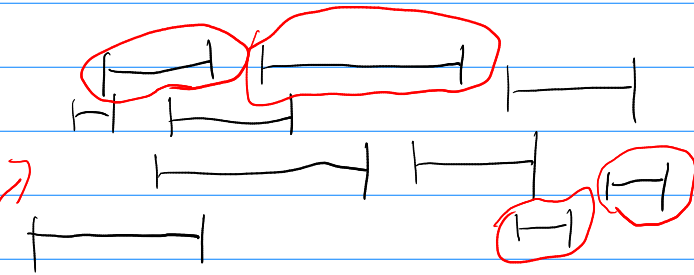


Interval Scheduling

Most basic version:

Given a set of n intervals:



Each defined by a start time s_i
and finish time f_i .

(think: people that want to rent your house).

What is the maximum number of
disjoint intervals you can pick?
That is, find

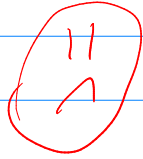
$S \subseteq [n]$ maximizing $|S|$
such that

$$[s_i, f_i) \cap [s_j, f_j) = \emptyset \quad \forall i, j \in S, i \neq j$$

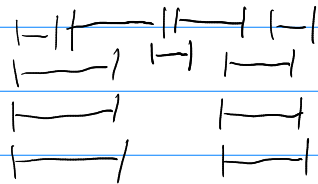
The example has 4.

Many plausible greedy algorithms.

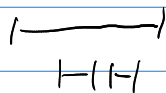
Shortest first?



Interval w/ fewest overlaps first?



First come first serve?



Earliest finish time?

YES

but needs a proof, to show it's different from the above fallacious arguments.

GreedySchedule (I): $\leftarrow I = \{(s_i, f_i)\}_{i \in [n]}$

Sort I by increasing f_i .

$S = \emptyset$

$t = -\infty$

For (s, f) in I:

if $s \leq t$: continue

$t = f$

$S.append((s, f))$

return S.

Theorem. GreedySchedule returns a valid set of intervals of maximum size.

Proof Because $s \leq f$ for every interval, t never decreases. Therefore t is always the largest f in S . This ensures each new interval added to S does not overlap with any interval already in S , so the returned S is valid.

We now prove by induction on n that the returned S has maximal size.

The base case of $n=0$ is trivial. Now consider some $n \geq 1$, and assume the theorem holds for all I' with $|I'| \leq n-1$.

For any time t , define $I_t = \{(s, f) \in I \mid s \geq t\}$.

Notice that, if (s_1, f_1) has the earliest finish time in I , then

$GreedySchedule(I) = [(s_1, f_1)] + GreedySchedule(I_{f_1})$.

Let S^* be a maximum disjoint subset of I_f which we can write out in order as

$$S^* = (s_1^*, f_1^*), (s_2^*, f_2^*), \dots, (s_k^*, f_k^*)$$

for $k^* = |S^*|$. Similarly, GreedySchedule returns an S with

$$S = (s_1, f_1), (s_2, f_2), \dots, (s_k, f_k)$$


for $k = |S|$. We claim $k = k^*$.

Since f_1 is the earliest possible finish time,
 $f_1 \leq f_1^* \leq s_2^*$.

Therefore $(s_2^*, f_2^*), \dots, (s_k^*, f_k^*)$ all lie in I_{f_1} . Since these are disjoint, the optimal solution for I_{f_1} has at least $k^* - 1$ intervals. By the inductive hypothesis, this means

$$|\text{GreedySchedule}(I_{f_1})| \geq k^* - 1$$

so $k = |\text{GreedySchedule}(I)| \geq k^*$.

Since k^* is the maximum possible, $k^* = k$ as desired. 

So: $O(n \log n)$ sort + $O(n)$ time.

Weighted interval scheduling

Each interval now has a weight w_i .

$$\text{Maximize } \sum_{i \in S} w_i$$

over sets of disjoint intervals.

Naive recursion:

Sched (I)

Let $i =$ first elt of I

Return min of

not chosen:

Sched (I \setminus i),

chosen:

Sched (I \setminus \{i \text{ or anything conflicting with } i\}) + w_i

Problem: 2^n possible inputs.

Solution: If I sorted by f_i , then

only $n+1$ inputs ever happen:

(suffix of I sorted by s_i)

\Rightarrow memoized time is $n \cdot (\text{time per input})$

n^2 naively

n more carefully.

Sched (index in I, f of last chosen)