

Discussion 7: Dynamic Programming

May 13

Caijie Zhang

<http://www.cs.ucsb.edu/~caijie>

1. Consider the recurrence relation $T(0)=T(1)=2$ and for $n>1$

$$T(n) = \sum_{i=1}^{n-1} T(i) T(i-1), \quad i = 1, 2, 3 \dots n-1.$$

We consider the problem of computing $T(n)$ from n .

- (a) Show that if you implement this recursion directly in say the C programming language, that the program would use exponentially, in n , many arithmetic operations.
- (b) Explain how, by not recomputing the same $T(i)$ value twice, one can obtain an algorithm for this problem that only uses $O(n^2)$ arithmetic operations.
- (c) Give an algorithm for this problem that only uses $O(n)$ arithmetic operations.

Solution:

- (a) The algorithm written in C

```
int T(int n){
    int sum=0;
    if(n==0 || n==1) return 2;
    for(int j=1;j<=n-1;j++)
        sum+=T(j)*T(j-1)
    return sum;
}
```

Let $N(n)$ denote the number of operations required to calculate $T(n)$, from the algorithm we have:

$$\begin{aligned} N(n) &= \sum_{i=1}^{n-1} 2 * N(i) - N(n-1) - N(0) + 2 * (n-1) \\ &\geq N(n-1) + N(n-2) \geq 2 * N(n-2) \\ &\geq 2 * 2 * N(n-4) \geq \dots \geq 2^{n/2} * N(0) = 2^{n/2} \end{aligned}$$

Hence this algorithm is exponential.

- (b) An algorithm in C:

```
T[0] = T[1] = 2;
for(j=2;j<=n;j++)
{
    T[j] = 0;
    for(k=1;k<=j-1;k++)
        T[j] += T[k] * T[k-1];
}
return T[n];
```

$$N(n) = 1+2+\dots+(n-1) = O(n^2).$$

(c) An algorithm is C:

```
T[0] = T[1] = 2
```

```
for(j=2;j<=n;j++)
```

```
    T[j] = T[j-1] + T[j-1]*T[j-2];
```

```
return T[n];
```

It is obvious that this algorithm is $O(n)$.

2. The input to this problem is a sequence S of integers (not necessarily positive). The problem is to find the consecutive subsequence of S with maximum sum. "Consecutive" means that you are not allowed to skip numbers. For example if the input was

12, -14, 1, 23, -6, 22, -34, 13

the output would be 1, 23, -6, 22. Give a linear time algorithm for this problem.

Solution:

We define two functions:

MCS(i) = Maximum Consecutive Sum of the first i integers.

MSS(i) = Maximum Consecutive Sum of the first i integers that uses x_i .

```
for(i=1;i<=n;i++){  
    MSS(i) = MAX(0, MSS(i-1)+xi);  
    MCS(i) = MAX(MCS(i-1), MSS(i-1)+xi)  
}
```