

Homework 4

Robbie McKinstry, Jack McQuown, Cyrus Ramavarapu

9 September 2016

Greedy Problems

Problem 12:

This greedy algorithm can be shown to be correct by an exchange argument.

Let Alg be the process by which the greedy algorithm operates. Assume that there is some input I such that $Alg(I)$ is incorrect. Let $Opt(I)$ be an optimal solution that agrees with the most number of steps with $Alg(I)$.

Problem 18:

A:

The following counter example shows that this algorithm does not work.

Let the jobs be

$$J_1 = (1, 5, 5)$$

$$J_2 = (4, 2, 100)$$

This algorithm will do the following scheduling:

Time	1	2	3	4	5	6	7
Job	J_1	J_1	J_1	J_1	J_2	J_2	J_1

Since J_1 was scheduled needed to finish after its deadline, the algorithm outputs 0. However, there is a feasible schedule for these two jobs.

Time	1	2	3	4	5	6	7
Job	J_1	J_1	J_1	J_1	J_1	J_2	J_2

B:

The following counter example show that this algorithm does not work.

Let the jobs be

$$J_1 = (1, 2, 2)$$

$$J_2 = (1, 1, 100)$$

This algorithm will do the following scheduling:

Time	1	2	3	4
Job	J_2	J_1	J_1	J_2

C:

Dynamic Programming

Problem 1:

A:

B:

To show that only $O(n^2)$ operations are needed if every duplicate $T(i)$ is calculated only once, begin by expanding the sum in the recurrence.

$$T(n) = \sum_{i=1}^{n-1} T(i)T(i-1)$$

$$T(n) = T(1)T(0) + T(2)T(1) + T(3)T(2) \cdots + T(n-2)T(n-3) + T(n-1)T(n-2)$$

Since every $T(i)$ will only be calculated once, following sequence can be observed by counting the number of operations needed to determine each $T(i)$.

T (i)	T (2)	T (3)	T (4)	T (5)	T (6)
Ops	1	3	5	7	9

It can be shown that the $T(i + 1)$ element of the sum requires two additional operations to calculate: *a multiplication and an addition*. Hence, this sequence will continue. It can be proven inductively that a closed form expression for the sum of operations required is n^2 . Therefore, in this case $O(n^2)$ operations are required.

C:

A $O(n)$ algorithm can be derived from the original recurrence relationship by first eliminating the summation by calculating $T(n + 1)$ in the following manner.

$$T(n + 1) = \sum_{i=1}^n T(i)T(i - 1)$$

$$T(n) = \sum_{i=1}^{n-1} T(i)T(i - 1)$$

$$T(n + 1) - T(n) = \sum_{i=1}^n T(i)T(i - 1) - \sum_{i=1}^{n-1} T(i)T(i - 1)$$

$T(n+1)$ and $T(n)$ overlap for all values $i : 1 \leq i \leq n-1$, therefore subtracting the two sums leaves only the final in the sum for $T(n + 1)$.

$$T(n + 1) - T(n) = T(n)T(n - 1)$$

The values for n can be shifted by setting $n = m - 1$.

$$T(m) - T(m - 1) = T(m - 1)T(m - 2)$$

However, the label m is without meaning, so label $m = n$.

$$T(n) - T(n - 1) = T(n - 1)T(n - 2)$$

Equivalently,

$$T(n) = T(n-1)[1 + T(n-2)]$$

This expression is easily expressed as a single $O(n)$ loop.

```
Array : T
T[0] = 2
T[1] = 2
for  $i \leftarrow 2$  to  $n$  do
  |  $T[i] = T[i-1] * (1 + T[i-2])$ 
end
Output :  $T[n]$ 
```