# Assignment 3

Algorithm Design and Analysis

November 22, 2019

**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. Dec 6 and digital version should be submitted before 11 pm. Dec 6.
3. Please choose at least two problems from Problem 1-4, and finish problems 5-6 on Universal Online Judge before 10 a.m. Dec 2.
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

**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.

## 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 a rope whose length is $n$($n$ is an integer), please cut the rope to $m$ parts(each part's length is an integer) 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.

# 4 Greedy Algorithm

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 boat's weight 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.

# 5 Programming

**Description**:
Say you have an array for which the $i$th(0-indexed) element is the price of a given stock on day $i$(0-indexed).
Design an algorithm to find the maximum profit. You can buy one and sell one share of the stock multiple times, but you must sell the stock before you buy again.

**Input**:
An array with int elements, separated by spaces.
**Output**:
An int number.
**Sample Input**:
7 1 5 3 6 4
**Sample Output**:
7
**Sample explanation**:
Buy in in day 1, sell out in day 2, buy in in day 3, sell out in day 4. $7 = 4 + 3 = (-1 + 5) + (-3 + 6)$.

# 6 Programming

**Description**:
Given an array of size n that has the following specifications:

1. Each element in the array contains either a monkey or a banana.
2. Each monkey can eat only one banana.
3. A monkey cannot eat a banana which is more than $K$ units away from the monkey.

We need to find the maximum number of bananas that can be eaten by monkeys.

**Input**:

1. A array of size $n$ contain chars 'B' (represent Banana) or 'M' (represent Monkey). $0 \leq n \leq 10000$.
2. An int $K$, $0 \leq K \leq 10000$

**Output**:
An int, the maximum number of bananas can be eaten by monkeys.
**Sample Input**:
{'M', 'B', 'B', 'M', 'B'}
$k = 1$
**Sample Output**:
2
**Sample explanation**:
2 bananas can be eaten, first monkey eat the first banana, second monkey eat either the second or third banana