Problem Set 2: Greedy Algorithm

1. Given a set of $2n$ integers, make n pairs $(a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)$ from these $2n$ integers such that the sum $\sum_{i=1}^{n} \max\{a_i, b_i\}$ is maximized. Design an algorithm that solves this problem in
   (a) (EASY) $O(n \log n)$ time
   (b) (MEDIUM) $O(n)$ time

2. (EASY) You have landed a cracking job in Bangalore in a company named PPWATERBALLS. Fortunately, PPWATERBALLS has agreed to pay k rupees for your relocation to Bangalore. You have stored your belonging into n boxes where the box i contains $box[i]$ items. The cost of transporting each box is 1 rupee. Design an algorithm that will relocate the maximum number of items.

3. (EASY) Given two numbers a and b, their absolute difference is $|a - b|$. Given an array of size n, design an algorithm to find two number in it with minimum absolute difference.

4. (MEDIUM) At IIT Gandhinagar, students undertake a total of $n + k$ courses as part of their B.Tech curriculum. Upon completion of their degree, students have the option to request the institute to convert the grades of k courses into a pass grade. This feature is thoughtfully designed to address situations where exceptionally bright students may have encountered difficulties in a few courses, potentially due to personal preferences or other reasons. By utilising this option, the CPI (Cumulative Performance Index) of such bright students remains unaffected by a few challenging courses, ensuring a fair evaluation of their overall academic performance. In this problem, we will change the definition of CPI and ask you to maximise it.

   The courses are arranged in an arbitrary order, denoted as $c_1, c_2, \ldots, c_{n+k}$. Each course c_i is assigned a grade g_i, ranging from 0 to 9. The temporary CPI is represented by the concatenation of these grades: $g_1 g_2 \ldots g_{n+k}$. For instance, if a student receives grades 1, 9, and 7 in courses 1, 2, and 3 respectively, their temporary CPI would be 197.

   The objective is to strategically select k courses from the transcript for removal, with the aim of improving the grades in the remaining n courses. For example, if given the opportunity to remove one course from the transcript, the student would select course 1 in the aforementioned scenario. Consequently, their final CPI would be 97.

   Formally, students are provided with grades for $n + k$ courses, and they must choose k grades for removal to maximise the remaining cumulative performance. An efficient algorithm is required to accomplish this task.

5. (EASY) You have a unlimited supply of k coins of denomination $2^0, 2^1, \ldots, 2^{k-1}$ and a number n. You want to get the change of n. Design an algorithm that will give you the change using minimum number of coins. For example, to get a change of 5, you require one coins of denomination 4 and 1 respectively.

6. (HARD) You have an unlimited supply of k coins of denomination $a_1, a_2, \ldots, a_k$ and a number n. Your want to check if you can get a change of each number from 1 to n. However, there is one important condition. For making the change of number i ($1 \leq i \leq n$), you can use each coin only once. Unfortunately, this condition is not easy to satisfy. For example, if you start with coins $1, 2, 5$ and $n = 7$, then you cannot make the change of 4.

   If we also had coins of denomination 4, then the above problem becomes easy. How many new denomination coin will you use to get the change of all number? Design an algorithm that adds minimum number of new denomination coins.

7. (MEDIUM) You are given n number (each ranging from 0 to 9) that may contain some duplicates. You want to remove all duplicate number but there is one important condition: the remaining number(after the removal of the duplicates) should be maximised. For example, if the initial numbers were 19232192, then the answer is 9321. Design and analyse an efficient algorithm that performs this task.

8. (EASY) n people are stranded on an island. You are given the job to rescue them. You have an unlimited supply of boats but each boat has a maximum weight limit of k and can accommodate at most two people. Given the weight of all n people on the island, design an efficient algorithm to find the minimum number of boats required to rescue them.

9. (MEDIUM) You are given an array A of n number which are initially all 0. There is another target array B of size n such that each entry is B is ≥ 0. You are allowed to perform the following operations: in each round, you take a subarray of A and increase each element in the subarray by 1. What is the minimum number of rounds required to reach the target B from A. Design an efficient algorithm to perform this task.

   For example, if $B = [2, 3, 4, 1]$, then in the first round we increment all the elements of array A to get $A = [1, 1, 1, 1]$. In the second round, we increment $A[0 \ldots 2]$ to get $A = [2, 2, 2, 1]$. In the third round, we increment $A[1, 2]$ to get $A = [2, 3, 3, 1]$ and in the fourth round we increment $A[2]$ to reach the target B.

10. (EASY) You are pairing couples for a very conservative formal ball. There are n men and m women, and you know the height and gender of each person there. Each dancing couple must be a man and a woman, and the man must be at least as tall as, but no more than 3 inches taller than, his partner. You wish to maximise the number of dancing couples given this constraint. Design an efficient algorithm that performs this task.

11. (EASY) You are baby-sitting n children and have $m \geq n$ cookies to divide between them. You must give each child exactly one cookie (of course, you cannot give the same cookie to two different children). Each child has a greed factor $g_i, 1 \leq i \leq n$ which is the minimum size of a cookie that the child will be content with; and each cookie has a size $s_j, 1 \leq j \leq m$. Your goal is to maximise the number of content children, i.e.., children i assigned a cookie j with $g_i \leq s_j$. Give a correct greedy algorithm for this problem, prove that it finds the optimal solution.

12. (MEDIUM) In a class, there are n assignments. You have H hours to spend on all assignments, and you cannot divide an hour between assignments, but must spend each hour entirely on a single assignment. The ith hour you spend on assignment j will improve your grade on assignment j by $B[i, j]$, where for each j, $B[1, j] \geq B[2, j] \geq \cdots \geq B[H, j] \geq 0$. In other words, if you spend h hours on assignment j, your grade will be $\sum_{i=1}^{h} B[i, j]$ and time spent on each project has diminishing returns, the next hour being worth less than the previous one. You want to divide your H hours between the assignments to maximise your total grade on all the assignments. Give an efficient algorithm for this problem.

13. (MEDIUM) Consider the following preemptive scheduling problem. You are trying to schedule jobs on a machine that are arriving at different times, and require different numbers of steps to finish. Your schedule can be preemptive, in that you can start one job, then switch to another, then finish the first job. You are trying to minimise the sum over all jobs of the time they finish. More precisely, the input is a sequence of n jobs, $Job_i = (a_i, d_i)$, where a_i is an integer giving the arrival time of the job (first time step when we could start the job), and d_i is a positive integer giving the duration of the job, the number of steps required to finish the job. A schedule specifies for each time step, which job we are working on. At time step, t, we can only work on Job_i if $a_i \leq t$; and there must be at least d_i steps where we are working on Job_i. The finish time for Job_i is the last time when Job_i is scheduled. The objective is to find a schedule that minimizes the sum of all the finish times.
   Example: Job 1: Arrives at 8 AM: Practice piano. Duration: 3 hours.
   Job 2: Arrives at 9 AM. Answer morning email. Duration 1 hour.
   Job 3: Arrives at 11 AM. Do CSE homework. Duration 4 hours.
   Schedule: 8-9: practice piano; 9-10 answer email; 10-12: practice piano; 12-16: do CSE homework.
   Finish times: email: 10; piano: 12; homework: 16. Total: 38.

   Give an efficient greedy algorithm for this problem.

14. (HARD) You are given an array $A[1 \ldots n]$. The element $A[i]$ is said to be a local maximum if $A[i] > A[i-1]$ and $A[i] > A[i+1]$. You have to change minimum number of entries in A such that there is no local maximum after the change. Design an efficient algorithm for this problem and prove it correctness.

15. (MEDIUM) You want to create a scientific laboratory capable of monitoring any frequency in the electromagnetic spectrum between L and H. You have a list of possible monitoring technologies, T_i, $i = 1, \ldots, n$, each with an interval $[l_i, h_i]$ of frequencies that it can be used to monitor. You want to pick as few as possible technologies

that together cover the interval $[L, H]$. Are the following two greedy strategies correct for this problem. Prove or disprove.

(a) Candidate Greedy Strategy I: First, buy the technology that covers the longest sub-interval within (L, H). (i.e.,The longest interval (l, h), but not including the sub-intervals (l, L) and (H, h) outside the interval we need covering.) At each subsequent step, buy the technology that covers the largest total length that is still uncovered.

(b) Candidate Greedy Strategy II: Look at all the technologies with $l \leq L$. Of these, buy the one $T_i = (l_i, h_i)$ with the largest value of h_i. Repeat the process to cover the remaining interval, (h_i, H).

16. (EASY) Given a weighted undirected graph G with distinct edge weights, you have to find an edge that is definitely not in the minimum spanning tree. Design a $O(m+n)$ algorithm to find such an edge. Prove that your algorithm is correct.

17. (EASY) A server has n customers waiting to be served. The service time required by each customer is known in advance: it is t_i minutes for customer i. So if, for example, the customers are served in order of increasing i, then the ith customer has to wait $\sum_{j=1}^{i} t_j$ minutes. We wish to minimize the total waiting time

$$T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i)$$

Give an efficient algorithm for computing the optimal order in which to process the customers.

18. (EASY) Consider an undirected graph $G = (V, E)$ with nonnegative edge weights $w_e \geq 0$. Suppose that you have computed a minimum spanning tree of G, and that you have also computed shortest paths to all nodes from a particular node $s \in V$. Now suppose each edge weight is increased by 1 :the new weights are $w'_e = w_e + 1$.

(a) Does the minimum spanning tree change? Give an example where it changes or prove it cannot change.

(b) Do the shortest paths change? Give an example where they change or prove they cannot change.

19. (EASY) There are n events, each takes one unit of time. Each event i will provide a profit of g_i dollars if it is started at or before time t_i, but will provide zero profit if it is not started by time t_i (so there is no point in scheduling event i unless it can be scheduled to start by time t_i). Here, g_i, $t_i \geq 0$ and t_i may NOT be an integer. An event can start as early as time 0 and no two events can be running simultaneously. The goal is to feasibly schedule a subset of the events to maximize the total profit.

(a) Prove that there exists an optimal schedule OPT in which every event that is scheduled is scheduled to start at an integral time. Note that in such a solution, each event i is either scheduled to start by time $\lfloor t_i \rfloor$ or not scheduled at all.

(b) Design an efficient greedy algorithm which only schedules events at integral start times. [Hint: Let $T = \max_i \lfloor t_i \rfloor$. Think about which event you would schedule to start at time T.]

(c) Prove that your algorithm always returns an optimal solution.

(d) Analyze the worst-case running time of your algorithm. Explicitly state the data structures that your algorithm uses.

20. (HARD) Your mother has defined a set of n tasks. For each task i, there is a reward r_i. The rewards are integers and may take negative values. For example, if you clean the house, then the reward is positive. But if you watch TV, then the reward is negative. You can choose any subset of these n tasks to maximize the total reward. Let us assume that you choose k tasks.

In general, it is better to choose tasks with positive rewards and avoid negative rewards. However, your mother knows that watching TV is not altogether a bad activity. So, here is the twist.

Suppose you choose to do k tasks out of n. If you do task 1 on day 1, then you will receive a reward of kr_1. Similarly, if you do task 2 on day 2, then your reward will be $(k-1)r_2$. Let us assume that you choose to do task i on day i, then your total reward is $\sum_{i=1}^{k}(k-i+1)r_i$. With this condition, can you come up with an example when even performing a task with negative reward is fine? Design an efficient algorithm that find the number of task that you should perform to maximize the reward.