**Algorithms**

Programming Assignment #2

Ambitious DVD Dealer

**Introduction:**

Jimmy just inherited the rights to *N* (1 ≤ *N* ≤ 100,000) songs recorded by the popular group Raucous Rockers. Jimmy plan to release a set of *M* (1 ≤ *M* ≤ 1,000) compact disks with a selection of these songs. Each disk can hold a maximum capacity of *T* (1 ≤ *T* ≤ 50) minutes, and a song cannot cross from one disk to another.

According to Jimmy’s personal preference, the songs on the set of disks must appear in the order of the dates that they were written. For example, you can put song #1, #2, #3 in one disk, and put song #4, #6, #8 in another disk. But, you cannot put song #1, #6, #8 in the first disk, and put song #2, #3, #4 in the second disk. In the other words, every song, if recorded, must appear in the same order as the date of written.

Your work is to help Jimmy put as many songs as possible in those disks.

**Input/output Files:**

In the input file (\*.in), the first line shows the number of the songs (*N*), the number of disks (*M*), and the number of the maximum capacity of each disk (*T*). For example, there are four songs (*N*=4), two disks (M=2), and 5 minutes maximum capacity of each disk (*T*=5).

The second line shows the time of every song which is in the order of dates written. The first song is 4 minutes, and the second song is 3 minutes and *etc*.

Please note that, if a song is longer than *T*, it must not be selected.

|  |
| --- |
| 4 2 5  4 3 4 2 |

In the output file (\*.out), the first line shows the maximum songs you can put in those disks. The second line shows the songs we choose, but you don’t need to show this line in the .out file. The answers are the first song, the second song, and the fourth song.

|  |
| --- |
| 3  1 2 4 |

**Jobs:**

1. **Greedy:** Write a greedy algorithm to implement this DVD dealer optimization tool. Since there are so many greedy method, for convenience, all you need to do is choose songs by their time. Please analyze the complexity in your report.
2. **Mathematical Analysis:** Write a recursive function for this problem. Prove that this problem has *optimal substructure* so that it can be solved by dynamic programming.

|  |
| --- |
| **Hint:**  Suppose *F[i][j][k]* means the number of maximum songs, up to the first *i*th song, use *j* DVDs, and the last DVD up to the first *k* minutes. So *F[N][M][T]* is our final answer. *s[i]* means the time of the *i*th song. Please fill in the following formula:  Please also show the following two boundary conditions in mathematical equations:  1) If a song is longer than *T*, it cannot be selected. *F*[*i*][*j*][*k-s*[*i*]] must be specified.  2) At the beginning of the first DVD, when *j*=1, *k-s*[*i*] <0, then *F*[*i*][*j*][*k-s*[*i*]] must be specified. |

1. **Complexity:** Please analyze the time complexity of Job 3 in your report.
2. **Dynamic Programming:** Use dynamic programming to solve this problem.

**Command line parameters:**

In the command line, you are required to follow this format

./DVD\_Dealer –[GD|DP] <input\_file\_name> <output\_file\_name>

where GD and DP represent the greedy algorithm, dynamic programming, respectively. For example, the following command invoke the tool and run the greedy algorithm.

./bin/DVD\_Dealer –GD ./inputs/case1.in ./outputs/case1.out

**Requirements:**

1. Your source code must be written in C or C++. The code must be executable on EDA union lab machines.

2. In your report, please fill in the following table and also plot a figure showing the memory and running time. Please use –O2 optimization and turn off all debugging message.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| case | GD | | | DP | | |
|  | Maximum songs | CPU time(s) | Memory(KB) | Maximum songs | CPU time(s) | Memory(KB) |
| case1  N= |  |  |  |  |  |  |
| case2  N= |  |  |  |  |  |  |
| case3  N= |  |  |  |  |  |  |
| case4  N= |  |  |  |  |  |  |
| case5  N= |  |  |  |  |  |  |
| case6  N= |  |  |  |  |  |  |
| case7  N= |  |  |  |  |  |  |
| case8  N= |  |  |  |  |  |  |
| case9  N= |  |  |  |  |  |  |

3. You should do Job 2, Job 3 in your report.

4. Your binary code should be compiled by the following commands under <student\_id>\_pa2 directory. The binary code is located in the bin directory.

make

cd bin

**Submission:**

**Make a directory <student\_id>\_pa2**

**Please copy 5 items /src, /lib, report, makefile, README into directory**

**Then submit a single *\*.tgz* file to CEIBA system.**

1. *<student\_id>\_pa2* directory contains your source code in *src* directory. By simply typing “make” can compile.

2. a report in the *doc* directory. *<student\_id>\_pa2\_report.doc*. Please also submit a printed report in class so that TA can grade faster.

3. a README file that explains your files.

4. We will use our own test cases so do NOT include the input/output directories.

5. The submission filename should be compressed in a single file *<student\_id>\_pa2.tgz*. (e.g. b90901000\_pa2.tgz). If you have a modified version, please add \_v2 as a postfix to the filename and resubmit it (*e.g. b90901000\_pa2\_v2.tgz*). You can use the following command to compress a whole directory:

tar -zcvf <filename>.tgz <dir>

**checkSubmitPA2:**

You must use the checksubmitPA2 script, which is under the /*utility* directory, to check whether your result is correct or not. Please put your *.tgz* file in the */utility* directory. To use this checker, simply type

cd utility

./checkSubmitPA2.sh <filename>.tgz

For example:

./checkSubmitPA2.sh b99901000\_pa2.tgz

This checker will check: 1 decompression, 2. compilation. Please check the correctness of your answers with other students.

We have so many students in the class so we need automatic grading. Any mistake in the submission will result in cost 20% off your score. Please be very careful in your submission.

**Grading:**

20% correctness for Job 1

10% correctness for Job 2

5% correctness for Job 3

30% correctness for Job 4

15% file format and location

20% report

**Bonus:**

10% bonus will be given to any extra features that you add to make your program faster or requires less memory. Please write clearly in your report.

**NOTE:**

Copying other source code can result in zero grade for all students involved.