Due: Tuesday, May 2, 2023, 11:59pm

Submit online at (clickable link): `https://forms.gle/DEb6CNyayoX3oXSY7`

---

1 Special Partition

Sometimes the special case of an NP-complete problem remains NP-complete. For example, the 3-Color problem restricted to planar graphs remains NP-complete. For this problem we consider the opposite situation: a special case of Partition that can be solved by a greedy algorithm.

Recall that in the Partition problem, we are given n numbers $a_1, \ldots, a_n$. The question is whether there is partition of the indices $\{1, \ldots, n\}$ into sets A and B such that

$$\sum_{i \in A} a_i = \sum_{j \in B} a_j$$

Here, A and B forms a partition means $A \cap B = \emptyset$ and $A \cup B = \{1, \ldots, n\}$. In this special case, all of the numbers are powers of 2. That is, the a_i are chosen from $\{1, 2, 4, 8, 16, \ldots\}$. Duplicates are allowed, so there maybe multiple 4's for example.

Assignment:

1. First prove this useful lemma:

   Lemma: Let $x_0, x_1, \ldots, x_k \in \mathbb{N}$ such that
   $$x_0 2^0 + x_1 2^1 + x_2 2^2 + \cdots + x_k 2^k \geq 2^{k+1}$$
   then there exists $y_0, y_1, \ldots, y_k \in \mathbb{N}$ such that each $y_i \leq x_i$ and
   $$y_0 2^0 + y_1 2^1 + y_2 2^2 + \cdots + y_k 2^k = 2^{k+1}$$

   The lemma says that if there is a way to combine lower powers of 2 to exceed 2^{k+1}, then there is a way to combine a subset of them to total exactly 2^{k+1}.

   Prove the lemma above using proof by induction.

2. Devise and describe a greedy algorithm that finds a partition A and B of $\{1, 2, 3, \ldots, n\}$ such that
   $$\sum_{i \in A} a_i = \sum_{j \in B} a_j$$
   where each a_i is a power of 2. Your algorithm should report either the sets A and B, or declare that no such partition is possible.

   Note: the fact that $\{a_1, \ldots, a_n\}$ may contain repeated values is an important part of this problem. You should find the lemma above useful.

3. State and briefly justify the running time of your algorithm.

4. Argue that any partition A and B produced by your algorithm does indeed have
   $$\sum_{i \in A} a_i = \sum_{j \in B} a_j$$

5. Argue that if your algorithm declares that no partition is possible, it is indeed correct.

   Hint: it is easier to prove the contrapositive that if some partition A' and B' of the indices exist then your algorithm will report a partition A and B.

2 Improving Approximations for Clique

In this problem, we will show that it is possible to improve the approximation factor for any algorithm that approximates the Clique problem. Much of the construction is given by the problem. You have to complete the tasks that are embedded below.

Let $G = (V, E)$ be any undirected graph and let $n = |V|$. We construct a new graph $G^{(4)}$ that has n^4 vertices. Each vertex of $G^{(4)}$ is a 4-tuple of vertices in G:

$$V^{(4)} = \{\ (v_1, v_2, v_3, v_4) \mid v_i \in V,\ \text{for } i = 1, 2, 3, 4\ \}.$$

If we used Cartesian product notation, $V^{(4)} = V \times V \times V \times V$.

Now, we define the set of edges for $G^{(4)}$. Given two vertices (u_1, u_2, u_3, u_4) and (v_1, v_2, v_3, v_4), we place an edge between these two vertices if for every $i = 1, 2, 3, 4$, either $u_i = v_i$ or $(u_i, v_i) \in E$. Let's call this set of edges $E^{(4)}$. Then, we have fully defined $G^{(4)}$ by saying $G^{(4)} = (V^{(4)}, E^{(4)})$.

Task 1: Let $C \subseteq V$ be a clique in G. Argue that

$$C^{(4)} = \{\ (v_1, v_2, v_3, v_4) \mid v_i \in C,\ \text{for } i = 1, 2, 3, 4\ \}$$

must be a clique in $G^{(4)}$.

Task 2: Let $C' \subseteq V^{(4)}$ be a clique in $G^{(4)}$. Argue that

$$C_1 = \{\ x \mid x = v_1 \text{ for some } (v_1, v_2, v_3, v_4) \in C'\ \}$$

must be a clique in G.

For C', we can similarly define

$$C_2 = \{\ x \mid x = v_2 \text{ for some } (v_1, v_2, v_3, v_4) \in C'\ \}$$

$$C_3 = \{\ x \mid x = v_3 \text{ for some } (v_1, v_2, v_3, v_4) \in C'\ \}$$

$$C_4 = \{\ x \mid x = v_4 \text{ for some } (v_1, v_2, v_3, v_4) \in C'\ \}.$$

Using the same argument as for C_1, each C_i must also be a clique in G.

Task 3: Argue that $|C'| \leq |C_1| \cdot |C_2| \cdot |C_3| \cdot |C_4|$.

Suppose that the largest clique in G has k vertices and the largest clique in $G^{(4)}$ has k' vertices. Using the above, we can show a tight relationship between k and k'.

Task 4: Show that $k' \geq k^4$.

Task 5: Show that $k' \leq k^4$.

Thus, we can conclude that k' is exactly equal to k^4.

Now suppose that you are given a clique C' in $G^{(4)}$ along with the guarantee that C' is a factor 10 approximation. That is $|C'| \geq k'/10$ where k' is once again the size of the largest clique in $G^{(4)}$.

Task 6: Explain how to construct a clique $C \subseteq V$ in G so that $|C| \geq k/\sqrt[4]{10}$ where k is the size of the largest clique in G. Do explain why C must have at least $k/\sqrt[4]{10}$ vertices.

Suppose that some algorithm $\mathcal{A}$ can always find a clique in any graph and guarantee that the size of that clique is within a factor of 10 of the largest clique. Then, we can run $\mathcal{A}$ on $G^{(4)}$ and obtain C'. Using your construction above, we can get a C that is within a factor of $\sqrt[4]{10}$ of k. Since $\sqrt[4]{10} \approx 1.78$, we have improved the approximation factor from 10 to about 1.78. The new algorithm is: given G, construct $G^{(4)}$, run $\mathcal{A}$ on $G^{(4)}$ to get C', construct C from C' and output C.

Presumably there was nothing special about the factor 10 in your construction and you can turn an algorithm that achieves a factor r approximation of Clique into an algorithm that achieves a factor $\sqrt[4]{r}$ approximation. For example, if we repeated the process on the factor $\sqrt[4]{10}$ algorithm, we get a factor $\sqrt[4]{\sqrt[4]{10}} \approx 1.15$ algorithm. If we do this yet again, we get a factor $\sqrt[4]{\sqrt[4]{\sqrt[4]{10}}} \approx 1.04$ algorithm. We can get arbitrarily close to a factor of 1, but we pay a price in running time.

Task 7: Suppose that we have an algorithm $\mathcal{A}$ that produces a factor 10 approximation of Clique and that $\mathcal{A}$ runs in $\Theta(n^d)$ time on graphs with n vertices. Using the process above we can construct an algorithm that produces a factor $(1+\epsilon)$ approximation of Clique. Bound the running time of such an algorithm. Assume that it takes $\Theta(n^2)$ time to construct a graph with n vertices and that $d \geq 2$. Give the running time in terms of n, d and ϵ. Show your work.

Task 8: Using your calculations above, suppose that $d = 2$ and you want an algorithm that produces a factor 1.001 approximation of Clique. What is the running time of that algorithm? Give your answer as $\Theta(n^\ell)$ and provide an actual number for ℓ. Show your work.