CS4102 - ALGORITHMS - SPRING 2022

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

Dadv(ylw4sj)

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: To show that IKEA Grill is NP-Complete, we must prove that it is NP and NP-Hard. This is a decision problem and assuming we can solve for this, we can then also solve for a search problem.

We know it is NP since we can verify a certificate in polynomial time. This can be done in $O(V^2)$ since we need check that every vertex is an IKEA Grill or is connected to one. In the worst case, there is only 1 IKEA Grill and all other vertices are adjacent to it.

Next, to show that the problem is NP-Hard we will compare it to the vertex cover problem (a known NP-Hard problem) and reduce that problem to ours. Let S be a vertex cover of size k for graph G. Since all edges are covered by S, every node not in S is adjacent to some node in S. Let S' contain the same set of nodes as S. If S' is the set of towns that have an IKEA Grill, then every town not in S' is adjacent to an IKEA Grill. Therefore, G contains a IKEA Grill solution of size k only if there is a vertex cover of size k. Therefore, the IKEA Grill problem is NP-Hard.

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 $W = (w_1, \ldots, w_n)$ 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: We can show that the Backpacking problem is NP-Complete by showing it is NP and NP-Hard.

   We know it is NP because we can verify a certificate in polynomial time. This is done by checking that the difference in total weight of backpack 1 and backpack 2 $\leq t$, which is $O(n)$.

   To prove it is NP-Hard, we first convert the Backpacking problem to a decision problem, which returns true/false depending on if it is possible for a set of items to be divided with a max difference of t. Next, we reduce the subset sum problem to ours.

   The input for subset sum would be the list of weights, W, and the target value, t. Let the total weight of W be T. To convert this input into the backpacking problem, we add two items to W. The first item (i_1) has a weight of $T + t$ and the second item (i_2) has a weight of $2T - t$. Now, $W' = W' + i_1 + i_2$ has a total weight of $4T$. Then we call the backpacking problem using W' and $t = 0$ as input. This results in backpacks of equal weight with each weighing $2T$. In order for the backpacks to be of equal weight, i_1 and i_2 would have to have been sorted into different backpacks, or else the backpacks would have weights of $3T$ and T. Looking at the backpack that contains i_2, if we were to remove i_2 then the weight of that backpack would equal t (since $2T - (2T - t) = t$, which is the target value for the subset sum problem. In addition, since i_2 has been removed, all the remaining weights are from the original list W. This means we have found a subset of W that sums to t, which solves the subset problem.

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: M is not a size but instead is a value, which means this problem is actually pseudo-polynomial and has exponential run time. Therefore, this problem is not in polynomial time. If P = NP were true, that means there is a polynomial solution to every NP problem.

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.