

The job-interval scheduling problem (JISP) is defined below.

**Instance:**  $(s_1, f_1, \mu_1), \dots, (s_n, f_n, \mu_n)$ , where  $s_i, f_i, \mu_i \in \mathbb{N}$  for  $1 \leq i \leq n$

**Solution:** A subset  $I \subseteq \{1, \dots, n\}$  such that

1.  $[s_i, f_i) \cap [s_j, f_j) = \emptyset$  for all  $i \neq j \in I$
2.  $\mu_i \neq \mu_j$  for all  $i \neq j \in I$

**Objective:** Maximize  $|I|$

Intuitively, all the intervals with the same  $\mu$ -value represent possible times when the same job could be scheduled. The goal is to schedule as many jobs as possible.

Below is an algorithm called EARLIEST-FINISH-TIME (EFT) for JISP.

```

1: sort intervals so that  $f_1 \leq f_2 \leq \dots \leq f_n$ 
2:  $S \leftarrow \emptyset$ 
3:  $t \leftarrow 0$ 
4: for  $i \leftarrow 1, \dots, n$  do
5:   if  $\mu_i \neq \mu_j$  for all  $j \in S$  and  $s_i \geq t$  then
6:      $S \leftarrow S \cup \{i\}$ 
7:      $t \leftarrow f_i$ 
8:   end if
9: end for

```

1. Give an example of an input where EFT achieves an approximation ratio of exactly  $1/2$ .

**Solution:** The instance consists of three intervals:  $(0, 1, 0), (0, 2, 1), (2, 3, 0)$ . The optimal solution takes  $(0, 2, 1)$  and  $(2, 3, 0)$ . The EFT algorithm takes  $(0, 1, 0)$  (because it has the earliest finishing time), and is unable to take either of the remaining jobs:  $(0, 2, 1)$  overlaps, and  $(2, 3, 0)$  belongs to the same job class. So in this case  $|I| = |\text{OPT}|/2$ , where OPT is the optimal solution and  $I$  is the solution returned by EFT.

2. Show that EFT is a  $1/2$ -approximation algorithm for JISP.

**Solution:** The idea is to use a “charging scheme”: let  $I$  be the solution returned by EFT, and let OPT be an optimal solution, and for each interval  $i \in \text{OPT}$ , “charge” a cost of 1 to an interval in  $I$ . If we can do this in such a way that no interval in  $I$  gets charged more than 2, it will follow that  $|\text{OPT}| \leq 2|I|$  (because  $|\text{OPT}|$  is equal to the sum of the charges to each element in  $I$ ).

So let  $i \in \text{OPT}$ , and we’ll decide where to charge it. If  $i \in I$  then assign a charge of 1 to itself. If  $i \notin I$ , then there is either an interval  $j \in I$  such that

$[s_i, f_i) \cap [s_j, f_j) \neq \emptyset$ , or there is an interval  $k \in I$  such that  $\mu_k = \mu_i$ . In the first case, charge a cost of 1 to the earliest-finishing job in  $I$  that overlaps  $[s_i, f_i)$ . Otherwise, charge a cost of 1 to the (only) interval in  $I$  that belongs to the same job class  $\mu_i$ .

Now consider how much a job  $j \in I$  got charged. The charges to  $j$  come in two ways: from intervals in OPT that overlap it, and from intervals in OPT having the same job class. Clearly there is at most one interval in OPT of the second type (since OPT is feasible, it only contains one job with class  $\mu_j$ ). Now if  $j$  gets charged from an interval  $i \in \text{OPT}$  that overlaps it, then  $j$  is the earliest-finishing job that overlaps  $i$  (because of our charging scheme). In particular,  $f_j \leq f_i$ : because if  $f_i < f_j$ , then our algorithm considers  $i$  before  $j$ , and since it didn't take interval  $i$  we conclude there must be an interval  $k \in I$  with  $f_k \leq f_i < f_j$  that overlaps interval  $i$ , so  $j$  is not the earliest-finishing interval in  $I$  that overlaps  $i$ . So if an interval  $i \in \text{OPT}$  is charged to  $j \in I$  in this way, then  $f_j \in [s_i, f_i)$ . There cannot be two such jobs in OPT, as they would overlap each other! (at the point  $f_j$ ).

Thus for every  $i \in I$ , the charge to  $i$  is at most two: at most one from intervals in the same job class, and at most one from intervals  $i \in \text{OPT}$  such that  $j$  is the earliest-finishing job in  $I$  that overlaps  $i$ .