Question Recall the dynamic program for longest increasing subsequence (LIS) for an input sequence of n numbers a1,a2,...,an. If L(i) is the length of the LIS that ends in and includes a_i, then L(i)= 1 + max{L(j) : j < i and a_j < a_i}

1. Give pseudocode that turns this formula for L(i) into an algorithm for finding the length of the LIS of the original sequence. Use the ideas of dynamic programming.

   Solution

   ```
   L(1) = 1
   k =0
   for i=2,...,n
    a.  L(i)=1
        for j =1,...,i
            if a_j <a_i
                L(i) = max{L(i),1+ L(j)}
        k = max{k,L(i)}

   return k
   ```

2. What is the running time of your algorithm in terms of n?

   solution $\Theta(n^2)$; being clever, you can implement this algorithm in O(n log n)time.

3. Prove that the formula L(i) = 1+max{ L(j) : j < i and a_j < a_i } correctly computes the LIS that ends in and includes a_i
4. What is the longest increasing subsequence of the following input sequence?

   0 8 4 12 2 10 6 14 1 9 5 13 3 11 7 15

Solution 0, 2, 6, 9, 13, 15 or 0, 4, 6, 9, 11, 15 or ?

Practice questions

1. Modify the dynamic program for the knapsack problem to find a set of items of maximum value whose total weight is exactly the capacity of the knapsack. Note that for a given set of items, there may not be a subset of items whose total weight is exactly the capacity of the knapsack; in this case, your algorithm should correctly say there is no solution.

   You should:

   (a) Define the dynamic programming table.

   (b) Give a recursive formula for an entry in the dynamic programming table.

   (c) Describe in words how to fill the dynamic programming table.

   (d) Give pseudocode for the final algorithm including how to find and return the items in the knapsack.

   Solution

   (a) aT (i,w) is the highest value knapsack using a subset of the items 1,2,3,..., i with total weight exactly equal to w. Let the entry $-\infty$ denote an infeasible knapsack (ie. there is no subset of the items 1,2,3,..., i with weight exactly w.

   (b) $T(i,0) = 0, \forall i=0,\dots,n$

   $T(0,w)= -\infty, \forall w =1,\dots,W$

   $T(i,w) = \max\{T(i- 1,w),T(i- 1,w- w_i)+ v_i \text{ if } w_i \leq w\}$

   (c) Fill in the table either column-wise or row-wise.

   (d) Input n items with item i's weight w_i and value vi and the capacity of the knapsack W.
   initialize p and T as n+1 by W + 1 arrays
   for i=0 ...n
       T(i,0) = 0
   for w =1 ...W
       T(0,w)= $-\infty$
   for i=1 ...n
       for w =1 ...W
           T(i,w)= T(i− 1,w)
           p(i,w)= w if $w_i \leq w$
           if T(i− 1,w− w_i)+ vi >T(i,w)
               T(i,w)= T(i− 1,w− w_i)+ vi
               p(i,w)= w− w_i

   The final value of the knapsack is T(n,W). If this is $-\infty$, there is no solution. Otherwise, the set of items in the best knapsack can be placed in a linked list O by:

   i= n, w = W
   initialize O as an empty linked list
   while i>0
           if p(i,w)= w− w_i % take item i add i to O w = w− w_i
           i= i− 1

2. Give an O(nt) dynamic programming algorithm for the following task:

Input: A list of n positive integers $a_1, a_2, \ldots, a_n$ and a positive integer K. Question: Does some subset of the a_i's add up to K? (You can use each a_i at most once.)

You should:

(a) Define the dynamic programming table.

(b) Give a recursive formula for an entry in the dynamic programming table.

(c) Describe in words how to fill the dynamic programming table.

(d) Give pseudocode for the final algorithm.

(e) Give the running time of your algorithm.

Solution

(a) Let T (i,k) = 1 if there is a subset of the integers $a_1, a_2, \ldots, a_i$ that adds to k and 0 otherwise.

(b) T (i,0)=1 for all i
T (0,k)=0 for all k> 0
$T(i,k) = \max\{T(i-1,k), T(i-1,k-a_i)(\text{if } a_i \leq k)\}$

(c) Column-wise or row-wise.

(d) initialize T as an n+1 × K + 1 array

```
for i=0 ...n
    T(i,0) = 1
for k =1 ...K
    T(0,k)=0
for i=1 ...n
    for k =1 ...K
        T (i,k)= T (i− 1,k)
         if aᵢ ≤ k
             if T (i− 1,k− aᵢ) = 1, T (i,k)=1
```

If T (n,K) = 0, then there is no subset of the a_i's that add up to K. If T (n,K) = 1, then there is a subset of the a_i's that add up to K.

(e) The running time is O(nK). Note that this is not polynomial time.