Midterm 2 Solution

Q1 Greedy

Consider the activity selection problem we discussed in class. There are other possible greedy choices than the ones we defined in class or homework. For each of the following alternative greedy choices, either prove or disprove that the resulting algorithm has the greedy choice property.

(Note: To prove: use the cut-and-paste method; to disprove: provide a counterexample as we did in homework.)

1. (15 points) Pick the compatible activity with the latest start time.

   Answer:

   Let $OPT = \{o_1, o_2, ..., o_k\}$ be an optimal solution to the problem, and $B = \{b_1, b_2, ...\}$ be a greedy solution using the greedy choice of choosing the latest start time.

   Suppose OPT is sorted in non-increasing order of start time. B is also sorted in non-increasing order of start time.

   Let $B' = OPT - \{o_1\} + \{b_1\} = \{b_1, o_2, o_3, ..., o_k\}$.

   - $|B'| = |OPT| - 1 + 1 = |OPT|$ so B' has maximum size
   - By greedy choice: $s[b_1] \geq s[o_1]$

     Since OPT is compatible: $s[o_1] \geq f[o_2]$

     So $s[b_1] \geq f[o_2]$.

     Similary, b_1 starts after $o_2, o_3, ..., o_k$ and $o_2, o_3, ..., o_k$ are compatible.

     So B' is compatible.

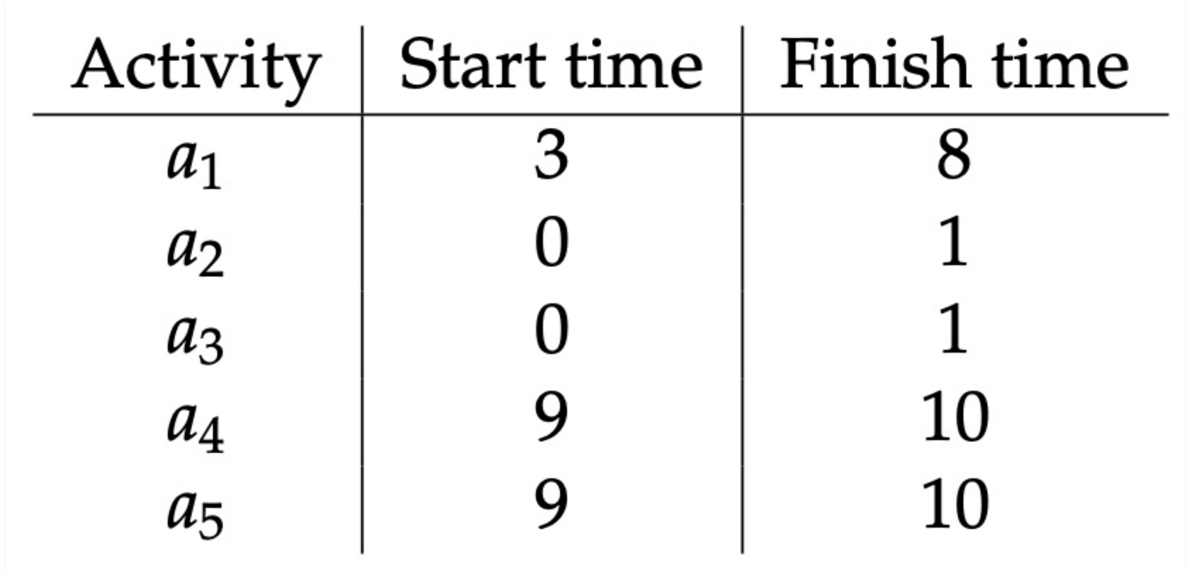
   Conclusion: B' is an optimal solution.

2. (15 points) If no activities conflict, choose them all. Otherwise, discard the activity with longest duration among the remaining activities and recurse.

   Answer:

   The resulting algorithm does not have the greedy choice property.

   Counter example:



Activity	Start time	Finish time
a_1	3	8
a_2	0	1
a_3	0	1
a_4	9	10
a_5	9	10

One of the optimal solution is $\{a_2, a_1, a_4\}$.

But the greedy choice will first discard a_1, then choose two of the remaining four activities. So the size of the greedy solution is 2.

Q2 Divide and Conquer

We want to find the largest and second largest values in an array of numbers.

1. (2 points) Suppose the input array is $A = [5, 10, 2, 7, 9, 8]$. What is the largest? What is the second largest?

   Answer: largest = 10, second largest = 9

2. Suppose the input array is $A = [5, 10, 2, 7, 9, 8]$. If we divide it into two halves: $A1 = [5, 10, 2]$ and $A2 = [7, 9, 8]$.
   1. (2 points) What is the largest and what is the second largest in $A1$?

      Answer: largest = 10, second largest = 5
   2. (2 points) What is the largest and what is the second largest in $A2$?

      Answer: largest = 9, second largest = 8
   3. (10 points) Given the four values computed in the above questions 2.1 and 2.2, how to determine the largest and second largest of the entire array A?

      Answer:

      Let the largest and second largest in $A1$ be a_1 and b_1, and the largest and second largest in $A2$ be a_2 and b_2.

      We have $a_1 \geq b_1$, and $a_2 \geq b_2$.

      If $a_1 \geq a_2$ and $b_1 \geq a_2$, then the largest in A is a_1, the second largest is b_1.

      If $a_1 \geq a_2$ and $b_1 < a_2$, then the largest in A is a_1, the second largest is a_2.

      If $a_1 < a_2$ and $a_1 \geq b_2$, then the largest in A is a_2, the second largest is a_1.

      If $a_1 < a_2$ and $a_1 < b_2$, then the largest in A is a_2, the second largest is b_2.

3. (19 points) Provide a divide-and-conquer algorithm for the problem. Write your algorithm in pseudo-code. The algorithm should run in $\Theta(n)$, where n is the number of elements in the input. For simplicity, you may assume the size of the problem to be an exact power of 2.

   Answer:

```
two_maxes(nums):
    return helper(nums, 0, nums.size - 1)

// return (largest, second largest)
helper(nums, l, r):
    if l >= r:
        return (infinity, infinity)
    if r == l + 1:
        if nums[l] > nums[r]:
            return (nums[l], nums[r])
        return (nums[r], nums[l])
    m = (r-l) / 2 + l
    a1, b1 = helper(nums, l, m)     // a1 >= b1
    a2, b2 = helper(nums, m+1, r)   // a2 >= b2
    if a1 >= a2:
        if b1 >= a2:
            return (a1, b1)
        return (a1, a2)
    else:
        if a1 >= b2:
            return (a2, a1)
        return (a2, b2)
```

Q3 Dynamic Programming

We have discussed the 0-1 Knapsack problem in one of the discussion sections.

Given inputs $C = 5$, $w = [1, 2, 3, 4]$, $v = [1, 10, 15, 20]$, compute $r[i][j]$ for each $0 \leq i \leq 4$ and $0 \leq j \leq 5$ and find the set of items we carry to get the maximum possible value.

Please write down the values for r in a table. You do not need to write down the detailed steps of computing the values.

Answer:

i \ j	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	1	1	1	1	1
2	0	1	10	11	11	11
3	0	1	10	15	16	25
4	0	1	10	15	20	25

Items we carry: items 2 and 3