CS 5592: Design and Analysis of Algorithms
Mid-term Make Up

Author: Hayden McParlane

(F.3) Given a sequence of n (n > 2) real numbers in an array, A[1], A[2], A[3], ..., A[n], we wish to find two numbers A[i] and A[j], where i < j, such that A[i] $\leq$ A[j] and their distance is the largest. That is,

$$j - i = \max\{v - u \mid 1 \leq u < v \leq n \text{ and } A[u] \leq A[v]\}$$

If no such pair, report -∞. Please design an $\mathcal{O}(n \cdot lgn)$ algorithm to solve this problem. You can use any method , including divide and conquer, greedy or dynamic programming.

The following should be noted about the algorithm below (in addition to the statements in the problem):

1. Function SortAndTrackIndices(A, I) returns array A sorted in non-decreasing order ($a_1 \leq a_2 \leq a_3 \leq ... \leq a_n$) as well as an array, I, which maps each a_i to its original position, I_i.

Algorithm 1 Greedy solution to the problem above. Make sure to take note of the assumptions above.

```
function MaxDist(A, n)
    I ← ∅
    A, I ← SortAndTrackIndices(A, I)
    i ← ∞
    j ← −∞

    for k from 1 to n do
        if j < I[k] then
            j ← I[k]
        end if
    end for

    for k from n to 1 do
        if I[k] < i and I[k] ≠ j and A[k] ≤ A[j] then
            i ← I[k]
        end if
    end for

    if i equals ∞ then
        return -∞
    else
        return i and j
    end if

end function
```

F.4 Consider the following activity selection problem. Suppose we have a lecture hall which is available from time t = 0. There are n activities, $a_1, a_2, a_3, ..., a_n$, that apply for using the lecture hall. If the application of $a_i (1 \leq i \leq n)$ is accepted, then it must start before or at $s_i (\geq 0)$ and it must continuously use the hall for t_i time. Moreover, at any time, at most one activity is allowed to use the lecture hall. We wish to select a set R of activities such that they can be arranged to use the lecture hall with no time conflict and the total time the lecture hall is used by these activities is maximized,

$$\sum_{a_i \in R} t_i \text{ is maximized.}$$

A Re-define the problem as a decision problem.

B Polynomially reduce the set partition problem to the decision problem in the first part.

F.4.A Because this optimization problem requests that we maximize the total time the lecture hall is used our restatement of the problem will focus on time usage of the hall. Therefore, the problem can be **restated as a decision problem in the following manner**:

Suppose we have a lecture hall which is available from time t = 0. There are n activities, $a_1, a_2, a_3, ..., a_n$, that apply for using the lecture hall. If the application of $a_i (1 \leq i \leq n)$ is accepted, then it must start before or at $s_i (\geq 0)$ and it must continuously use the hall for t_i time. Moreover, at any time, at most one activity is allowed to use the lecture hall. **Can we select a set of compatible activities such that they occupy the lecture hall for a duration of k units of time?**

F.4.B Transformation of the set partition problem into an instance of this problem can be done using the following strategy.

The set partition problem is defined as follows: Given a set, $S = \{v_1, v_2, v_3, ..., v_n\}$, can we partition that set into two subsets such that

$$\sum_{v_i \in A} v_i = \sum_{v_i \in S-A} v_i$$

Following this transformation algorithm we can accomplish that task:

1. Compute $M = \sum_{v_i \in S} v_i$.
2. We can assume any number v_i in S is less than or equal to M/2. Otherwise, no partition is possible and the transformation is trivial.
3. For each $v_i \in A$, construct an activity such that:

   $t_i = v_i$

   $s_i = (M/2) - v_i$

4. For each $v_i \in S - A$, construct an activity such that:

$$t_i = v_i$$
$$s_i = M - v_i$$

5. Set $k = M$.

An explanation of logic follows. Each numer in the list below corresponds to the same numbered step above.

1. We compute the total sum of all values in S because that value is needed in steps that follow.

2. We assume all numbers are less than or equaled to M/2 because if that's not the case a valid partition cannot be done satisfying the requirements of the set partition problem. Such sets simply result in the answer No, and are therefore trivial.

3. For all $v_i \in A$, we transform v_i directly into t_i because it is a sensical approach to this transformation. The more interesting aspect of this step is transformation of v_i into s_i by subtracting v_i from $(M/2)$. This is necessary because we can view this subset of S as occurring in the first half of the time during which we have scheduled the lecture hall. All activities that result from this transformation will occupy the lecture hall at some instance between time t_0 and time $(M/2)$.

4. This step is necessarily separated from the previous step because we can view all of the activities that take place in this subset as occuring after all of those that came from the subset A. All activities in the subset A occurred between t_0 and $(M/2)$, whereas all activities that result from this step will occur between time $(M/2)$ and M. Therefore, activities that are part of this second set, $S - A$, have s_i calculated by subtracting v_i from M.

5. Finally, we need to set our k value. In our new problem, k is to represent the total duration during which the lecture hall is occupied. Because $M = \sum_{v_i \in S} v_i$ and v_i was transformed to t_i directly and without operation (i.e, they are one-to-one), $M = \sum_{v_i \in S} v_i = \sum_{t_i \in A} t_i + \sum_{t_i \in S-A} t_i =$ the total duration during which the lecture hall will be occupied. This value is, therefore, our desired k, which we defined as being the duration during which the lecture hall will be occupied.