CS4102 - ALGORITHMS - SPRING 2022

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

D_{ADV}(HZ9XS)

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: wx8mcm, hl8zd

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:
1. Show that IKEA Grill problem belongs to NP
To check whether every town in America either has an IKEA Grill, or its neighboring town has an IKEA Grill, we need to run over every vertices(towns), and check their neighbours(every edges) to see if they have IKEA Grill or their neighbours do. To do this, we need $O(|V|^2)$, which is polynomial time. So, the problem is NP.
2. Show that the problem is NP-hard
To show that IKEA Grill problem is NP-hard, we will have to reduce independent set problem to IKEA Grill problem. To do so, we will just call IKEA Grill problem using the input of independent set problem. If input is k and IKEA Grill problem returns true for that input, we know the corresponding independent set problem will return false because there will not be a set containing two vertices and both without IKEA Grill. Similarly, if IKEA Grill problem returns false for that input, we know that the corresponding independent set problem will return true because there must be a set of two vertices that both do not have an IKEA Grill. In this way, we connect two problems together and reduce the independent set problem to the IKEA Grill problem in polynomial time. Because independent set problem is NP-complete as proved in the lecture, we can say that IKEA Grill problem is NP-hard.
3. Because we have proved the above two statements, we can conclude that IKEA Grill problem 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 $W = (w_1, \dots, w_n)$. 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 $W = (w_1, \dots, w_n)$ 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 $a_1, \dots, a_n$ and a target value t, does there exist a subset $S \subseteq \{1, \dots, n\}$ such that $\sum_{i \in S} a_i = t$?

   Solution:
   1. Show that Backpacking problem belongs to NP
   For backtracking problem, we have designed an algorithm that can solve this problem in polynomial time in unit C, so it must belong to NP.
   2. Show that it is NP-hard
   To show that Backpacking problem is NP-hard, we will reduce Subsetsum problem to backtracking. Let $(a_1, ..., a_n, t)$ be a Subsetsum instance. First, we will need to find the value of the total weight and $2 * t$, mark them, and put them into the instance list. Call backtracking on the list and let $t = 0$ for backtracking(meaning items must be divided evenly). If t cannot be 0, we return impossible for subsetsum problem. If we can find such division, then we trace to the divison with the sum of total weight in it. Then, we eliminate the total weight and return the rest as the result for the subsetsum problem. Because Subsetsum problem is NP-complete, we can say that the Backtracking is NP-hard. This reduction is polynomial time because marking the value, adding and finding value in the list, and removing them takes $O(polynomial)$ time.
   3. Because we have proved the first and second statement, we can say that the backtracking problem is 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 $M = \max(w_1, \dots, w_n)$ 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:
   Because the algorithm we used in Unit C has one limit: the difference in size of two sublists should be at most one. This constraint makes my algorithm for Unit C advance problem not working here. For example, if we have a list of objects with weights: $[1, 2, 2, 2, 2, 2, 10]$. In unitC, our algo will divide two sublists as $[2, 2, 2, 2]$ and $[1, 2, 10]$. However, here we will divide the list into: $[2, 2, 2, 2, 2]$ and $[1, 10]$. So our algo in unit C is not correct here, which can not be used to prove $P = NP$.

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.