Algorithm Design and Analysis

Homework #3

Due: 2:20pm, Thursday, October 31, 2013

TA email: ada@csie.ntu.edu.tw

=== Homework submission instructions ===

- For Problem 1, commit your source code to the SVN server (katrina.csie.ntu.edu.tw). You should create a new folder "`hw3`" and put the source code file in it. The filename of the source code should be "`main.c`"; you will get some penalties in your grade if your submission does not follow the naming rule. You have to login to judgegirl system (http://katrina.csie.ntu.edu.tw/judgegirl) to make it judge your program.

- You need to submit your `report of Problem 1` and your `answer to other Problems` via the SVN server (electronic copy, all in a single pdf file named "`hw3_[student ID].pdf`") or to the TA at the beginning of class on the due date (hard copy).

- For hard copies
  - Please use A4 paper.
  - Please write down `your name` and `the page number` on every page of your homework.
  - Please do not use paper with things printed on the other side.
    You are encouraged to write down your answers on both sides of the paper.
  - Please don't use staples to bind you papers.
    The TA will provide you paperclips when you hand in your homework.
  - Please take your homework out of the L-folder when you hand it in, or you might lose your L-folder.

  You may get some penalties in your grade if you do not follow the above rules.

- Discussions with others are encouraged. However, you should write down your solutions by your own words. In addition, for each problem you have to specify the references (the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem.

- **NO LATE SUBMISSION IS ALLOWED** for the homework submission in hard copies - no score will be given for the part that is submitted after the deadline. For submissions via SVN (including the programming assignment and electronic copies of the writing problems), up to one day of delay is allowed; however, the score of the part that is submitted after the deadline will get some penalties according to the following rule (the time will be in seconds):

$$\text{LATE SCORE} = \text{ORIGINAL SCORE} \times (1 - DelayTime/86400)$$

Problem 1. Dam System

(30%)

As you have solved the "Giant Rectangle" problem, you became the prime minister of rectangles. One day, the Rectangle King assigned you a new problem:

" There is a big river in front of our palace. It is composed of many smaller rivers, and each smaller river is also composed of smaller ones until they reaches the estuary. You may notice that our river looks like a tree structure: the big river is the root; each intersection is like a node, and each river is like an edge. The flow after the intersection of several smaller rivers is the sum of every river that merges into it. You can regard the estuaries as the leaves and the original source of flow. The problem is as follows:

If you build a dam on a node, you need to make sure that the capacity of the dam is greater than or equal to the flow of the node. And after building such a dam, there would be no flow coming out of the node (no outflow for the node).

Your goal is to make sure that there is no flow coming out of the root. Given that you cannot build more than k dams due to the budget limit, what is the minimum capacity each dam should have? "

Note: Every dam has the same capacity. A dam can only be built on a node if the flow in the node is less than or equal to the capacity of the dam so that there is no outflow for the node.

Note: Given n nodes, you are allowed to use $O(n)$-space in storing the information related to the given tree.

Hint: You can convert this problem to a similar problem: "Given capacity C, find the minimum number of dams you need." And use the relation to solve the original problem.

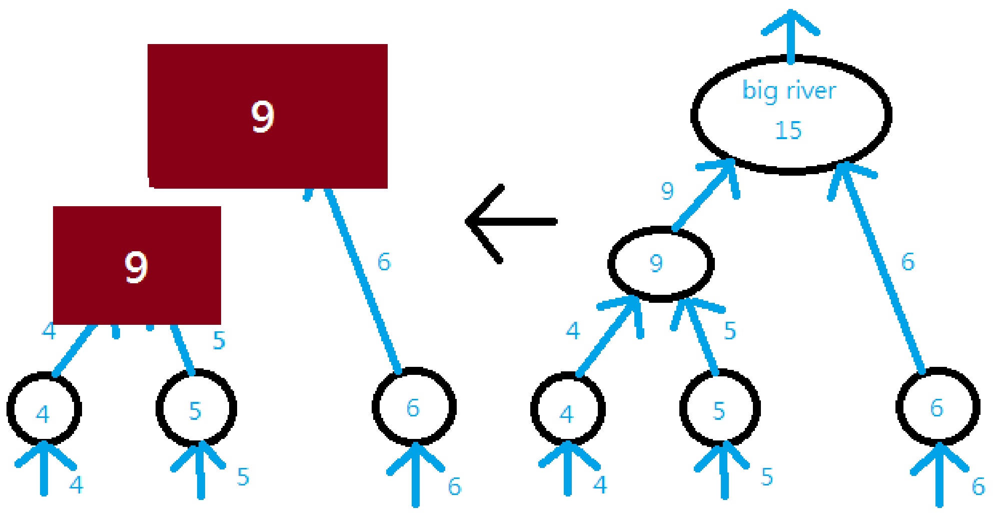

Figure 1: an example

Report(10%)

Please analyze the running time of your program, clearly define the subproblem, and prove that under your definition of subproblems the problem exhibits optimal substructure as well as the greedy property.

Coding(20%)

Input format:

The first line contains the total number of nodes $N \leq 5000$, the total number of dams $K \leq 500$, and the total number of leaves R. And the following N-1 line are the edges. For each edge (a, b), a is the parent node and b is the child node. Note that the node numbered 1 is the root. The last R line give the flows of the leaves. The number R depends on the tree. We guarantee that the sum of flow of these leaves is an **integer**, each flow is also an **integer**, and is greater than 0.

$N\ K\ R$

$a1\ b1$

...

$an-1\ bn-1$

$a1\ f1$

...

$aR\ fR$

Output format:

Please output the minimum capacity the dams can have.

Sample input:

```
5 2 3
1 2
1 3
2 4
2 5
3 6
4 4
5 5
```

Sample output:

```
9
```

***Problem* 2.** Minimum Cost Spanning Tree (15%)

Recall that in last semester we learned "Kruskal's algorithm", which is an algorithm designed to solve the minimum cost spanning tree problem. After the lecture today, you can easily find out that Kruskal's algorithm is actually a greedy algorithm.

But let's try another one, namely, "Prim's algorithm".

Assume that there is an undirectional graph $G = (V, E)$, you are now asked to

1. (3%) write down the pseudo-code of "Prim's algorithm";
2. (6%) prove that the problem exhibits optimal substructure;
3. (6%) prove its greedy property.

For the last two questions, please make sure that you clearly define the subproblem.

***Problem* 3.** Task Scheduler (20%)

There are n tasks in the waiting list, and the time factor of task i is denoted by f_i. Each task may have dependency on another task; for example, if task j depends on task k, you must finish task k before you start to execute task j. Each task can have dependency on at most one task, but there may be more than one task that depends on the same task. Note that there is no cyclic dependency. You need to finish all tasks. Assume each task takes a unit of time. At the beginning, the time is set to 0. If the finished time of task i is t_i, then the cost of task i is $f_i t_i$. Your goal is to find the minimum total cost for finishing all the tasks, $\sum_i f_i t_i$.

For this problem, we define the sub-problem with $n - 1$ tasks by construction. The first step is to find the task with maximum f_i. If the task has no dependency on any other task, delete it; otherwise, combine it with those tasks that it directly depends on into a new task, and the time factor of the new task is the average of that of all sub-task merged into the new task. Figure 2 is an example.

1. (5%) Prove that the first step is correct greedy choice and that the problem has optimal substructure by the above definition of the subproblem.
2. (5%) Design a greedy algorithm to find the minimum total cost, $\sum_i f_i t_i$, and justify that your algorithm runs in $O(n^2)$. Please give the pseudo code of your algorithm and describe how your pseudo code works.
3. (5%) If you adopt suitable data structures in your implementation, the algorithm can run in $O(n \log n)$. Please modify your pseudo code to achieve this running time. You can use the data structures taught in last semester.
4. (5%) If a task is allowed to have dependencies on more than one tasks, is your proof of optimal substructure still valid? If yes, prove it; otherwise, give a counterexample.

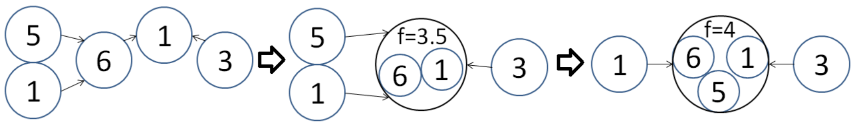

Figure 2: an example

Problem 4. Super Computer Revised (15%)

1. The company that owned the super computer is going to provide service to other companies. User can submit jobs to the super computer and it would only run some of the jobs because there are too many. Each job requires 1 unit of time to run and can be started at any moment. The company has a billing strategy. Every job has a price that can be different from other jobs. Now there are n submitted jobs. If job i is executed before its deadline d_i, the user that submits the job needs to pay the company p_i dollars. The problem is to schedule the jobs during the time interval $[0, \max_{1 \le i \le n} \{d_i\})$ to maximize the profit.
   Please prove that the problem exhibits optimal substructure and has greedy property. You need to define what a subproblem is in the proof. (8%)

2. The company is going to encounter a power failure during the time interval $[S, F)$. Fortunately, the company has n power generators. The i-th power generator is operational in the time duration of $[s_i, f_i)$. The company would like to minimize the number of power generators used during the power failure and still makes sure that the power is never interrupted. Please prove that the problem exhibits optimal substructure and has greedy property. You need to define what a subproblem is in the proof. (7%)

Problem 5. K Partitions (20%)

Given an positive integer sequence of size n, we want to partition it into k pieces. For example, assume that the sequence is [1 2 3 4 5 6 7 8 9] and $k = 3$, there are several ways to partition it, e.g., [1 2 3 | 4 5 6 | 7 8 9] or [1 2 3 4 | 5 6 7 | 8 9] or [1 2 3 4 5 | 6 7 | 8 9]. For each way of partition, we define a variable q = the maximum sum of the k pieces. In the above example, q will be 24(7+8+9), 18(5+6+7) and 17(8+9).

Now, we want to compute the minimum q in all partition methods, defined as q_{min}. In the above example, q_{min} would be 17.

1. (5%) Please define the subproblem, and prove that the problem exhibits optimal substructure.

2. (5%) Use dynamic programming to find q_{min} in the given sequence and k. Please describe your algorithm with the psuedo code and some explanation. (restriction: $O(kn^2)$)

3. (10%) Prove that there is at least a valid greedy choice for the problem, then design an efficient greedy algorithm to find q_{min} in the given sequence and k. Please describe your algorithm with the psuedo code and some explanation.