CS:3330 Homework 9 Solution, Spring 2018

1(a) $B = 10$, $a_1 = 2, a_3 = 9$. For this input, the algorithm returns $S = \{2\}$, but there is a feasible solution $\{9\}$ whose total sum is more than twice the total sum of S.

1(b) First sort A in non-increasing order and then relabel the numbers $a_1, a_2, \ldots, a_n$ in this order. The run the algorithm in part (a).

1(c) The proof is by contradiction. Suppose that there is a feasible subset S^* such that the total sum of S (the subset returned by the algorithm) is less than half the total sum of S^*. Then there is an element $a \in S^*$ such that $a \notin S$. Now let us consider why a is not added to S. The answer is simple: a is not added to S because when a is considered by the algorithm, adding a to the elements currently in S causes the total sum to exceed B. Therefore, the total sum of S plus a is greater than B. Hence, (i) either the total sum of S is greater than $B/2$, in which case we have proved the claim or (ii) $a > B/2$. So we assume (ii) holds. Since the algorithm considers elements in non-increasing order, the elements already in S when a was considered are also $> B/2$. Thus the total sum of S is $> B/2$.

2(a) Consider an input with three intervals, A, B, and C. Suppose that B is a short interval (say, 1 unit long) and A and C are long intervals (say, each 2 units long). Also, suppose that A starts first, then B starts, then A ends, then C starts, then B ends, and finally C ends. Thus B overlaps with A and C, but A and C are non-overlapping with each other. The "shortest interval first" algorithm outputs $\{B\}$, whereas the optimal solution is $\{A, C\}$.

2(b) Suppose that $|O| = t$ and the intervals in O are labeled $x_1, x_2, \ldots, x_t$ in left-to-right order. To obtain a contradiction, we suppose that there is an interval $y \in A$ such that y overlaps with 3 or more intervals in O. Call the intervals that y overlaps: $x_i, x_{i+1}, \ldots, x_{i+p}$, where $p \geq 2$. Since y overlaps x_i and x_{i+2}, the interval x_{i+1} starts after the start time of y and ends before the end time of y. Thus x_{i+1} is strictly shorter than y. The question then is why did the "shortest Interval first" algorithm not pick x_{i+1} instead of y. The only reason for not picking x_{i+1} is that the algorithm picked an interval x' even shorter than x_{i+1} and x' overlapped with x_{i+1} and eliminated it. But, any interval x' that overlaps with x_{i+1} will also overlap with y and eliminate it. Thus y cannot be in A – a contradiction.

2(c) (i) Each interval in A is charged at most 2 dollars. (ii) Thus the total number of dollars charged is *at most* $2|A|$. We already know that the total number of dollars charged is $|O|$. Therefore, $|O| \leq 2|A|$ and equivalently $|A| \geq 1/2 \cdot |O|$. (iii) This tells us that the "shortest interval first" algorithm always produces a solution whose size is at least $1/2$ the size of an optimal solution. Therefore, this algorithm is a $1/2$-approximation.

3(a) Suppose that are two bins B_i and B_j, $j > i$, such that both are half empty or more. Then B_j contains an item of size at most 0.5. When this item was processed, B_i had enough space for it and the item would have been placed in B_i. Hence, it cannot be the case that both B_i and B_j are half empty or more.

Suppose that the First Fit algorithm uses t bins. We know from the above argument that at least $t - 1$ of these are more than half full and therefore the total size of the all items in the input is more than $(t - 1)/2$.

3(b) Suppose that an optimal bin packing uses b^* bins and suppose that the First Fit algorithm uses t bins. By the argument in (a) we know that the total input size is more than $(t-1)/2$ and since eash bin has size 1 unit, $b^* > (t - 1)/2$. Hence, $t < 2b^* + 1$, implying that the First Fit algorithm uses at most $2b^*$ bins.