CS4102 - ALGORITHMS - SPRING 2022

DUE TUESDAY, MAY 3, 2022 AT 11:30 PM VIA GRADESCOPE

Dadv(wyz5rge)

Collaboration Policy: You are encouraged to collaborate with up to 3 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite by naming the book etc. or listing a website's URL. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators: list collaborators's computing IDs

Sources: Cormen, et al, Introduction to Algorithms. *(add others here)*

PROBLEM 1 *IKEA Grill*

IKEA is growing in popularity across the US, however their stores are only found in a handful of larger metropolitan areas. While their main product is furniture, they have become known for their signature meatballs. To increase profits and make their delicious food more accessible, they have decided to open local IKEA Grill restaurants in towns across the country. Restaurant storefronts are expensive to rent and maintain, so they are happy with customers needing to drive at most to the next town over to get their IKEA meatball fix. Specifically, their goal is that every town in America either has an IKEA Grill, or its neighboring town has an IKEA Grill.

Given a graph representing the towns (as vertices) and roads between them (edges connecting neighboring towns), the *IKEA Grill* problem is to decide whether k IKEA Grill locations can be placed in order to ensure that each town or its neighbor has an IKEA Grill location. Show that *IKEA Grill* is NP-Complete.

Note: You are not being asked to explicitly solve the *IKEA Grill* problem; you are only required to show that it is NP-Complete.

Solution: We begin by showing that the *IKEA Grill* problem is NP by establishing that it can be verified in polynomial time - given a graph and a solution, it can be determined in polynomial time that each vertex either has an IKEA Grill or is adjacent to one that does. This verification has a worst case running time of $O(V^2)$.

We then proceed to show that the problem is NP-Hard. We do this by showing that *K-Vertex Cover*, a known NP-Hard problem, reduces to *IKEA Grill*. *K-Vertex Cover* can be used to solve *IKEA Grill* because it determines whether a set of K vertices can be selected such that each vertex is an endpoint of every edge in the graph, which is synonymous to the constraint that each vertex/town has an IKEA Grill or is next to one with an IKEA Grill. This is because since every edge has an endpoint in the set of K vertices, that means that if a vertex cover exists of size K, every vertex either is in that set, or is connected by an edge to another vertex in that set. If there is a k-vertex cover for the graph, there is a solution to the *IKEA Grill* problem, and vice versa.

By showing the *IKEA Grill* problem is NP and NP-Hard, we can conclude that it is NP-Complete.

PROBLEM 2 *Backpacking, Revisited*

After your first successful backpacking adventure (Unit C's Advanced, Problem 1), you have decided to return to Shenandoah National Park. Similar to before, you and your friend have completed your packing list, and you need to bring n items in total, with the weights of the items given by $W = (w_1, \ldots, w_n)$. Your goal this time is to divide the items between the two of you such that the difference in weight is as small as possible. There is no longer a restriction on the total number of items that each of you should carry. Here, we will define a decisional version of this BACKPACKING problem:

BACKPACKING: Given a sequence of non-negative weights $W = (w_1, \ldots, w_n)$ and a target weight difference t, can you divide the items among you and your friend such that the weight difference between backpacks is at most t?

1. Show that the BACKPACKING problem defined above is NP-complete (namely, you should show that BACKPACKING $\in$ NP and that BACKPACKING is NP-hard). For this problem, you may use the fact that the SUBSETSUM problem is NP-complete:

   SUBSETSUM: Given a sequence of non-negative integers $a_1, \ldots, a_n$ and a target value t, does there exist a subset $S \subseteq \{1, \ldots, n\}$ such that $\sum_{i \in S} a_i = t$?

   Solution: The BACKPACKING problem is NP, as a solution to the problem can be verified in polynomial time by simply summing up the weights in each backpack, subtracting, and seeing if the different is less than or equal to t. This verification would have a runtime of $O(n)$.

   We can show this problem is NP-Hard by reducing SUBSETSUM, which is already known to be NP-complete, to it. Consider an instance of the SUBSETSUM where given input $a_1, \ldots, a_n, t$, there exists a set $S \in a_1, \ldots, a_n$ such that $\sum_{i \in S} a_i = t$. Also consider a quantity A equal the the sum of all a in the input. This means that if we were to assign items to the two backpacks, we'd have backpack S weigh t and our second backpack T would weigh $A - t$. If we were to add two new items with weights $A + t$, $2A - t$, then they must be disjoint, as if they were in the same backpack, it would weight $3A$, which violates the difference condition of the problem. Assigning $2A - t$ to S and $A + t$ to T, we are left with the weights $|S| = t + (2A - t) = 2A$, and $|T| = A - t + (A + t) = 2A$. This means that there exists a solution to BACKPACKING if there exists one to SUBSETSUM, as the reduction requires the condition that some weights in the input to BACKPACKING add up to t, which is the same as running SUBSETSUM on a subset of the input. Because of this dependency, it can be concluded that BACKPACKING is NP-Hard and thus also NP-Complete.

2. Your solution to Unit C Advanced's Problem 1 can be adapted to solve this version of the BACKPACKING problem in time that is *polynomial* in n and M, where $M = \max(w_1, \ldots, w_n)$ is the maximum weight of all of the items. Why did this not prove P = NP? (*Conversely, if you did prove that P = NP, there's a nice check waiting for you at the Clay Mathematics Institute.*)

   Solution: As previously discussed in Unit C, because M is an input value and not an input size, the time complexity depends on the space (bits) needed to represent M, meaning the problem has a pseudo-polynomial running time. This distinction means our algorithm is not actually polynomial, thus P $\neq$ NP

PROBLEM 3 *Gradescope Submission*

Submit a version of this `.tex` file to Gradescope with your solutions added, along with the compiled PDF. You should only submit your `.pdf` and `.tex` files.