Reading Quiz #4

ⓘ This is a preview of the published version of the quiz

Started: Feb 1 at 11:43am

Quiz Instructions

Read section 4.1, and one of sections 4.2, 4.3 of your textbook.

You should answer questions 1 to 4, and then either questions 5 and 6 (if you read section 4.2) or questions 7 and 8 (if you read section 4.3). Canvas will compute a grade based on all eight questions, but we will later readjust the grade to only take into account either questions 5 and 6, or questions 7 and 8 (but not more than one of these pairs).

Question 1 — 1 pts

Section 4.1: Number of solutions

Among the following values, which is the largest with the property that it can be the number of optimal solutions of an interval scheduling problem with *n* intervals? (Note: this is not necessarily the largest possible, just the largest among these values.)

Hint: some of these answers can be ruled out fairly quickly through an example. For others, think about what a problem instance with that number of solutions would look like.

- ○ 2^n
- ○ $2^{n/2}$
- ○ 1
- ○ n
- ○ n^2

Question 2 — 1 pts

Section 4.1: Mix and match

In an instance of the "Interval Scheduling" problem, assume that $i_1, \ldots, i_k$ is the list of jobs in A (the output of our greedy algorithm), and $j_1, \ldots, j_k$ is another optimal solution. Consider these new sequences of jobs:

$$S_1 = i_1, \ldots, i_l, j_{l+1}, \ldots, j_k \ (for\ some\ l\ s.t.\ 1 < l < k)$$
$$S_2 = j_1, \ldots, j_l, i_{l+1}, \ldots, i_k \ (for\ some\ l\ s.t.\ 1 < l < k)$$

Which one(s) are always **valid** (non-overlapping) solutions?

- ○ S1 and S2 are both valid
- ○ S1 is always valid, but S2 can sometimes be invalid
- ○ S1 and S2 can both be invalid sometimes
- ○ S2 is always valid, but S1 can sometimes be invalid

Question 3

1 pts

Section 4.1: Interval partitioning via multiple interval scheduling

Here is an alternative algorithm aimed at solving the interval partitioning problem:

```
    num = 0
    while R is not empty
        A = the optimal sol. returned by the greedy ISP algorithm o
f page 118 (sec 4.1)
        assign num as label to every interval in A
        remove every interval in A from R
        increment num
```

How frequently will this algorithm use the smallest possible number of labels?

Hint: look at a few examples. This should allow you to narrow down the possible answers.

- Never
- Always
- Sometimes, but not always

Question 4

1 pts

Section 4.1: Depth

Assume you want to write a program to find the **depth** for instances of the "Interval Partitioning" problem with the following algorithm:

1. For all the points $t \in T$ in the time-line, compute the number of intervals that pass over these specific points in time.

2. Report the maximum as the depth.

Specify which set(s) of points can be used as the set T to find the correct depth.

- [ ] The set of the finishing times of all the intervals
- [ ] The set of midpoints of the intervals
- [ ] The set of the starting times of all the intervals

Question 5

1 pts

In the following list, count the number of inversions. (The numbers are the deadlines of each job)

$$7,\ 3,\ 4,\ 10,\ 8,\ 2$$

Question 6

1 pts

Assume you already have a (not necessarily optimal) solution to an instance of the "Scheduling to minimize the (maximum) lateness" problem. In this solution the jobs A and B are scheduled next to each other and both are past their deadlines. Now, what happens if you swap the order of A and B in your solution?

- ○ Lateness might increase
- ○ Lateness might increase or stay the same
- ○ Lateness can either increase, decrease, or stay the same
- ○ Lateness might decrease or stay the same
- ○ Lateness always decreases

Question 7

1 pts

Which of the following statements about the proof of Theorem 4.12 are true?

- [ ] Schedule S' may not be an optimal schedule.
- [ ] Schedules S and S' make the same eviction decisions on the first j+1 items.
- [ ] Even if schedules S and S' have different cache contents after j+1 steps, their cache contents will once again be the same after j+2 steps.
- [ ] Some of the steps of the construction of S' may result in a non-reduced schedule.
- [ ] If S and SFF evict items f and e respectively, with e≠ f, in step j+1, then the schedule S' will have the cache content as S by the time there is a request for e.

Question 8

1 pts

Suppose there is a schedule *S* that is not reduced, and brings in *n* items into the cache over the execution of an algorithm. The reduced schedule obtained by only bringing items into the cache the next time they are requested will:

- ○ Bring in more or the same number of items as S.
- ○ Bring in fewer items than S.
- ○ Bring in more items than S.
- ○ Bring in the same number of items as S.
- ○ Bring in fewer or the same number of items as S.

Not saved Submit Quiz