Санкт-Петербургский национальный исследовательский университет
информационных технологий, механики и оптики

Кафедра информатики и прикладной математики

Дисциплина «Алгоритмы и структуры данных»
Лабораторная работа №5

Вариант 12

Выполнил:
Съестов Дмитрий Вячеславович
Группа P3217

Преподаватель:
Зинчик Александр Адольфович

Санкт-Петербург
2018

Задание

В картинной галерее каждый сторож работает в течение некоторого непрерывного отрезка времени. Расписанием стражи называется множество пар [T1(i), T2(i)] - моментов начала и конца дежурства i-го сторожа из интервала [0,EndTime].

Для заданного расписания стражи требуется:

(а) проверить, в любой ли момент в галерее находится не менее двух сторожей.

Если условие (а) не выполнено, то:

(б) перечислить все интервалы времени с недостаточной охраной (менее 2 сторожей).

(в) добавить наименьшее число сторожей с заданной, одинаковой для всех длительностью дежурства, чтобы получить правильное расписание (т.е. удовлетворяющее условию (а)).

(г) проверить, можно ли обойтись без добавления новых сторожей, если разрешается сдвигать времена дежурства каждого из сторожей с сохранением длительности дежурства.

(д) Если это возможно, то составить расписание с наименьшим числом сдвигов.

Типы данных

```
-- | Момент времени
newtype Time = Time { minutes :: Int } deriving (Eq, Ord)

-- | Временной интервал
data Interval = Interval { start :: Time, end :: Time }

-- | Охранники
data Guard = Guard { id :: Int, watch :: Interval } deriving Show
type Schedule = [Guard]

-- | Входные данные
data InputData = InputData { endTime :: Time, schedule :: Schedule, watchLength :: Time }
deriving Show

-- | Сдвиг дежурства
data WatchShift = WatchShift { guardId :: Int, watchShift :: Time }
type ScheduleChange = [WatchShift]
```

Решение

```
-- | Возвращает интервалы времени из расписания охраны.
getIntervals :: Time -> Schedule -> [Interval]
getIntervals end = join . sort . nub . addBorders . concatPairs . map getTime
    where getTime (Guard _ (Interval a b)) = (a, b)
          concatPairs = uncurry (++) . unzip
          addBorders xs = let start = Time 0 in start:end:xs
          join xs = zipWith Interval xs (tail xs)
```

```
-- | Для каждого интервала находит количество охранников.
intervalsByGuardCount :: Time -> Schedule -> [(Interval, Int)]
intervalsByGuardCount end [] = error "Не задан график дежурства охранников"
intervalsByGuardCount end gs = map guardCount $ getIntervals end gs
    where guardCount i = let guards = filter (\(Guard _ watch) -> watch `includes` i) gs
                         in (i, length guards)

-- | Находит интервалы времени с недостаточной охраной.
weakGuardIntervals :: Time -> Schedule -> [(Interval, Int)]
weakGuardIntervals = filter (\(_, n) -> n < 2) .@. intervalsByGuardCount

-- | Находит интервалы времени с хорошей охраной.
strongGuardIntervals :: Time -> Schedule -> [(Interval, Int)]
strongGuardIntervals = filter (\(_, n) -> n > 2) .@. intervalsByGuardCount

-- | Вычисляет, сколько дополнительных охранников понадобится для заполнения малоохраняемых
интервалов.
countAdditionalGuards :: Time -> Time -> [(Interval, Int)] -> (Int, [Interval])
countAdditionalGuards _ _ []  = error "Не заданы малоохраняемые интервалы"
countAdditionalGuards endTime watchTime ints = count watchTime ints []
    where count _ [] guards = (length guards, guards)
          count t ints guards
              | null covered = let t' = minimum $ map (start . fst) ints
                               in count (t' +: watchTime) ints guards
              | otherwise = let t' = t +: watchTime
                                n  = 2 - (minimum covered)
                                endt = min t endTime
                                interval = Interval (endt -: watchTime) endt
                                guards' = guards ++ replicate n interval
                                ints' = dropWhile (\i -> end (fst i) <= t) ints
                            in count t' ints' guards'
              where covered = map snd $ takeWhile (\i -> start (fst i) < t) ints

-- | Меняет график таким образом, чтобы малоохраняемых интервалов не оставалось (если это
возможно)
reorganizeSchedule :: Time -> Schedule -> Maybe Schedule
reorganizeSchedule endTime gs
    | manHours < endTime +: endTime = Nothing
    | otherwise =
        case find halfSchedule $ subsets gs of
            Nothing    -> Nothing
            Just first ->
                let second = gs \\ first
                in Just $ map adjustWatch $ align (Time 0) first ++ align (Time 0) second
    where watchSum = foldl (+:) (Time 0) . map (duration . watch)
          manHours = watchSum gs
```

```
halfSchedule gs = watchSum gs `between` (endTime, manHours -: endTime)
align _ [] = []
align start (g:gs) =
  let start' = start +: (duration $ watch g)
  in shift start g : align start' gs
adjustWatch g@(Guard _ watch)
  | end watch > endTime = shiftEnd endTime g
  | otherwise = g
```