CS4335 Tutorial8 （Dynamic Programming Approach）

Question 1. Consider the interval with maximum sum problem given in week 6. Design a dynamic programming algorithm with running time O(n) to solve the problem.
Input: An array A[1..n] of n integers (positive or negative).
Problem: find an interval [i, j] such that A[i]+A[i+1]+…+A[j] is maximized.

Solution:
Let d[i]be the cost of the max sum interval ending at position i.
d[i]= max {A[i], d[i-1]+A[i]]}.
We then choose the biggest d[i] for d[1], d[2]…, d[n].

Pseudo codes:

```
for (i=1 to n){
    compute d[i]
    if (d[i] == A[i])
        b[i]=i; // the interval starts at i-th position
    else   b[i]=b[i-1];
   // the starting position of the interval is the same as the previous position.
}// end of for
// Backtracking
 m is the index such that d[m] is the biggest among d[1], d[2]…, d[n];
 return the interval [b[m], m];
```

Question 2. Maximum sum subsequence problem
Input: An array A[1..n] of n integers (positive or negative).
Problem: find a subsequence of A[1...n] such that the total of the subsequence is maximum and if A[i] is in the subsequence then A[i-1] and A[i+1] cannot be in the subsequence.

Solution:
Let d[i] be the cost of the opt solution for the first I letters.
d[0]= d[-1] =0; // assume that am empty subsequence has a total 0
d[i]= max {A[i]+d[i-2], A[i-1]}.
Pseudo codes:

```
for (i=1 to n){
    Compute d[i] using the equation
  If   (d[i] == d[i-1])
     B[i]=0;
  Else
     B[i]=1;
//backtracking
```

```
m=n;
while (m>0){
     if (B[m] ==1){
          print (A[i]);
          m=m-2;
     }
     else m=m-1;
}// end of while
```