Homework 10, 4100-01, Winter 2017

I have not received unauthorized aid on this assignment. I understand the answers that I have submitted. The answers submitted have not been directly copied from another source, but instead are written in my own words.

1) Write an algorithm that given an unsorted array returns whether an increasing subsequence of length 3 exists or not in the array. Your algorithm should run in O(n) time complexity and O(1) space complexity.

```
IncSub_EXISTS (A)
    x, y = ∞
    for i = 1 to A.length
        if A[i] < x
            x = A[i]
        if x < A[i] < y
            y = A[i]
        if y < A[i]
            return true;
```

2) Consider the following multithreaded algorithm for performing pairwise multiplication on n-element arrays A[1..n] and B[1..n], storing the multiplications in C[1..n]:

```
MUL-ARRAYS (A, B, C)
    parallel for i = 1 to A.length
        C[i] = A[i] * B[i]
```

Analyze the work, span and parallelism of this algorithm.

Work: n processes
Span: 1 process length because all can be run at the same time with unlimited processors
Parallelism: n/1 = n. Perfect parallelism where reduce the run time for all the processes down to the length of time for 1.

3) Design a dynamic programming algorithm for the version of the knapsack problem in which there are unlimited quantities of copies for each of the n item kinds given. Indicate the time efficiency of the algorithm.

1 Store an n x (K + 1) matrix to contain solutions for all the P(i, k)
2 Fill in the table row by row using the logic:
- if P(n – 1, K) has a solution then P(n, K) has a solution. Store a 0
- y = 1, while n * y < k
  if P(n – 1, K – (k_n * y)) has a solution then P(n, K) has a solution. Store y

- else P(n, K) has no solution.

3 Trace back through the table from (n, K) following that if:

- value = 0, a solution exists with P(i, k) omitted
- value > 0, a solution exists with P(i, k) repeated the stored number of times
- also if value > 0, a solution may exist with P(i, k) omitted if P(i – 1, k) has a solution

In worst case, if the weight of all the items is 1 then each item would be a solution for every capacity from 0 to K. This would be like filling in the table row by row. This would take $\Theta(nK)$ time. Tracing back could also take this amount of time which would mean $\Theta(2nK) = \Theta(nK)$.