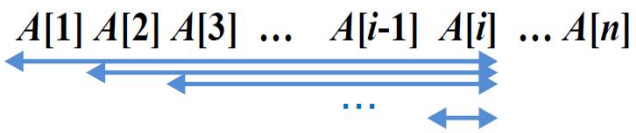
最大字段和问题

最大字段和问题有暴力解法、分值策略和动态规划几种方法，相比较之下，动态规划方法更为高效，利用空间换时间的方法，减少了时间复杂度，在一定数据规模的情况下，动态规划算法独树一帜。在慕课学习里面，对最大子段和动态规划这一块进行了详细讲解。

课程截图

子问题界定：前边界为 1，后边界 i，
$C[i]$ 是 $A[1..i]$中必须包含元素 $A[i]$ 的
向前连续延伸的最大子段和

$$C[i]=\max_{1\le k\le i}\{\sum_{j=k}^{i}A[j]\}$$



最大子段和 $C[i]$

递推方程:

$$C[i]=\max\{C[i-1]+A[i], A[i]\}$$

$i=2, \ldots, n$

$C[1]=A[1]$ 若$A[1]>0$

$C[1]=0$ 否则

解： $\mathrm{OPT}(A)=\max_{1\le i\le n}\{C[i]\}$

问题：求解 A[1,n]的最大字段和，最好给出是哪一子段，A[1,n]由外部文件导入。

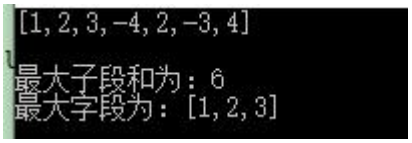
思路：界定子问题为 A[1,i] (i<0<=n)，局部最大子段和 C[i]满足递推方程 $C[i]=\max\{C[i-1]+A[i], A[i]\}$，i=2,....,n C[1]=A[1] (A[1]>0) C[1]=0 (A[1]<=0)

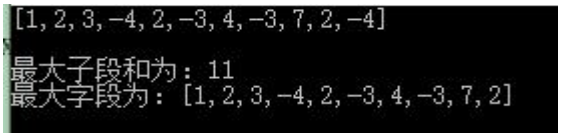
在 i 迭代递增的过程中，设置备忘录记录局部最大子段和和其始末位置，若后面有更大字段和则更新备忘录，整个过程是按照自底向上的结构处理。

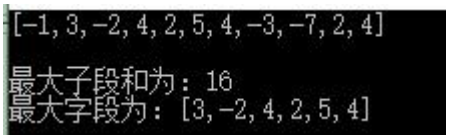
时空复杂度分析：i 从 1 到 n 遍历，对于每个 C[i]的计算，复杂度都是常数级别的，所以总时间复杂度为 O(n)，空间复杂度也为 O(n)

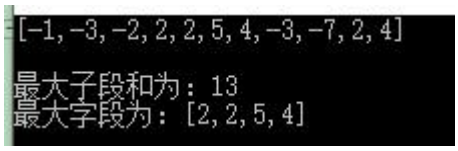
实验结果截图：

设置几组不同的数据进行测试：









体会：体会到了动态规划求解的高效便捷，对于最大字段和问题，优化函数不一定是原问题的优化函数，可以是子问题的优化函数，但是和原问题优化函数有着联系。而且备忘录的设置也很巧妙，可以不用记录所有步骤的局部最大字段和及其始末位置，可以实时更新，覆盖原来的记录，减少了空间开销。对于动态规划，仅仅做这个实验尚且不够，因为在课程里面，最大字段和问题相比较而言属于比较简单的动态规划问题，所以在课余时间，我会尝试将其他的动态规划问题都实现一下。