Assignment 2 (75 marks)

Due Monday Feb. 14, by 5pm.

- Assignments in MS Word format should be handed in via D2L.

---

1. **(15 marks)**A student taking a test uses greedy strategies to maximize the test results. The input to this problem consists of the test time K in minutes, and the (estimated) times to solve each of the n questions on the test, $T[1..n]$, where time $T[i] > 0$ is the time in minutes for solving question i, $1 \leq i \leq n$. We assume no partial credits will be given; thus a completed test question gets full credits while incomplete answers get 0 credits. We also assume that $\sum_{i=1}^{n} T[i] > K$ so that there is not enough time to solve all test questions.

   (a) **(4 marks)**Assume that the goal is to maximize the total amount of time the student uses for the completed test questions. Does a greedy algorithm that chooses the longest question at each step (always) generates optimal solutions? If not, given a counterexample.

   (b) **(11 marks)**Assume that the goal is to maximize the total number of completed test questions. Design a greedy algorithm for it and prove that your algorithm is correct (always generates optimal solutions).

2. **(25 marks)** A friend of yours has a lot of books, and also has a large bookcase with movable shelves. She wants to organize her books onto shelves in a way that minimizes the total shelf height needed, and has asked you for help.
   You investigate her situation and discover that she has n books of different heights (set of heights $= \{b_1, \cdots, b_n\}$). All books are of width 1 unit, and each shelf is 10 units wide (and so has space for 10 books). Since the shelves are adjustable, the height of a shelf is defined as the height of the tallest book on the shelf. The total shelf height is the sum of all the individual shelf heights. Much to your relief, she is not concerned about the order of the books.

   (a) (7 marks) Design a greedy algorithm that will find the organization of minimum height.

   (b) (2 marks) Analyse your algorithm's running time.

(c) (6 marks) Prove that your algorithm works correctly.

(d) (10 marks)Implement your algorithm in Java. Your implementation takes the number of books , a sequence of integers (heights of books) as input, and outputs the total shelf height.

3. **(15 marks)** Suppose that n files with lengths L_1, L_2, $\cdots$, L_n are to be stored on a tape, so that access to them is sequential. If the files are stored in the order i_1, $\cdots$, i_n, then the time to retrieve file i_k is the sum of the file lengths of all files before and including i_k:

$$T_k = \sum_{j=1}^{k} L_{i_j} \tag{1}$$

We would like to order the files to minimize the average retrieval time, so we want to minimize

$$\frac{1}{n}\sum_{k=1}^{n} T_k \tag{2}$$

(a) Design and write a greedy algorithm that will find the minimum average retrieval time for the files.

(b) Analyse your algorithm's running time.

(c) Prove that your algorithm works correctly.

4. **(5 marks)** Consider the greedy algorithm for coin changing problem, given the denominations 1, 4, 8, 10, and 25. Does the greedy algorithm work? If yes, prove it. Otherwise, give a counterexample.

5. **(15 marks)** You have a ladder with n rungs, and a supply of identical glass jars. You are trying to find the highest safe rung from which you can drop a jar without the jar breaking. If you use binary search, you can find this rung in $\Theta(logn)$ worst-case time, but you will break a lot of jars. Alternatively, you can find the highest safe rung while breaking only one jar, but it will take worst-case time in $\Theta(n)$. You would like to determine a compromise between the worst-case time and the number of jars broken. We assume that there is no time taken to climb the ladder itself, so that any rung can be tested in constant time.

(a) Give an algorithm for finding the highest safe rung, assuming that you are allowed to break 2 jars. This algorithm should run in worst-case time in $\Theta(f(n))$, where $f(n)$ is sublinear, i.e. $f(n)$ is $o(n)$ (little-o). Analyse your algorithm's running time.

(b) Extend your solution to give an algorithm to find the highest safe rung, assuming that you are allowed to break k jars (where k is a given value with $1 \leq k \leq n$). Analyse your algorithm's running time.