We present NP-completeness proofs of some more problems, and we introduce the model of Boolean circuits.

# 1 Bin Packing

The **Bin Packing** problem is one of the most studied optimization problems in Computer Science and Operation Research, possibly the second most studied after TSP. It is defined as follows:

- Given items of size $a_1, \ldots, a_n$, and given unlimited supply of bins of size $B$, we want to pack the items into the bins so as to use the minimum possible number of bins.

The search version of the problem is:

- Given items of size $a_1, \ldots, a_n$, given bin size $B$, and parameter $k$,
- Find a way to pack all the items in $k$ bins of size $B$, if it is possible to do so.

Clearly the problem is in NP. We prove that it is NP-hard by reduction from Partition.

Given items of size $a_1, \ldots, a_n$, make an instance of Bin Packing with items of the same size and bins of size $(\sum_i a_i)/2$. Let $k = 2$.

There is a solution for Bin Packing that uses 2 bins if and only if there is a solution for the Partition problem.

# 2 Scheduling to Minimize Makespan

We consider a scheduling problem in which we are given $k$ identical machines and $n$ tasks, where task $i$ takes $t_i$ to be executed on a machine. The tasks cannot be interrupted. We want to assign tasks to machines in order to minimize the makespan of the schedule, which is the maximum, over the machines, of the sum of the times of the tasks assigned to that machine.

Formalized as a seach problem:

- Given task completion times $t_1, \ldots, t_n$, given the number of machines $k$, and given a makespan target bound $T$,
- Find a way to assign the tasks to the $k$ machines so that that the sum of times in each machine is at most $T$, if it is possible

We can immediately see that, as search problems, scheduling to minimize makespan and bin packing are exactly the same problem, so the scheduling problem is NP-hard even in the special case of 2 machines.

# 3 Hamiltonian Cycle

The Hamiltonian Cycle problem is, given an undirected graph $G = (V, E)$ to find a simple cycle that goes through every vertex of the graph exactly once. The problem is called Rudrata Cycle in the textbook. In class we presented a reduction from Vertex Cover for which we will provide a reference separately. The book presents a different reduction, which is also worth studying.

In the book you can see that, from Hamiltonian Cycle, you can get NP-completeness of Longest Path (which also implies the NP-completeness of shortest simple path in graphs with negative edge lengths) and of TSP.