Problem 5

> Refs & people discussed with:
>
> b09902011

1

Reduction

1. If $T > \sum a_+$ or $T < \sum a_-$, just return "no". $a_+ = \{a_i | a_i > 0\}$ and $a_- = \{a_i | a_i < 0\}$.
2. Choose $K = 1 + \sum_{i=0}^{n} |a_i|$.
3. Define transformation f as:

$$f(a_1, a_2, \cdots, a_n; T) = (a_1 + K, a_2 + K, \cdots, a_n + K, \underbrace{K, \cdots, K}_{nK's}; T + nK)$$

Correctness

The first step filters impossible cases.

Let $L = (a_1, a_2, \cdots, a_n; T)$ be an instance of $JJBAP$, $f(L)$ be an instance of $JJBAP_+$.

L is yes $\Rightarrow f(L)$ is yes

Suppose S is a subsequence of a that solves L. $\sum S = T$.

The solution S' to $f(L)$ can be constructed by selecting $s_i + K$ for all $s_i \in S$ and $n - |S|$ K's.

$$\sum S' = (\sum S) + |S|K + (n - |S|)K = T + nK$$

$f(L)$ is yes $\Rightarrow L$ is yes

Suppose S' solves $f(L)$. $\sum S' = T + nK$.

The solution S to L can be constructed by selecting $s_i' - K$ from a for all $s_i' \in S'$. During selection, $s_i' - K = 0$ but there are no 0's left, it means s_i' correspond to one of the n K's'.

Because $K > \sum |a_i| > \max |a_i|$, for any subsequence C of $\{a_1 + K, a_2 + K, \cdots, a_n + K, \underbrace{K, \cdots, K}_{nK's}\}$:

$$\begin{aligned}
(-\sum |a_i|) + |C|K \leq \sum C &\leq (\sum |a_i|) + |C|K \\
(|C| - 1)K \leq \sum C &\leq (|C| + 1)K \\
\frac{\sum C}{K} - 1 \leq |C| &\leq \frac{\sum C}{K} + 1
\end{aligned}$$

$$\begin{aligned}
\frac{T + nK}{K} - 1 = \frac{T}{K} + (n - 1) \leq |S'| &\leq \frac{T + nK}{K} + 1 = \frac{T}{K} + (n + 1) \\
n - 1 < |S'| &< n + 1
\end{aligned}$$

Therefore S' has exactly n elements, and:

$$\sum S = (\sum S') - |S'|K = T + nK - nK = T$$

Polynomial time complexity

Detection in the first line and the transformation can be naively done in $O(n)$ time.

2

Reduction

1. Transform $JJBAP$ instance L to $JJBAP_+$ instance $f(L) = (b; T')$ by the method in subproblem 1.
2. Add $b_{2n+1} = |\sum b - 2T'|$ to construct sequence b'.

Correctness

Transformation from $JJBAP$ to $JJBAP_+$ has been proved in subproblem 1. Therefore we prove $JJBAP_+$ is yes $\iff$ QQP is yes.

$JJBAP_+$ is yes $\Rightarrow QQP$ is yes

Suppose S solves $JJBAP_+$, and $\hat{S} = \{b_i | b_i \in (b - S)\}$. Let $H = \sum b$.

$\sum S = T'$ and $\sum \hat{S} = H - T'$.

- If $H \geq 2T'$
  - Add $b_{2n+1} = H - 2T'$ to S to construct S'.
  - $$\sum S' = T' + (H - 2T') = H - T' = \frac{H + (H - 2T')}{2} = \frac{\sum b + b_{2n+1}}{2} = \frac{\sum b'}{2}$$
- If $H < 2T'$
  - Add $b_{2n+1} = 2T' - H$ to $\hat{S}$ to construct $\hat{S}'$.
  - $$\sum \hat{S}' = (H - T') + (2T' - H) = T' = \frac{H + (2T' - H)}{2} = \frac{\sum b + b_{2n+1}}{2} = \frac{\sum b'}{2}$$

QQP is yes $\Rightarrow JJBAP_+$ is yes

Suppose S' solves QQP and $\hat{S}' = \{b_i | b_i \in (b - S)\}$.

- If $H \geq 2T'$
  - $\sum S' = \frac{H+(H-2T')}{2} = H - T' = \sum \hat{S}'$
  - Remove b_{2n+1} from the subsequence it belongs, and the sum of that subsequence becomes $H - T' - (H - 2T') = T'$.
- If $H < 2T'$
  - $\sum S' = \frac{H+(2T'-H)}{2} = T' = \sum \hat{S}'$
  - The subsequence that doesn't contain b_{2n+1} has sum T'.

Polynomial time complexity

First step takes polynomial time (shown in subproblem 1). Second step takes another $O(2n) = O(n)$ time.

3

$DDBP$

Given n balls with weight $a_1, \cdots, a_n \in [0,1]$ and bound N. Is it possible to partition balls into less or equal to N bins such that all bins weigh at most 1 kilogram.

Reduction

1. Let $M = \frac{\sum a}{2}$. If there is an $a_i > M$, return no. Because QQP obviously has no solution in this case.
2. Define transformation:

$$f(a_1, a_2, \cdots, a_n) = (\frac{a_1}{M}, \frac{a_2}{M}, \cdots, \frac{a_n}{M}; 2)$$

Correctness

QQP is yes $\Rightarrow$ $DDBP$ is yes

Suppose S solves QQP. $\sum S = \sum(a - S) = M$.

Simply choose $\frac{s_i}{M}$ to construct S'. $\sum S' = \sum \frac{s_i}{M} = \frac{\sum S}{M} = 1$.

The remaining balls has sum $\sum(a' - S') = \frac{\sum a}{M} - 1 = 2 - 1 = 1$.

$DDBP$ is yes $\Rightarrow$ QQP is yes

Suppose a' is partition into S_1' and S_2'. $\sum S_1' \leq 1$ and $\sum S_2' \leq 1$

Because $\sum a' = \sum \frac{a_i}{M} = \frac{\sum a}{M} = 2$, $\sum S_1' = \sum S_2' = 1$.

Transform each element in S_1' and S_2' back yields S_1 and S_2. Both have sum $1 \times M = M = \frac{\sum a}{2}$.

Polynomial time complexity

Both steps in reduction takes $O(n)$ time.

$QQP \leq_p DBP$

The decision version $DDBP \leq_p DBP$. And as shown above, $QQP \leq_p DDBP$.

By transitivity, $QQP \leq_p DBP$.

4

NP-ness

If S is said to solve QQP, we can verify by summing S and a, which is done in $O(n)$ (polynomial) time.

If partition P is said to solve $DDBP$, we can verify by summing and checking each bin in $O(n)$ (polynomial) time.

NP-completeness

Since $JJBAP$ is known to be NP-complete, and by the result above we have $JJBAP \leq_p QQP \leq_p DDBP$. Therefore QQP and $DDBP$ is NP-hard.

Because QQP and $DDBP$ are both NP and NP-hard, they are NP-complete.

5

Suppose there exist a $\frac{3}{2} - \epsilon$-approximation polynomial time algorithm Orcale for DBP.

Transform an instance of QQP to DBP by $f(a_1, a_2, \cdots, a_n) = (\frac{a_1}{M}, \frac{a_2}{M}, \cdots, \frac{a_n}{M})$ where $M = \frac{\sum a}{2}$.

If $(a_1, \cdots, a_n)$ is solvable for QQP, DBP should output 2, and Orcale's output c should satisfy:

$$\begin{aligned} \frac{c}{2} &\leq \frac{3}{2} - \epsilon \\ c &\leq 3 - 2\epsilon \\ c &< 3 \\ \Rightarrow c &= 2 \end{aligned}$$

And for any unsolveable $(a_1, \cdots, a_n)$, $c \geq 3$. So $QQP(a_1, \cdots, a_n)$ is yes $\iff$ Oracle outputs 2.

Therefore, Oracle can solve QQP in polynomial time. But since we assume P $\neq$ NP and QQP is NP-complete, Oracle must not exist.

6

Because the lowest weight per ball is c, the largest number of balls that fits is $\lfloor \frac{1}{c} \rfloor$.

Denote the number of type i balls chosen into a single bin be x_i. To calculate the upper bound of possible choices, let's assume that they all have enough supply.

This should be satisfied:

$$\begin{aligned} \sum_{i=1}^{m} x_i &\leq \lfloor \frac{1}{c} \rfloor \\ x_i &\geq 0 \qquad\qquad \forall\, 1 \leq i \leq m \end{aligned}$$

Let $x_{m+1} = \lfloor \frac{1}{c} \rfloor - \sum_{i=1}^{m} x_i$. We can rewrite it as:

$$\begin{aligned} \sum_{i=1}^{m+1} x_i &= \lfloor \frac{1}{c} \rfloor \\ x_i &\geq 0 \qquad\qquad \forall\, 1 \leq i \leq m+1 \end{aligned}$$

And the upper bound of possible choices is the combinations of x_i:

$$\frac{(\lfloor \frac{1}{c} \rfloor + m)!}{(\lfloor \frac{1}{c} \rfloor)! m!}$$

7

Suppose there are x_i bins with i-th "choice of balls in a bin". This should be satisfied:

$$\sum_{i=1}^{M} x_i = k$$
$$x_i \geq 0 \qquad \forall\, 1 \leq i \leq M$$

And the number of "combination of x_i that satisfies this" is:

$$\begin{aligned}
\frac{(k+M-1)!}{k!(M-1)!} &= \frac{1}{(M-1)!}(k+(M-1))(k+(M-2))\cdots(k+1) \\
&< 1 \times (k+M)^{M-1} \\
&\leq (Mk+kM)^{M-1} \text{ (Because } k > 1 \text{ and } M > 1\text{)} \\
&< (Mk+kM)^{M} = (2M)^M k^M
\end{aligned}$$

Therefore it can be bound with $C_M = (2M)^M$.

8

Algorithm

1. Compute c and m for the given n balls.
2. Generate all "choices of balls that fits in a bin" by brute-force.
3. Start with lower bound 1 and upper bound n, binary search the minimum number of needed bins.
   - To examine a mid-point p, brute-force all possible combinations and check each combination if it is valid.
4. Return the result of binary search.

Time complexity

1. Use a set to maintain how many kinds of weight there are. Step 1 takes $O(n)$ time.
2. Brute-forcing went through all possible combinations. Step 2 takes $O(M)$ time.
3. Each examination takes $O(n^M) \cdot O(n)$ time. So the entire binary search takes $O(\log n) \cdot O(n^{M+1})$ time.

Total time complexity is $O(n) + O(M) + O(\log n) \cdot O(n^{M+1}) = O(n^{M+1} \log n) = O(n^{M+2})$ (is polynomial).