INSTITUTE OF TECHNOLOGY

TCSS 343 - Assignment 5

Version 1.0

May 31, 2018

1 GUIDELINES

Homework should be electronically submitted to the instructor by midnight on the due date. A submission link is provided on the course Canvas Page. The submitted document should be typeset using any common software and submitted as a PDF. We strongly recommend using LaTeX to prepare your solution. You could use any LaTeX tools such as Overleaf, ShareLatex, TexShop etc. Scans of handwritten/hand drawn solutions are acceptable, but you will be docked 1 point per problem if the handwritting is unclear.

Each problem is worth a total of 10 points. Solutions receiving 10 points must be correct (no errors or omissions), clear (stated in a precise and concise way), and have a well organized presentation. Show your work as partial points will be awarded to rough solutions or solutions that make partial progress toward a correct solution.

Remember to cite all sources you use other than the text, course material or your notes.

2 PROBLEMS

2.1 UNDERSTAND

You and your friends are driving to Tijuana for spring break. You discover that you are bringing a lot of gear, luggage, and people and may have to take multiple cars. You have n items you want to bring. Each item weighs between 1 pound and 1000 pounds. Each car can hold at most 1000 pounds. You want to determine how to assign items to cars so that you use the minimum number of cars.

1. (4 points) State the input and output conditions for this problem as precisely as possible.

   Answer:

   CarAmount

   Input: $A[1 \dots n]$, a list of item weighs between 1 and 100

   Output: n, the minimum number of cars need to carry all the item weighs, such that $1000n \geq sum(A[1 \dots n])$.

2. (4 points) If you have n items what is the maximum number of cars you will need? If you have n items what is a lower bound on the minimum number of cars you will need?

   Answer:

   If you have n items,

   The maximum number of cars is $\frac{1000n}{1000} = n$.

   The lower bound of on the minimum number of cars is $\lceil \frac{1}{1000} \sum_{i \in N}^{n} W[i] \rceil$. N stands for Item set and W stands for the corresponding weight set.

3. (6 points) Your friend Alice has a plan to pack the cars. She suggests placing each item in the first car that it will fit in. She calls this the first-fit algorithm. Come up with a counter example to show this algorithm will not produce the least number of cars.

   Answer:

   Items: $\{400, 500, 600, 500\}$.

   If we use the first-fit algorithm, the first two items will be put in the first car; the second item will be put in the second car; and the third item will be put in the third car. However, we can actually put the second and the forth item in the first car; the first and the third item to the second car. Then all we need is two cars instead of three cars.

4. (6 points) Your friend Bob has a plan to pack the cars. He suggests placing each item in the car it fits best in. He thinks an item fits best in a car if it leaves the car with the least amount of free space. He calls this the best-fit algorithm. Come up with a counter example to show this algorithm will not produce the least number of cars.

   Answer:

   Items: $\{600, 700, 100, 300, 300\}$.

If we use the best-fit algorithm, the first items will be put in the first car; the second item will be put in the second car; the third item will be put in the second car; the forth item will be put in the first car; and the fifth item will be put in the third car. However, we can put the third item in the first car after the first and second item was put. Then we can put the forth item in the first car and the fifth item in the second car. In this way, we just need two cars instead of three cars.

Grading You will be docked points for errors in your counter examples or a failure to show how the algorithm fails.

2.2 EXPLORE

You have two sets of positive integers $A = \{a_1, ..., a_n\}$ and $B = \{b_1, ..., b_n\}$. You want to place these sets in order to compute this sum so that it is as large as possible.

$$\sum_{i=1}^{n} a_i \cdot b_i$$

1. (4 points) What is the best ordering for these sets to maximize the sum?

   $$A = \{10, 100, 1, 1000\}$$

   $$B = \{12, 42, 8, 16\}$$

   Answer:

   $$A = \{1000, 100, 10, 1\}$$

   $$B = \{42, 16, 12, 8\}$$

   $$\sum_{i=1}^{n} a_i \cdot b_i = 1000 * 42 + 100 * 16 + 10 * 12 + 8 = 43728$$

2. (4 point) What order should you place the sets in to maximize this sum?

   Answer:

   We should place both sets in descending or ascending order at the same time to maximize the sum.

3. (12 points) Prove your ordering is correct for any two sets A and B of size n.

Theorem 1. *If we placed both sets in descending or ascending order, then the sum is maximized.*

Proof. Let S be the ordering from my algorithm and O be the optimal ordering solution. Every element in S is a pair of two elements, one from A and another one from B correspondingly. Let the both solutions be in descending order.

Assume $S_i \neq O_i$, let S_1 and O_1 be the first pair for my ordering and optimal ordering. Since we know $S_1 = a_1 * b_1$ where a_1 and b_1 are both maximal, $O_1 = a_1 * b_k$ for some valid $b_k \in B$ such that $b_k < b_1$ because b_1 is the maximal. O_1 does not multiply b_1 with a_1, there exists $b_1 * a_j \in O_i$ for some $a_j \in A$ and $a_j < a_1$.

To make them easier to compare, we will modify the optimal ordering solution, $O = a_1 * b_k + a_j * b_1 + \ldots$ into $O' = a_1 * b_1 + a_j * b_k + \ldots$. Noticed that the result of O should be greater than O' because O is the optimal ordering solution by design. O' here is exactly the set S. The difference between O and O' is $a_1 * b_k + a_j * b_1 - a_1 * b_1 + a_j * b_k + \cdots = (b_k - b_1)(a_1 - a_k) + \ldots$. $b_k - b_1$ is negative because $b_1 > b_k$ and $a_1 - a_k$ is positive. So the difference is actually negative, which means that $O' > O$. Hence the contradiction. □

Grading You will be docked points for errors in your math, disorganization, unclarity, or incomplete proofs.

2.3 EXPAND

A set of Pokemon cards consists of n different cards. The cards are sold as different sized randomly selected packs. Each pack is a subset of the set of Pokemon cards. There are m such packs. You happen to know which cards were placed in which packs. You want to select a minimum set of packs such that you will get at least one of every every card in the set.

1. (4 point) Express this problem (Pokemon) formally with input and output conditions as a *decision problem*.

   Answer:

   Pokemon

   Input: $C[1 \ldots n]$, a set of Pokemon cards consists of n different cars, and $P[1 \ldots m]$, m such packs which each pack is subsets of the set of Pokemon cards, and k numbers of packs selected.

   Output: 'Yes', if we can get a whole set of Pokemon cards with selected packs and it's less than and equal to k. Otherwise, 'no'.

2. Show this problem is in NP.

   a) (4 points) Give a verification algorithm for this problem.

   PCVerifier(C, P, k, W)

   if $|W| > k$, say 'no';

   if W is not a subset of P, say 'no';

   if $\exists C_i \in C$ such that $C_i \notin \bigcup_{j \in W} j$, say 'no';

   say 'yes';

   EndPCVerifier

   b) (4 points) Prove your algorithm is correct.

   Proof.

   'yes' ⇒ 'yes'

   Assume there is some $W \subseteq P$ such that $|W| \leq k$ and $C \subseteq \bigcup_{i \in W} i$. Let W be the proof. So W passes all three tests and we say 'yes'.

'no' ⇒ 'no'

Assume there is no $W \subseteq P$ such that $|W| \leq k$ and $C \subseteq \bigcup_{i \in W} i$.

Then there exists $W \subseteq P$ and $C \subseteq \bigcup_{i \in W} i$, so $|W| > k$ and we say 'no'. □

3. The Vertex Cover problem is already known to be NP-complete. We can show that Pokemon is NP-complete by reducing Vertex Cover to Pokemon.

   a) (4 points) Give a reduction from Vertex Cover to Pokemon. The reduction takes input to Vertex Cover and converts it into input to Pokemon.

   $VC \leq_p Pokemon$

   Reduction $m(G = (V, E), k)$

   For each edge in E, create a pokemon card;

   For each vertex in V, create a pack of cards which each pack represents all the edges connected to this vertex;

   Let $k' = k$;

   return (C, P, k')

   b) (4 points) Prove your reduction is correct.

   Proof.

   'yes' ⇒ 'yes'

   Assume G has a vertex cover of size k. Consider a subset of vertices of size k that covers all edges. Let each edge represents a pokemon card and each vertex represents a pack of cards. So we can translate the previous statement as 'there exists k' packs of cards that cover all the pokemon cards'.

   'yes' ⇐ 'yes'

   Assume there are k' packs of cards that contains all the pokemon cards. Let each pokemon card be an edge and each pack of cards be a vertex. There is a subset of vertices of size k that covers all the edges. □

Grading You will be docked points for errors in your math, disorganization, unclarity, or incomplete proofs.

2.4 CHALLENGE

Recall your trip to Tijuana from the first problem. After proving Alice and Bob wrong they are convinced their algorithms are not that bad. They have researched *approximation algorithms* and want your help to prove their algorithms are approximately correct.

1. (1 point) Restate the lower bound on the number of cars needed to pack your n items.

   The lower bound is $\lceil \frac{1}{1000} \sum_{i \in N}^{n} W[i] \rceil$.

2. (2 points) Prove that the first-fit algorithm will never leave more than one car less than half full.

   Proof. When we put the first item in the car, consider two cases:

   a) The first car is more than a half after the first item. We really don't have much things to do for this case. For next item, we either put it in the first car or put it in the second car.

   b) The first car is less than a half after the first item. Now we need to put in the second item. If the second item is less than or equal to a half, we are going to put the second item in the first car; if the second item is greater than a half, we probably can put it in the first car depends on how many space left or we need another car.

   So we are using the first-fit algorithm, if the algorithm lefts two cars less than half full, this is just impossible because the item in the second car will be put in the first car based on the algorithm. □

3. (2 points) Prove that the best-fit algorithm will never leave more than one car less than half full.

   Proof. For this proof, it will be very similiar to the previous proof. Assume the algorithm left us two cars less than half full. This is just simply impossible. Based on the best-fit algorithm, the item in the second car must be put into the first car because the algorithm says that an item fits best in a car if it leaves the car with the least amount of free space. □

4. (5 points) Prove that if an algorithm never leaves more than one car less than half full that it will never use more than twice as many cars as is used by an optimal solution.

   Proof. So, we have a lower bound from part 1, $\lceil \frac{1}{1000} \sum_{i \in N}^{n} W[i] \rceil$. That should be the least number of cars possible for our algorithm. It can't be less than the lower bound.

   Consider the worst case scenario, we left one of the cars less than half full and other cars have exactly a half free spaces. The algorithm should leaves cars more than or equal to half full except one. When we keep putting in items weighted 500 pounds, it's going to use a new car every time we put in the next item. The cars we are going to use should be $\lceil \frac{1}{500} \sum_{i \in N}^{n} W[i] \rceil = 2 \lceil \frac{1}{1000} \sum_{i \in N}^{n} W[i] \rceil$, which is exactly twice as many cars as the lower bound, AKA the best possible solution.

   □

Grading You will be docked points for errors in your math, disorganization, unclarity, or incomplete proofs.