

Teddy Tran 2 months ago
Whoever was SPY210P_DIAMOND_HANDS, I expect you to be most upvoted in r/wallstreetbets

~ An instructor (Ajay Raj) thinks this is a good comment ~

helpful! | 9

Anthony Bajoua 2 months ago they're the guh of 170!

helpful! | 1

[✓ Resolved] Unresolved

Anonymous Poet 2 months ago
@964_f28 So... which one was staff?

Also, I'm guessing that the last batch of tests in large was staff created? What were the chracteristics of those graphs, as our algorithm seemed to perform rather poorly on those compared to student-made tests?

helpful! | 3

Anonymous Calc 2 months ago I'm not staff but I did see a lot of those graphs had a low density (relatively low number of edges), which I think means solution trees had more nodes in it and thus were harder to find.

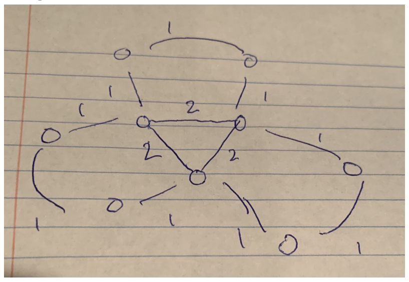
helpful! | 0

Ajay Raj 2 months ago I added three types of inputs (they were the 100 extra large inputs).

1) Start with a random tree with weights chosen from a normal distribution, and then with some probability add a path of two edges between two nodes with weight from a normal distribution with large mean. This ensures that the tree we started with was still a valid network.

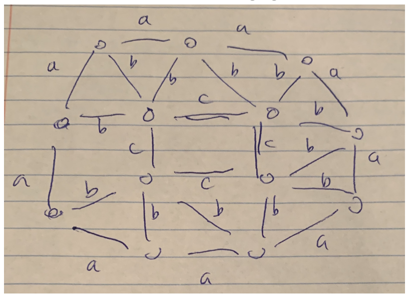
2) "Radioactive" graphs (because they look a little like the radioactive symbol): Graphs where the optimal network had no edges from the MST of the graph. Someone had asked on Piazza during Phase 1 if it were the case that the optimal network always has to have some of the edges in an MST.

It turns out that the answer to this is no (credit to **@Rishi Veerapaneni**). Here is an example of such a graph, where the MST will contain only the 1 edges, but the optimal network only needs 2 of the 2 edges.



So, you can start with any tree, and then for every edge, connect a vertex to each endpoint and connect the endpoints, all with weight slightly lower than the weights of the tree, and the MST will not intersect with the optimal network.

3) Two-tiers: a greedy algorithm (that works on many graphs) is as follows: select the vertex of highest degree, and then, of its neighbors, choose the vertex that covers the most uncovered neighbors so far. Consider the following graph:



Taking the vertices of highest degree forces the algorithm to use the "c" edges, while using the "a" edges may be more optimal depending on the weight. Also, depending on the weight of the "b" edges, the optimal network add leaves to the tree (same as that one problem from the homework). So, all of the graphs of this type were created by creating two separate trees, one smaller and one larger, and connecting each of the nodes in the smaller tree to many of the nodes in the bigger tree.

~ An instructor (Rishi Veerapaneni) thinks this is a good comment ~

helpful! | 4

Ajay Raj 2 months ago Suffice to say, I also had a lot of fun with the project :)

~ An instructor (Rishi Veerapaneni) thinks this is a good comment ~

helpful! | 2

Dee Guo 2 months ago Nice fidget spinner :P

~ An instructor (Noah Kingdon) thinks this is a good comment ~

helpful! | 13

[✓ Resolved] Unresolved

Tom Shen 2 months ago

Team Salieri uses Artificial Bee Colony Algorithm. The algorithm has a number of observers running in parallel. For each observer, it first randomizes the graph by sampling the weight of edges in a normal distribution, and then randomly find MST or SPT with a random starting vertex as a starting point. Then, it randomly prunes leaves. The program tracks the best solution ever observed. If you are interested, read the code and algorithm here: https://github.com/tsunrise/cs170-proj

~ An instructor (Ajay Raj) thinks this is a good comment ~

undo helpful | 10

Anonymous Atom 2 2 months ago Super interesting solution. Nice.