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 $w_{i+1} > \sum_{1 \leq j \leq i} w_j$. 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'.

13.4.3

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 of a bin) 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.

Problem and Solution

We have to show that the approximation threshold for BINPACKING is at least $\frac{1}{3}$.

First note that if the approximation threshold for BINPACKING is less than $\frac{1}{3}$, then there is a polynomial time (approximation) algorithm M such that it can distinguish whether the optimal number of bins is 2 or 3 since $\frac{|c(M(x)) - \mathrm{OPT}(x)|}{\max\{c(M(x)), \mathrm{OPT}(x)\}} = \frac{3-2}{3} = \frac{1}{3}$. That is, given a BINPACKING instance for which the optimum number of bins is 2, the polynomial time algorithm would be bound to answer 2.

Now consider the following problem called PARTITION:

> Given N positive integers $a_1, a_2, \ldots, a_N$ adding up to $2K$, ¿is there a subset that adds up to exactly K?

PARTITION is **NP**-complete (problem 9.5.33 in the Book). Now, given an instance $a_1, a_2, \ldots, a_N$ of PARTITION, form the instance $a_1, a_2, \ldots, a_N$, $C = K = \frac{1}{2} \sum_{1 \leq i \leq N} a_i$ of optimization version of BINPACKING. Clearly the optimum number of bins is 2 iff the original PARTITION instance has a subset adding up to exactly K.

To sum up, if the approximation threshold for BINPACKING is less than $\frac{1}{3}$, then we have a polynomial time algorithm for BINPACKING and thus **P** would equal to **NP**.

13.4.5

We consider again the optimization version of BINPACKING.

(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 than 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.

(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 $\lceil \frac{k}{2} \rceil$. First, $k-1$ bins are at least half-full. Then, (i) if k is even, the items in bins $1, \ldots, k$ require at least $\frac{k}{2} = \lceil \frac{k}{2} \rceil$ bins in the optimal case and (ii) if k is odd, then items in bins $1, \ldots, k$ require at least $\frac{k+1}{2} = \lceil \frac{k}{2} \rceil$ 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 $\frac{1}{2}$.