

Ph.D. Qualifying Exam: Analysis of Algorithms

This is a closed book exam. The total score is 100 points. Please answer all questions.

1. You are given a set A of n different numbers.

- (20 points) (a) Can you find an $o(n^2)$ (little-oh) algorithm to determine if there exist two numbers x, y in A such that $x + y = m$? Please justify why your algorithm is $o(n^2)$.

Solution:

1. $B \leftarrow$ Sort A using an efficient algorithm, such as Merge-Sort. ($O(n \log n)$)
2. For each element y in A , perform binary search for $m - y$ in B . ($O(n \log n) = O(n \log n)$)
3. If such a y is found, let $x = m - y$. If $x \neq y$, then $x + y = m$ is satisfiable. Otherwise, it is not.

The total complexity is thus $O(n \log n)$ which is $o(n^2)$.

- (25 points) (b) Can you find an $o(n^3)$ (little-oh) algorithm to determine if there are three numbers x, y, z in A such that $x + y + z = m$? Please justify why your algorithm is $o(n^3)$.

Solution:

1. Create an array B of size n^2 and save all possible values of $x + y$, ($x \neq y$). ($O(n^2)$)
2. Sort B using an efficient sort algorithm, such as Merge-Sort. ($O(n^2 \log n^2) = O(n^2 \log n)$)
3. For each element z in A , perform binary search for $m - z$ in B . ($O(n \log n^2) = O(n \log n)$)
4. If such a z is found and $z \neq x, z \neq y$ and $x + y = m - z$, then $x + y + z = m$ is satisfiable. Otherwise, it is not.

The total complexity is thus $O(n^2 \log n)$ which is $o(n^3)$.

2. Let I_1, \dots, I_n be n intervals, where interval I_i is defined by set $[a_i, b_i]$, i.e., starting from a_i and ending at b_i . We assume that the intervals are sorted by the ending time b_i in non-decreasing order. We also give a value v_i to interval I_i . We define the following two problems:

The interval scheduling problem P_1 : Find a maximum number of disjoint intervals that do not overlap with each other.

Example: Four intervals are given as $I_1 = [1, 2], I_2 = [2, 3], I_3 = [1, 4], I_4 = [4, 5]$. A solution is $\{I_1, I_2, I_4\}$.

The weighted interval scheduling problem P_2 : Find some subset of disjoint intervals, such that the sum of the values of the disjoint intervals is maximized.

Example: Four intervals and their values are $I_1 = [1, 2], v_1 = 0.9; I_2 = [2, 3], v_2 = 0.5; I_3 = [1, 4], v_3 = 4; I_4 = [4, 5], v_4 = 2$. A solution is $\{I_3, I_4\}$.

(15 points)

- (a) Please give a linear greedy algorithm to solve the interval scheduling problem P_1 .

Solution:

```

INTERVAL-SCHEDULER( $a, b$ )
1   $n \leftarrow \text{length}[a]$ 
2   $i \leftarrow 1$ 
3   $DI \leftarrow \{I_1\}$ 
4  for  $m \leftarrow 2$  to  $n$ 
5      do if  $a_m \geq b_i$ 
6          then  $DI \leftarrow DI \cup \{I_m\}$ 
7           $i \leftarrow m$ 
8  return  $DI$ 

```

Running time: $\Theta(n)$.

(15 points)

- (b) The following greedy algorithm is proposed to solve the weighted interval scheduling problem P_2 . The intuition is to select intervals with high "density", i.e., value-to-length ratio.

GREEDY-WEIGHTED-INTERVAL-SCHEDULER(a, b, v)

```

1  Compute the value density  $\rho_i$  of each interval by  $\frac{v_i}{b_i - a_i}$ 
2   $J \leftarrow \text{Sort } I_1, \dots, I_n \text{ by decreasing value density } \rho_i$ 
3   $DI \leftarrow \{\}$ 
4  for  $i \leftarrow 1$  to  $n$ 
5      do  $\text{Overlap} \leftarrow \text{false}$ 
6          for each interval  $L$  in  $DI$ 
7              do if  $J_i$  overlaps with  $L$ 
8                  then  $\text{Overlap} \leftarrow \text{true}$ 
9                  Break out of the for-loop
10     if  $\text{Overlap} == \text{false}$ 
11         then  $DI \leftarrow DI \cup \{J_i\}$ 
12 return  $DI$ 

```

Example: Let four intervals and their values be $I_1 = [1, 2], v_1 = 0.9; I_2 = [2, 3], v_2 = 0.5; I_3 = [1, 4], v_3 = 4; I_4 = [4, 5], v_4 = 2$. By following the above algorithm, we can

- compute value densities $\rho_1 = 0.9, \rho_2 = 0.5, \rho_3 = 4/3, \rho_4 = 2$
- sort the intervals by the densities, we get I_4, I_3, I_1, I_2
- find disjoint intervals in order from above. Let DI be the final solution, we get $DI = \{I_4, I_3\}$. We cannot choose I_1 or I_2 any further because they both overlap with I_3 .

It is not hard to verify by enumerating all $2^4 - 1 = 15$ possible selections that DI gives the maximum total value of 6.

However, this greedy algorithm does not guarantee an optimal solution. Give a counter example to show that it does not work.

Solution: This greedy algorithm will not work. Counter example:

- $I_1 : [1, 2], v_1 = 0.9 \quad (\rho_1 = 0.9)$
- $I_2 : [2, 3], v_2 = 0.5 \quad (\rho_2 = 0.5)$
- $I_3 : [1, 4], v_3 = 4 \quad (\rho_3 = 4/3)$
- $I_4 : [3, 6], v_4 = 3 \quad (\rho_4 = 1)$

Using the greedy algorithm, $\{I_3\}$ will be selected with a total value of 4. However, a better solution (not found by the greedy algorithm) is to select $\{I_1, I_2, I_4\}$ with a total value of 4.4. Thus the greedy algorithm does not guarantee an optimal solution.

(25 points)

- (c) Can you find an efficient dynamic programming solution to the weighted interval scheduling problem P_2 ?

Solution: $DI_{i,j}$: a subset of activities that

- happen after I_i finishes and before I_j starts, **and**
- are compatible with all other activities occurring **outside** of $[b_i, a_j]$.

Or equivalently,

$$DI_{i,j} = \{I_k | b_i \leq a_k \leq b_k \leq a_j\}$$

$V[i, j]$: The maximum value of **compatible** events in $DI_{i,j}$. $V[i, j] = 0$ when $i \geq j$.

The following recursive solution can be implemented with a dynamic programming algorithm.

$$V[i, j] = \begin{cases} 0 & \text{if } DI_{i,j} = \phi \\ \max_{i < k < j, I_k \in DI_{i,j}} V[i, k] + V[k, j] + v_k & \text{otherwise} \end{cases}$$