Optimal schedule of jobs given their deadlines and durations

Table of Contents

- Solving
- Implementation

Suppose, we have a set of jobs, and we are aware of every job's deadline and its duration. The execution of a job cannot be interrupted prior to its ending. It is required to create such a schedule to accomplish the biggest number of jobs.

Solving

The algorithm of the solving is **greedy**. Let's sort all the jobs by their deadlines and look at them in descending order. Also, let's create a queue q, in which we'll gradually put the jobs and extract one with the least run-time (for instance, we can use set or priority_queue). Initially, q is empty.

Suppose, we're looking at the i-th job. First of all, let's put it into q. Let's consider the period of time between the deadline of i-th job and the deadline of $i-1$-th job. That is the segment of some length T. We will extract jobs from q (in their left duration ascending order) and execute them until the whole segment T is filled. Important: if at any moment of time the extracted job can only be partly executed until segment T is filled, then we execute this job partly just as far as possible, i.e., during the T-time, and we put the remaining part of a job back into q.

On the algorithm's completion we'll choose the optimal solution (or, at least, one of several solutions). The running time of algorithm is $O(n \log n)$.

Implementation

The following function takes a vector of jobs (consisting of a deadline, a duration, and the job's index) and computes a vector containing all indices of the used jobs in the optimal schedule. Notice that you still need to sort these jobs by their deadline, if you want to write down the plan explicitly.

```
struct Job {
    int deadline, duration, idx;

    bool operator<(Job o) const {
        return deadline < o.deadline;
    }
};
```

```
vector<int> compute_schedule(vector<Job> jobs)
    sort(jobs.begin(), jobs.end());

    set<pair<int,int>> s;
    vector<int> schedule;
    for (int i = jobs.size()-1; i >= 0; i--) {
        int t = jobs[i].deadline - (i ? jobs[i
        s.insert(make_pair(jobs[i].duration, j
        while (t && !s.empty()) {
            auto it = s.begin();
            if (it->first <= t) {
                t -= it->first;
                schedule.push_back(it->second)
            } else {
                s.insert(make_pair(it->first -
                t = 0;
            }
            s.erase(it);
        }
    }
    return schedule;
}
```