Assignment 3

Algorithm Design and Analysis

November 8, 2018

Notice:

1. Please submit your answer in hard copy AND submit a digital version to UCAS website http://sep.ucas.ac.cn.
2. Hard copy should be submitted before 9 am. Nov 23 and digital version should be submitted before 11 pm. Nov 23.
3. Please choose at least two problems from Problem 1-4, and finish problems 5-6 on Universal Online Judge before 11 pm. Nov 23.
4. When you're asked to give an algorithm, you should do at least the following things:
   - Describe the basic idea of your algorithm in natural language **AND** pseudo-code;
   - Describe the greedy-choice property and optimal substructure;
   - Prove the correctness of your algorithm.
   - Analyse the complexity of your algorithm.

1 Greedy Algorithm

Given a list of n natural numbers d_1, d_2,...,d_n, show how to decide in polynomial time whether there exists an undirected graph $G = (V,\ E)$ whose node degrees are precisely the numbers $d_1, d_2, \cdots, d_n$. G should not contain multiple edges between the same pair of nodes, or " loop" edges with both endpoints equal to the same node.

2 Greedy Algorithm

There are n distinct jobs, labeled $J_1, J_2, \cdots, J_n$, which can be performed completely independently of one another. Each jop consists of two stages: first it needs to be *preprocessed* on the supercomputer, and then it needs to be *finished* on one of the PCs. Let's say that job J_i needs p_i seconds of time on the supercomputer, followed by f_i seconds of time on a PC. Since there are at least n PCs available on the premises, the finishing of the jobs can be performed on PCs at the same time. However, the supercomputer can only work on a single job a time without any interruption. For every job, as soon as the preprocessing is done on the supercomputer, it can be handed off to a PC for finishing.

Let's say that a *schedule* is an ordering of the jobs for the supercomputer, and the *completion time* of the schedule is the earlist time at which all jobs have finished processing on the PCs. Give a polynomial-time algorithm that finds a schedule with as small a completion time as possible.

3 Greedy Algorithm

Given two strings s and t, check if s is subsequence of t?
A subsequence of a string is a new string which is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. (ie, "ace" is a subsequence of "abcde" while "aec" is not).

4 Greedy Algorithm

Given a rope whose length is n, please cut the rope to m parts to get the maximum product of the length of each part $\prod_{l_1+l_2+..+l_m=n} l_1 * l_2 * ... * l_m$. For example, if a rope with length 8, when we cut it to 2, 3, 3, we can get the maximum product 18.

5 Programming

Description: Some people want to cross a river by boat. Each person has a weight, and each boat can carry a maximum weight of limit. Each boat carries at most 2 people at the same time, provided the sum of the weight of those people is at most limit. Return the minimum number of boats to carry every given person. Note that it is guaranteed each person can be carried by a boat.

Input: The input will be 3 lines. The first line is a single integer n ($1 <= n <= 50000$), which means the number of people. The second line is a single integer l ($1 <= l <= 30000$), represents the weight limit. And the last line contains n integers $p_1, p_2, ..., p_n (1 <= p_i <= l)$, separated by spaces, that represent peoples weights.
Output: The output will consist of one line containing one integer, which represents the minimum number of boats.
Sample Input:
4
3
3, 2, 2, 1
Sample Output:
3

6 Programming

Description: There are N Monkeys and N bananas are placed in a straight line. Each monkey want to have a banana, if two monkeys want to own the same banana, there will be a fight! A monkey can stay at his position, move

one step right from x to $x + 1$, or move one step left from x to $x - 1$. Any of these moves consumes 1 second. Assign monkeys to banana so that not monkey fight each other and the time when the last monkey gets a banana is minimized.

Sample Input:
position of monkeys: 1, 3, 6. Position of banana: 2, 4, 6.
Sample Output:
1 (second).
Sample explanation:
Assign monkey at position 1 to banana at position 2. (1 second)
Assign monkey at position 3 to banana at position 4. (1 second)
Assign monkey at position 6 to banana at position 6. (0 second)
overall time is max(1, 1, 0) = 1 second.