IN4301 Advanced Algorithm - Homework Assignment 2 - Lu Dai (4506677)

Question 1

1.
According to the algorithm, a new bin is added when the current used bins can not fit the item anymore. Thus, **the total weight of any two random bins are bigger than 1.**
If there are 2 bins having load $\leq \frac{1}{2}$, then the two bins can be merged. Thus, at most 1 bin can have load $\leq \frac{1}{2}$. **Intuitively, there are at least k-1 bins having total load bigger than** $\frac{1}{2}$. So $\sum_{i=1}^{n} w_i > \frac{k-1}{2}$.
To explain more, Let L_i denotes the total load in bin b_i.

> For b_1, b_2, we know $L_1 + L_2 > 1$
> For b_1, b_2, b_3,, we know $L_1 + L_3 > 1, L_2 + L_3 > 1, L_1 + L_2 > 1$
> ...
> For $b_1, b_2, b_3, \ldots, b_k$, the sum of total load of any pair of bins is larger than 1. This means $L_i + L_j > 1$ $_{(i \neq j,\ 1 \leq i \leq k,\ 1 \leq j \leq k)}$

- if k is odd:
  $\sum_{i=1}^{n} w_i = (l_1 + l_2) + (l_3 + l_4) + \ldots + (l_{k-2} + l_{k-1}) + l_k > \frac{k-1}{2} + l_k > \frac{k-1}{2} = \lfloor \frac{k}{2} \rfloor$
- if k is even:
  $\sum_{i=1}^{n} w_i = (l_1 + l_2) + (l_3 + l_4) + \ldots + (l_{k-1} + l_k) > \frac{k}{2} = \lfloor \frac{k}{2} \rfloor$

Above all, the total load of the k bins used by A is at least $\lfloor \frac{k}{2} \rfloor$, generally formulated as $\sum_{i=1}^{n} w_i \geq \lfloor \frac{k}{2} \rfloor$.

2.
From previous results, we know $\sum_{i=1}^{n} w_i \geq \lfloor \frac{A(x)}{2} \rfloor \geq \frac{A(x)-1}{2}$.
As the optimal number of bins is opt(x), so $\sum_{i=1}^{n} w_i \leq opt(x) * 1$.
Therefore,

$$\frac{A(x) - 1}{2} \leq \sum_{i=1}^{n} w_i \leq opt(x) * 1$$
$$A(x) \leq 2 * opt(x) + 1$$

3.
Consider ϵ is very small and there are only three different sizes, we discuss all the possible situations which a bin can load and the space wasted:

No.	Items	Waste space	No.	Items	Waste space
1	6 ($\frac{1}{7} + \epsilon$)	$\sim \frac{1}{7}$	4	4 ($\frac{1}{7} + \epsilon$) + 1 ($\frac{1}{3} + \epsilon$)	$\sim \frac{2}{21}$
2	2 ($\frac{1}{3} + \epsilon$)	$\sim \frac{1}{3}$	5	2 ($\frac{1}{7} + \epsilon$) + 2 ($\frac{1}{3} + \epsilon$)	$\sim \frac{1}{21}$
3	1 ($\frac{1}{2} + \epsilon$)	$\sim \frac{1}{2}$	6	3 ($\frac{1}{7} + \epsilon$) + 1 ($\frac{1}{2} + \epsilon$)	$\sim \frac{1}{14}$
7	1 ($\frac{1}{3} + \epsilon$) + 1 ($\frac{1}{2} + \epsilon$)	$\sim \frac{1}{6}$	8	1 ($\frac{1}{7} + \epsilon$) + 1 ($\frac{1}{3} + \epsilon$) + 1 ($\frac{1}{2} + \epsilon$)	$\sim \frac{1}{42}$

From the table, we know that using **any combination is always better than packing items with the same size**. So the worst case is that a bin is always loaded with items of the same size, while the best case is the 8th combination which minimize the total waste space.

- Worst case: $A_{worst}(x) = \frac{m}{6} + \frac{m}{2} + \frac{m}{1} = \frac{5}{3}m$
- Best Case: $opt(x) = \frac{3m}{3} = m$

Thus, $A(x) \leq \frac{5}{3}opt(x)$ means the performance ratio on such instances is at least $\frac{5}{3}$.

4.
Let L_i denotes the total weight in bin b_i. Next-Fit gets the results $(L_1, L_2, \ldots, L_{k-1}, L_k)$. According to its definition, we can easily deduce that **the sum of weight of neighboring bins is always larger than 1, which is $L_i + L_{i+1} > 1$.** There are generally two situations,

- (1) $L_i > \frac{1}{2}, L_{i+1} > \frac{1}{2}$
- (2) $(L_i > \frac{1}{2}, L_{i+1} \leq \frac{1}{2})\ or\ (L_i \leq \frac{1}{2}, L_{i+1} > \frac{1}{2})$

For situation (1), no bins can be merged anymore, thus the result stays k.
For situation (2). **At most**, there can be $\lceil \frac{k}{2} \rceil$ bins which has total load less than $\frac{1}{2}$ while at least $\lfloor \frac{k}{2} \rfloor$ bins larger than $\frac{1}{2}$. This happens for every two neighbouring bins, there is one bin having load $\leq \frac{1}{2}$.
Intuitively, the best case happens when we can merge these lighter $\lceil \frac{k}{2} \rceil$ bins togethers into 1 bin assuming their weights are really small. Then the optimized result become the heavier $\lfloor \frac{k}{2} \rfloor$ bins plus the merged 1 bin which is $\lfloor \frac{k}{2} \rfloor + 1$. Thus,

$$\frac{k}{2} + 1 \geq \lfloor \frac{k}{2} \rfloor + 1 \geq OPT$$
$$k \geq 2 * (OPT - 1) = (2 - \frac{2}{OPT}) * OPT$$

Thus, the performance ratio of Next-Fit can't be better than $2 - \frac{2}{OPT}$.

Question 2

As X is an NP-hard minimization problem, thus for every instance x, $c(x) \geq opt(x)$.
Suppose there exist an FPTAS algorithm with approximation ratio $1 + r\ \ (r \geq 0)$. For all instances, $c(x) \leq (1 + r) * opt(x)$.
Choosing $0 \leq \epsilon \leq r$ which satisfies $\epsilon^{-1} > p(|x|)$ for some instances. Then, for these instances, we have $c(x) \leq (1 + \epsilon) * opt(x)$ and $\epsilon * p(|x|) < 1$.
As it is given that $opt(x) \leq p(|x|)$, therefore

$$opt(x) \leq c(x) \leq opt(x) + \epsilon * opt(x) \leq opt\ (\ x) + \epsilon * p(|x|)$$
$$opt(x) \leq c(x) < opt(x) + 1.$$

As it is given c(x) is always an interger, thus it is obviously $c(x) = opt(x)$. This means the FPTAS algorithm can always produce the optimal solution in polynomial time.
Above all, if X has sucn an FPTAS algorithm, then X can be solved in polynomial time which implies P = NP. Thus, X doesn't have FPTAS unless P=NP.