12.3.2 (a)

The problem KNAPSACK is the following:

> We are given n items and each item i is attached with a value v_i and a weight w_i, both positive integers. Given a goal K and a weight limit W, is there a set $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i \leq W$ and $\sum_{i \in S} v_i \geq K$?

We know that KNAPSACK is **NP**-complete (Theorem 9.10 in the book).

We now consider a special case of KNAPSACK (which is also **NP**-complete by Theorem 9.10): we assume that $v_i = w_i$ for all i and that $K = W$.

Our goal is now to show that if $w_{i+1} > 2w_i$, then it is easy to solve KNAPSACK. So the problem is now reduced to the following: "Is there a set $S \subseteq \{1, 2, \ldots, n\}$ such that $\sum_{i \in S} w_i = W$?". Because $w_{i+1} > 2w_i$, it is easy to show that $w_{i+1} > \sum_{1 \leq j < i} w_j$. Now the problem can be solved by the following procedure:

$S := \emptyset$
$W' := W$
$i := n$
do
  if $w_i \leq W'$ **then**
    $S := S \cup \{i\}$
    $W' := W' - w_i$
  $i := i - 1$
while $i > 0$
return *true* iff $W' = 0$

The procedure is clearly correct since if the weight to cover is W' and the largest unprocessed weight is w_i, $w_i \leq W'$ has to be in S because with all other unprocessed weights w_j, $j < i$, we cannot sum up to W'.

13.4.5

Preliminaries

The problem BINPACKING is the following:

> We are given N positive integers $a_1, a_2, \ldots, a_N$ (the items), and two more integers C (the capacity) and B (the number of bins). Can the items be partitioned into B subsets, each of which has total sum at most C?

Theorem 9.11 in the book establishes that BINPACKING is **NP**-complete. In the following we study the optimization version of BINPACKING where we try to find the smallest value of B.

(a)

The problem is to show that the following "first-fit" heuristic algorithm leaves at most one (used) bin less than half-full.

$B[j] := 0$ for $j = 1, 2, \ldots, n$
for $i = 1, 2, \ldots, n$ **do**
  Find the smallest j with $B[j] + a_i \leq C$
  and set $B[j] := B[j] + a_i$

Initially there are no used bins and thus no used bins less than half-full. From a situtuation in which there are no used less than half-full bins, one execution of the for-loop body can produce at most one such bin. Suppose now that there is a j such that $0 < B[j] < \frac{C}{2}$. Now if the current a_i is less or equal to $\frac{C}{2}$, then it is put into $B[j]$ and after this there is at most one used bin less than half-full. On the other hand, if a_i is greater than $\frac{C}{2}$, then it is put into a yet unused bin and after this there is at most one used bin ($B[j]$) less than half-full. Q.E.D.

(b)

Assume the result of an invocation of the above "first-fit" heuristic algorithm, using k bins. We now claim that the minimum number of bins into which a_is can be partitioned is at least $\lceil \frac{k}{2} \rceil$. Let the bins, let's say bins $1, 2, \ldots, k-1$, that are at least half-full be exactly half-full and assume that the capacity C is even. Then, (i) if k is even, the items in bins $1, \ldots, k$ can be packed into $\frac{k}{2} = \lceil \frac{k}{2} \rceil$ bins and (ii) if k is odd, then items in bins $1, \ldots, k$ can be packed into $\frac{k+1}{2} = \lceil \frac{k}{2} \rceil$ bins. Now $\frac{|c(M(x)) - \mathrm{OPT}(x)|}{\max\{\mathrm{OPT}(x), c(M(x))\}} \leq \frac{\frac{k}{2}}{k} \leq \frac{1}{2}$.

Therefore, the "first-fit" algorithm is an $\frac{1}{2}$-approximation algorithm for BINPACKING and thus the approximation threshold for BINPACKING is at most $\frac{1}{2}$.