CS300 – Spring 2018-2019 - Sabancı University

Homework #4 – Job Scheduling

Due Date: May 15 Wednesday 22:00

Brief Description

You are the boss of a painting company. You take jobs to paint houses, offices, schools etc. You are a very greedy person and all you think is to make as much profit as possible. Thus, you want to take full advantage of all your workers.

A customer approaches you having *M* places waiting to be painted. In order to finish all jobs in a timely manner, you need to assign all the jobs to your workers by the rules listed below. (You may own the company, but we, as the instructors, are still the big bosses).

Rules

- There are *N* workers (painters) at your company. All of them work alone, i.e., not more than 1 worker is assigned to any single job.
- No worker will remain without a painting job if there are still waiting jobs to be completed.
- Every worker and every job will have an unique ID, e.g., *workerID* and *jobID*. Worker IDs will be assigned by you starting from 1, i.e., the first worker will have and ID of 1, the second worker with ID of 2, and so on.
- If there are more than one worker waiting jobs to be assigned (2 workers may finish their jobs at the same time), first job in the queue will be assigned to the worker having lower ID.
- The deadlines are given as number of days. The workers can always finish the assigned job at the exact deadline, not earlier than its deadline.

Approach / Algorithm

Given *N* workers and *M* jobs:

1. Read the input data and store it in a vector (or linked list, array etc.).
2. Take the first *N* jobs and assign them to the workers in order with *workerID*.
3. Store the workers with the assigned jobs in a heap.
4. Repeat
   - 4.1. Get the worker whose currently assigned job (say *jobID*=5) with the minimum number of days deadline, say 2 days. Make sure that there is no other job having 2 days of deadline for any worker at the moment.
   - 4.2. While deleting this worker from the heap, this means that the job with *jobID*=5 has been completed. Thus, 2 days have been passed for all the jobs. Hence, you need to decrease 2 days from all jobs assigned to the workers in the heap.
   - 4.3. Assign a new job from the job queue for this worker.

NOTE THAT

1. There may be more than one worker whose jobs having the same remaining days to be completed in the heap. Normally, if there are workers with jobs of equal deadline days in the heap, the smaller *workerID* has to be placed on the top of heap. However, for this homework, while implementing your project, you can either change the overloaded comparison function, or you can simply check the next minimum element in the heap until you find a different minimum element. Then, you can sort with respect to their *workerID*'s.
2. You can perhaps solve this problem using some other more efficient methods. However, what we want from you is NOT to come up with an optimized scheduling solution, instead we want you to generate this scheduling as fast as possible.

Input

You will be given an input file as follows:

```
3          // There are 3 painters in your company
5          // There are 5 jobs for painting
1 3        // First job's (jobID=1) deadline is 3 days, i.e., it will be completed in 3 days
2 5        // Second job's (jobID=2) deadline of 5 days, i.e., it will be completed in 5 days
3 6        // Third job's (jobID=3) deadline of 6 days, i.e., it will be completed in 6 days
4 4        // Fourth job's (jobID=4) deadline of 4 days, i.e., it will be completed in 4 days
5 4        // Fifth job's (jobID=5) deadline of 4 days, i.e., it will be completed in 4 days
```

Output

Outputs will be written in the exact format as given below into an output file. Your output file name will have an extension of ".out" appended to the provided input file name. For example, if the input file name is "abc.txt", then output file name will be "abc.txt.out".

Here is the output of sample input given above:

```
Job-1->Worker-1 (Day: 0)        // Job-1 is assigned to Worker-1 on Day 0
Job-2->Worker-2 (Day: 0)
Job-3->Worker-3 (Day: 0)
Job-4->Worker-1 (Day: 3)
Job-5->Worker-2 (Day: 5)
All jobs are completed in 9 days.
```

What Is Expected from You

1. Prompt a message to the user to enter the input file name. Read all the jobs into a queue. You need to store each job in a struct with the below overloaded operators:

```
struct Job
{
    ...

    bool operator<(const Job & j)  { }  // TODO
    bool operator>(const Job & j)  { }  // TODO
};
```

2. Create another struct for the workers with the below overloaded operators:

```
struct Worker
{
    ...

    bool operator<(const Worker & w);  // TODO
    bool operator>(const Worker & w);  // TODO
    void operator-(const int & d);     // TODO

};
```

3. You need to use a templated Heap class for this homework. You can get it from your slides, lecture book or internet.

4. You will need a *getMin()* function which basically returns the minimum element of the heap. Note that in your lecture notes, there is no *getMin()* function, instead you have *deleteMin()*. For further information, you may check your lecture notes.

5. You will need a *decreaseElements(int t)* member function for your *Heap* class. This function will be used when the minimum item is deleted from heap. (**Hint:** When a job with a deadline of *t* days, has been completed, these *t* days must be subtracted from all the jobs in the heap.)

    a. It will decrease the days of all elements in the heap by *t*.

    b. It **must use the** subtraction operator of the **`Worker`** struct as given in 2**.**

Assumptions

1. The input file is valid, thus, you do not need to check the validity of the input file.

2. Both for the job deadlines and the number of workers will be nonzero positive values.

Important Notes

1. The given sample input file will **NOT** be used while testing your code. Therefore, even though you can run all the sample runs as expected, it does not guarantee that you will get a full grade. You need to try to come up with your own test cases as well.
2. Exactly follow the same output structure as we ask for. Since, we will grade this homework automatically, if you make even a small change, you may get lower grades.

An advice for all of your homework assignments

We happen to see a lot that most of you change the templated class implementations whenever you need it. However, this should not be the case. In real life, you will generally have modules for your needs. By importing these modules into your project, you will have to adopt your code to those modules, not other way around. If you change the imported module, then every project which uses the corresponding modules have to adopt their implementations with respect to your modification.

Thus, if you ever happen to feel that you need to make some modifications on a templated class implementation (unless it is specifically asked for), most probably, your design has some flaws. Reconsider your design, and try to come up with a better implementation without altering the templated class. The most common techniques for that is to use overloading operators and/or functions.

Note that this advice will not be considered while grading your homework, this is only for your benefit.

Sample Runs

For the sake of simplicity, we did not include the console messages here as they are not important. You can print any messages on the console. Just be careful about the name of the input and output files as described in the document.

Sample Run 1

Input File:

```
1
3
1 1
2 2
3 3
```

Output File:

```
Job-1->Worker-1 (Day: 0)
Job-2->Worker-1 (Day: 1)
Job-3->Worker-1 (Day: 3)
All jobs are completed in 6 days.
```

Sample Run 2

Input File:

```
4
3
1 2
2 5
3 3
```

Output File:

```
Job-1->Worker-1 (Day: 0)
Job-2->Worker-2 (Day: 0)
Job-3->Worker-3 (Day: 0)
All jobs are completed in 5 days.
```

Sample Run 3

Input File:

```
3
7
1 3
```

```
2 3
3 4
4 3
5 4
6 2
7 5
```

Output File:

```
Job-1->Worker-1 (Day: 0)
Job-2->Worker-2 (Day: 0)
Job-3->Worker-3 (Day: 0)
Job-4->Worker-1 (Day: 3)
Job-5->Worker-2 (Day: 3)
Job-6->Worker-3 (Day: 4)
Job-7->Worker-1 (Day: 6)
All jobs are completed in 11 days.
```

General Rules and Guidelines about Homeworks

The following rules and guidelines will be applicable to all homeworks, unless otherwise noted.

How to get help?

You may ask questions to TAs (Teaching Assistants) of CS300. Office hours of TAs are at the syllabus. Recitations will partially be dedicated to clarify the issues related to homework, so it is to your benefit to attend recitations.

What and Where to Submit

Please see the detailed instructions below/in the next page. The submission steps will get natural/easy for later homeworks.

Grading and Objections

Careful about the semi-automatic grading: Your programs will be graded using a semi-automated system. Therefore, you should follow the guidelines about input and output order; moreover, you should also use same prompts as given in the Sample Runs. Otherwise semi-automated grading process will fail for your homework, and you may get a zero, or in the best scenario you will lose points.

Grading:

- Late penalty is 10% off the full grade and only one late day is allowed.
- **Having a correct program is necessary, but not sufficient to get the full grade. Comments, indentation, meaningful and understandable identifier names, informative introduction and prompts, and especially proper use of required functions, unnecessarily long program (which is bad) and unnecessary code duplications will also affect your grade.**
- Please submit your own work only (even if it is not working). It is really easy to find out "similar" programs!
- For detailed rules and course policy on plagiarism, please check out http://myweb.sabanciuniv.edu/gulsend/courses/cs201/plagiarism/

Plagiarism will not be tolerated!

Grade announcements: Grades will be posted in SUCourse, and you will get an Announcement at the same time. You will find the grading policy and test cases in that announcement.

Grade objections: It is your right to object to your grade if you think there is a problem, but before making an objection please try the steps below and if you still think there is a problem, contact the TA that graded your homework from the email address provided in the comment section of your announced homework grade or attend the specified objection hour in your grade announcement.

- Check the comment section in the homework tab to see the problem with your homework.
- Download the .zip file you submitted to SUCourse and try to compile it.
- Check the test cases in the announcement and try them with your code.
- Compare your results with the given results in the announcement.

What and where to submit (IMPORTANT)

Submissions guidelines are below. Most parts of the grading process are automatic. Students are expected to strictly follow these guidelines in order to have a smooth grading process. If you do not follow these guidelines, depending on the severity of the problem created during the grading process, 5 or more penalty points are to be deducted from the grade.

Add your name to the program: It is a good practice to write your name and last name somewhere in the beginning program (as a comment line of course).

Name your submission file:

- Use only English alphabet letters, digits or underscore in the file names. Do not use blank, Turkish characters or any other special symbols or characters.
- Name your cpp file that contains your program as follows.

  "**SUCourseUserName_yourLastname_yourName_HWnumber.cpp**"
- Your SUCourse user name is actually your SUNet user name which is used for checking sabanciuniv e-mails. Do NOT use any spaces, non-ASCII and Turkish characters in the file name. For example, if your SUCourse user name is cago, name is Çağlayan, and last name is Özbugsızkodyazaroğlu, then the file name must be:

  cago_ozbugsizkodyazaroglu_caglayan_hw4.cpp
- Do not add any other character or phrase to the file name.
- Make sure that this file is the latest version of your homework program.
- You need to submit ALL .cpp and .h files including the robot and minifw files in addition to your main.cpp in your VS solution. Compress all your necessary files using WINZIP or WINRAR programs. Please use "**zip**" compression. "rar" or another compression mechanism is NOT allowed. Our homework processing system works only with zip files. Therefore, make sure that the resulting compressed file has a zip extension.
- Check that your compressed file opens up correctly and it contains your **cpp** file. You will receive no credits if your zip file does not expand or it does not contain the correct file.
- The naming convention of the zip file is the same as the cpp file (except the extension of the file of course). The name of the zip file should be as follows.

  "**SUCourseUserName_yourLastname_yourName_HWnumber.zip**"

  For example zubzipler_Zipleroglu_Zubeyir_hw4.zip is a valid name, but hw4_hoz_HasanOz.zip, HasanOzHoz.zip are NOT valid names.

Submission:

- Submit via SUCourse ONLY! You will receive no credits if you submit by other means (e-mail, paper, etc.).
  1. Click on "Assignments" at CS300 SUCourse.
  2. Click Homework 4 in the assignments list.
  3. Click on "Add Attachments" button.
  4. Click on "Browse" button and select the zip file that you generated.
  5. Now, you have to see your zip file in the "Items to attach" list.
  6. Click on "Continue" button.
  7. Click on "Submit" button. We cannot see your homework if you do not perform this step even if you upload your file.

Resubmission:

- After submission, you will be able to take your homework back and resubmit. In order to resubmit, follow the following steps.
  1. Click on "Assignments" at CS300 SUCourse.
  2. Click Homework 4 in the assignments list.
  3. Click on "Re-submit" button.
  4. Click on "Add/remove Attachments" button
  5. Remove the existing zip file by clicking on "remove" link. This step is very important. If you don't delete the old zip file, we get both files and the old one may be graded.
  6. Click on "Browse" button and select the new zip file that you want to resubmit.

7. Now, you have to see your new zip file in the "Items to attach" list.
8. Click on "Continue" button.
9. Click on "Submit" button. We cannot see your homework if you do not perform this step even if you upload your file.

Successful submission is one of the requirements of the homework. If, for some reason, you cannot successfully submit your homework and we cannot grade it, your grade will be 0.

Good Luck!
Hanefi Mercan and Gülşen Demiröz