The task-scheduling problem

June 5, 2018

Student: Tolgoi Dragos

Computers and Information Technology
(Engligh Language)

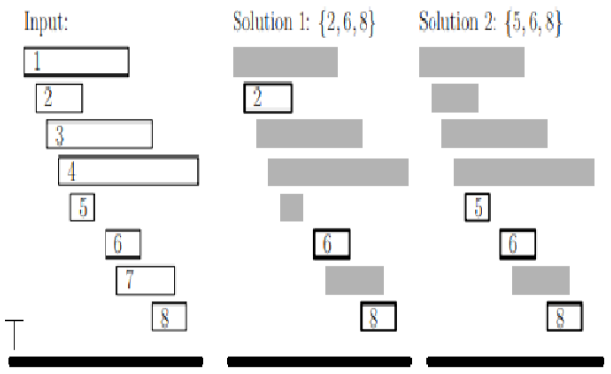
Group 1.3 B

Introduction

The problem can be solved using 3 sorting methods:
1.Earliest Activity First: Repeatedly select the activity with the earliest start time, provided that it does not overlap any of the previously scheduled activities. Shortest Activity First: Repeatedly select the activity with the smallest duration (fisi), provided that it does not conflict with any previously scheduled activities.
2.Earliest Activity First: Repeatedly select the activity with the earliest start time, provided that it does not overlap any of the previously scheduled activities. Shortest Activity First: Repeatedly select the activity with the smallest duration (fisi), provided that it does not conflict with any previously scheduled activities.
3.Lowest Conflict Activity First: Repeatedly select the activity that conflicts with the smallest number of remaining activities, provided that it does not conflict with of the previously scheduled activities. (Note that once an activity is selected, all the conflicting activities can be effectively deleted, and this affects the conflict counts for the remaining activities.)

Problem statement

Schedule several competing activities that require exclusive use of a common resource, with a goal of selecting a maximum-size set of mutually compatible activities. Given that an activity has a start time and a finish time two activities are compatible if the intervals between theirs start and finish times don not overlap. Implement two different algorithms that resolve this problem.
An example of an input and two possible solutions is given in the representation below.



Pseudocode

In the first function we will sort our task from the lowest duration to the higest duration using the bubblesort algorithm.In the second function we will check if the end time of a task overlaps the start time of the next task.
We will use functions which will be called by the main program.Here are the functions usedby the program:

```
void bubbleSort( task arr[], int n)

     for i <- 0 to n-1 do
         for j <- 0 to n-i-1 do
              if arr[j].duration > arr[j+1].duration
                 taskAux.start <- arr[j].start
                 taskAux.finish <- arr[j].finish
                 taskAux.duration <- arr[j].duration

                 arr[j].start <- arr[j+1].start
                 arr[j].finish <- arr[j+1].finish
                 arr[j].duration <- arr[j+1].duration

                 arr[j+1].start= taskAux.start
                 arr[j+1].finish= taskAux.finish
                 arr[j+1].duration= taskAux.duration

             else if arr[j].duration = arr[j+1].duration

                 if arr[j].start > arr[j+1].start

                     taskAux.start <- arr[j].start
                     taskAux.finish <- arr[j].finish
                     taskAux.duration <- arr[j].duration

                     arr[j].start <- arr[j+1].start;
                     arr[j].finish <- arr[j+1].finish
                     arr[j].duration <- arr[j+1].duration

                     arr[j+1].start <- taskAux.start
                     arr[j+1].finish <- taskAux.finish
                     arr[j+1].duration <- taskAux.duration
```

```
int sortare( task taskuri[],int n)

  aux <- taskuri[0].finish

  for i <- 0 to n-2 do
    for j <- i+1 to n-1 do

        if aux <= taskuri[j].start

            k <- k+taskuri[j].duration;
            aux <- taskuri[j].finish;

            i++;



  return k;
```

Application design

Source Code

```c
//----------------------------functions.h------------------------------

#ifndef MAIN_H_INCLUDED
#define MAIN_H_INCLUDED

#include <stdio.h>
#include <stdlib.h>

typedef struct{
    int start;
    int finish;
    int duration;
}task;

void bubbleSort( task arr[], int n);
int sortare(task taskuri[],int n);

#endif
```

```c
//----------------------------functions.c------------------------------

#include "function.h"
int aux=0;
task taskuri[20],taskAux;
void bubbleSort( task arr[], int n)
{

    int i, j;
    for (i = 0; i < n-1; i++)
        for (j = 0; j < n-i-1; j++)
            if (arr[j].duration > arr[j+1].duration)
            {
                taskAux.start = arr[j].start;
                taskAux.finish=arr[j].finish;
                taskAux.duration=arr[j].duration;

                arr[j].start = arr[j+1].start;
                arr[j].finish = arr[j+1].finish;
                arr[j].duration = arr[j+1].duration;

                arr[j+1].start= taskAux.start;
                arr[j+1].finish= taskAux.finish;
                arr[j+1].duration= taskAux.duration;
            }
            else if(arr[j].duration == arr[j+1].duration)
            {
                if(arr[j].start > arr[j+1].start)
                {
                    taskAux.start = arr[j].start;
                    taskAux.finish=arr[j].finish;
                    taskAux.duration=arr[j].duration;

                    arr[j].start = arr[j+1].start;
                    arr[j].finish = arr[j+1].finish;
                    arr[j].duration = arr[j+1].duration;

                    arr[j+1].start= taskAux.start;
                    arr[j+1].finish= taskAux.finish;
                    arr[j+1].duration= taskAux.duration;

                }

            }


}

int sortare( task taskuri[],int n)

{
    int i;
    int j;
    int k;

    aux=taskuri[0].finish;
    for(i=0; i<=n-2; i++)
    {
        for(j=i+1; j<=n-1; j++)
        {
            if(aux<=taskuri[j].start)
            {
                k=k+taskuri[j].duration;

                aux=taskuri[j].finish;
                i++;
            }
        }

    }

    return k;


}
```

```
//--------------------------------main.c-------------------------------


#include "function.h"

int k=1;
int i;
int j;
int number_of_tasks;
int p;
task taskuri[20];

int main()
{

    FILE *myFile;
    myFile = fopen("in.txt", "r");
    int numberArray[1000];
    int i,j;
    int n;
    int t;
    printf("time=");
    scanf("%d",&t);

    if (myFile == NULL)
    {
        printf("Error Reading File\n");
        exit (0);
    }

    fscanf(myFile, "%d,", &numberArray[0]);
    n=numberArray[0];
    j=0;

    for (i = 1; i <= n*2; i++)
    {
        fscanf(myFile, "%d,", &numberArray[i] );
        if(i%2!=0)
        {
            taskuri[j].start=numberArray[i];
        }
        if(i%2==0)
        {
            taskuri[j].finish=numberArray[i];
            j++;
        }
    }

    for (i = 0; i < n; i++)
    {
        taskuri[i].duration=taskuri[i].finish-taskuri[i].start;
    }

    bubbleSort(taskuri,n);

    for (i = 0; i < n; i++)
    {
        printf("task_%d: start_%d, finish_%d , duration_%d\n\n",
             i,taskuri[i].start,taskuri[i].finish,taskuri[i].duration );
    }

    k=sortare(taskuri,n);
    printf("The maximum duration is:");
    printf("k=%d",k);
    fclose(myFile);

    return 0;
}
```

Experiments and results

```
Number of tasks is: 8
The first row represents the total number of tasks.
The first column represents the start time of a task.
The second column represents the end time of a task.
8
5 10
1 2
1 3
2 4
4 7
6 7
2 5
9 22



The duration of each task is calculated by substracting the end time of
    the task form the start time of the task
(the duration of each task is representated in the third column)
5 10 5
1 2  1
1 3  2
2 4  2
4 7  3
6 7  1
2 5  3
9 22 13

The algorithm will start ordering all the tasks by the duration of each
    task.

1 2  1
6 7  1
1 3  2
2 4  2
4 7  3    *
2 5  3    *
5 10 5
9 22 13
```

```
Now, the algorithm will start ordering each task keeping in mind the
    duration and the start time of a task

1 2  1
6 7  1
1 3  2
2 4  2
2 5  3   *
4 7  3   *
5 10 5
9 22 13

Finally the algorithm will print the maximum duration(k):
1 2 1
6 7 1
9 22 13

k=1+1+13=15
```

References

1)http://www.cs.umd.edu/class/fall2017/cmsc451-0101/Lects/lect07-greedy-sched.pdf

2)https://www.sharelatex.com

3)https:https://stackoverflow.com/questions

4)https://www.geeksforgeeks.org