计算机问题求解 – 论题1–7

- 不同的程序设计方法

课程研讨

- DH第3章第2、3单元

问题1：编译型语言 vs. 解释型语言

- 它们的区别是什么？
  它们各有哪些优缺点？

- 我们生活中哪些语言是解释型的？

问题2：编译器

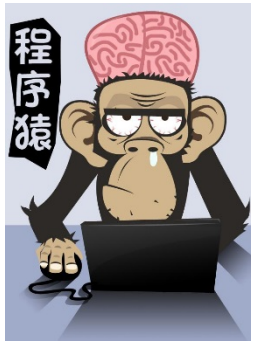
- 为什么一个C语言编译器自身（的绝大部分）可以用C语言来编写？

程序设计语言……

如果你能让整个论坛的人都吵起来，我就答应跟你约会。

（程序猿默默地发了一个帖子：C++是最好的编程语言！）

（论坛迅速炸开了锅，各种吵架……）



服了你了，我们去约会吧。

今天不行，我一定要说服他们，C++必须是最好的语言！

问题3：不同范型的语言

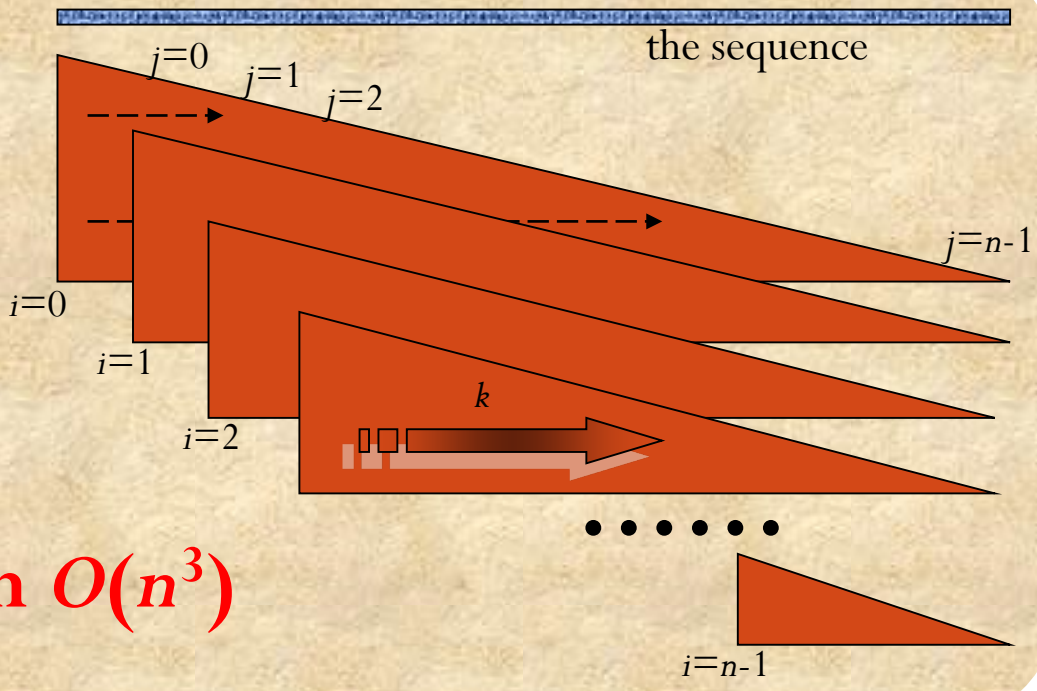
- 你理解这些语言范型了吗？
  它们各有哪些优缺点？
  因此，它们分别适合于哪些应用领域？
  - Imperative
  - Functional
  - Logic
  - Object-oriented
- 未来10年间，你认为哪种范型会成为主导？为什么？

Maximum Subsequence Sum

- The problem: Given a sequence S of integer, find the **largest sum** of a consecutive subsequence of S. (0, if all negative items)
  - An example: -2, 11, -4, 13, -5, -2; the result 20: (11, -4, 13)

A brute-force algorithm:

```
MaxSum = 0;
 for (i = 0; i < N; i++)
  for (j = i; j < N; j++)
  {
   ThisSum = 0;
   for (k = i; k <= j; k++)
   ThisSum += A[k];
   if (ThisSum > MaxSum)
     MaxSum = ThisSum;
  }
 return MaxSum;
```



in $O(n^3)$

More Precise Complexity

The total cost is : $\sum_{i=0}^{n-1}\sum_{j=i}^{n-1}\sum_{k=i}^{j}1$

$$\sum_{k=i}^{j}1 = j-i+1$$

$$\sum_{j=i}^{n-1}(j-i+1) = 1+2+\ldots+(n-i) = \frac{(n-i+1)(n-i)}{2}$$

$$\sum_{i=0}^{n-1}\frac{(n-i+1)(n-i)}{2} = \sum_{i=1}^{n}\frac{(n-i+2)(n-i+1)}{2}$$

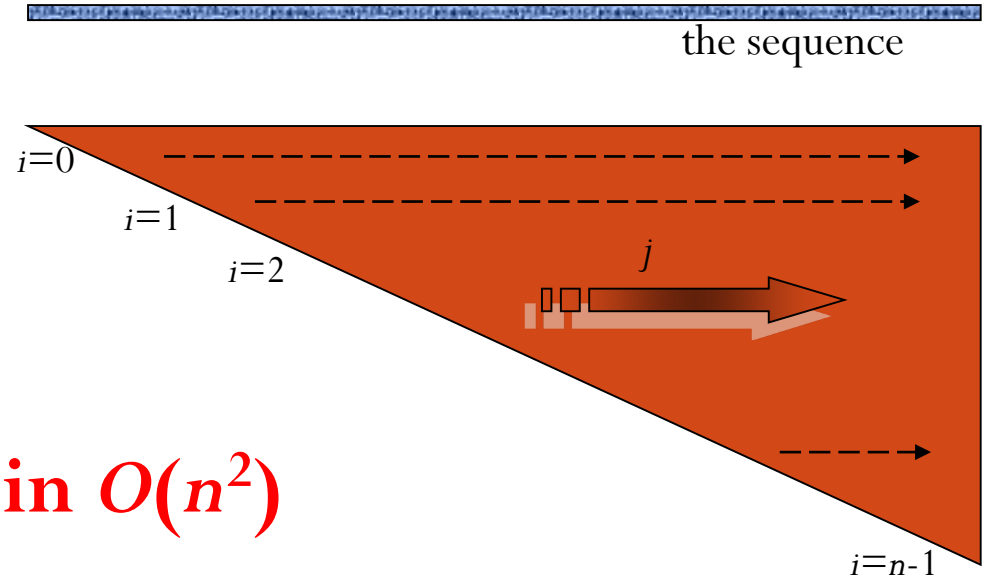
$$= \frac{1}{2}\sum_{i=1}^{n}i^2 - (n+\frac{3}{2})\sum_{i=1}^{n}i + \frac{1}{2}(n^2+3n+2)\sum_{i=1}^{n}1$$

$$= \frac{n^3+3n^2+2n}{6}$$

Decreasing the number of loops

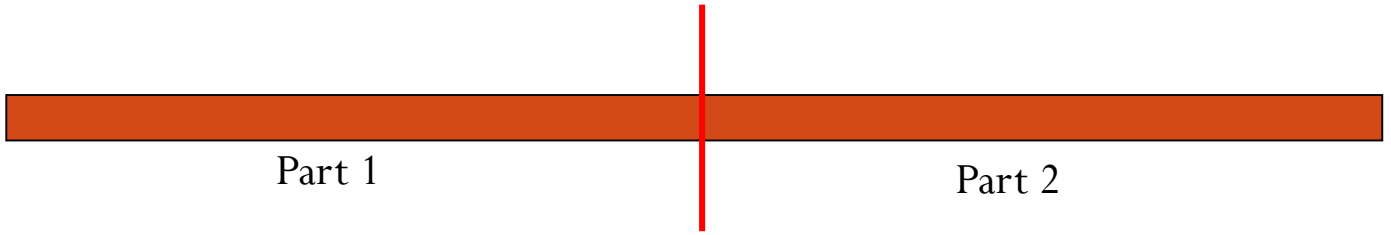
An improved algorithm

```
MaxSum = 0;
 for (i = 0; i < N; i++)
 {
  ThisSum = 0;
  for (j = i; j < N; j++)
  {
   ThisSum += A[j];
   if (ThisSum > MaxSum)
     MaxSum = ThisSum;
  }
 }
 return MaxSum;
```

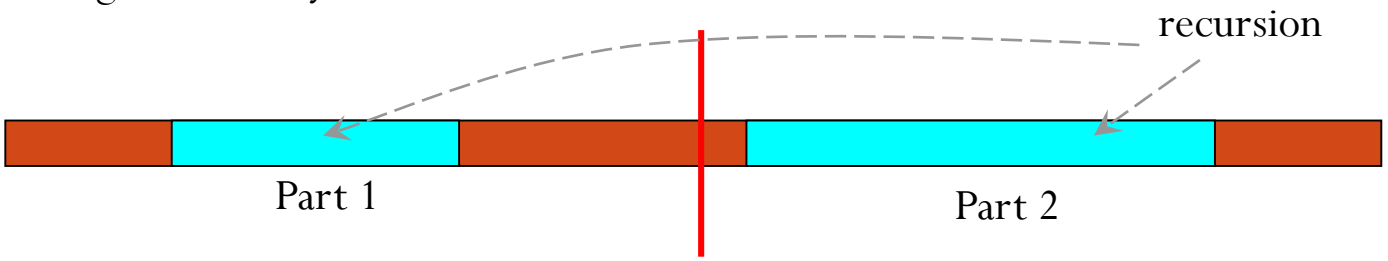


in $O(n^2)$

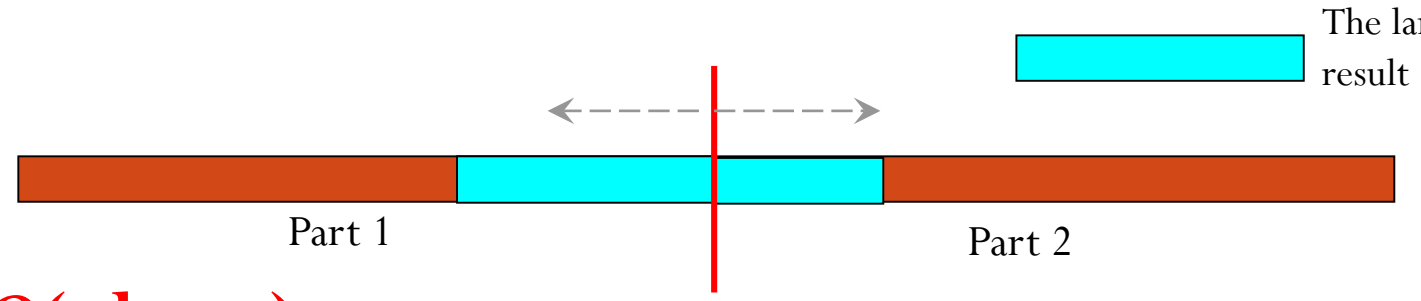
Power of Divide-and-Conquer



the sub with largest sum may be in:



or:



in $O(n \log n)$

Divide-and-Conquer: the Procedure

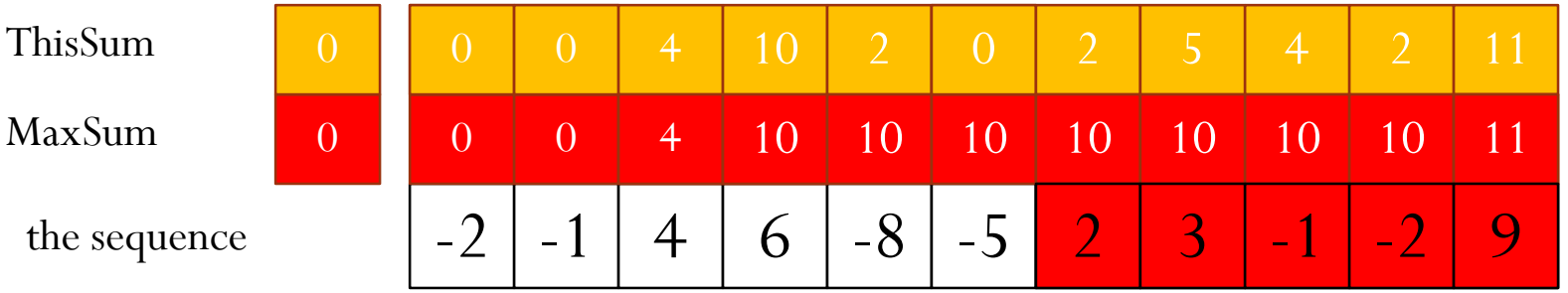
```
Center = (Left + Right) / 2;
 MaxLeftSum = MaxSubSum(A, Left, Center);  MaxRightSum = MaxSubSum(A, Center + 1, Right);

 MaxLeftBorderSum = 0; LeftBorderSum = 0;
 for (i = Center; i >= Left; i--)
 {
  LeftBorderSum += A[i];
  if (LeftBorderSum > MaxLeftBorderSum)    MaxLeftBorderSum = LeftBorderSum;
 }

 MaxRightBorderSum = 0; RightBorderSum = 0;
 for (i = Center + 1; i <= Right; i++)
 {
  RightBorderSum += A[i];
  if (RightBorderSum > MaxRightBorderSum)  MaxRightBorderSum = RightBorderSum;
 }
 return Max3(MaxLeftSum, MaxRightSum, MaxLeftBorderSum + MaxRightBorderSum);
```

Note: this is the core part of the procedure, with base case and wrap omitted.

A Linear Algorithm




```
ThisSum = MaxSum = 0;
 for (j = 0; j < N; j++)
 {
  ThisSum += A[j];
  if (ThisSum > MaxSum)
    MaxSum = ThisSum;
  else if (ThisSum < 0)
    ThisSum = 0;
 }
 return MaxSum;
```

This is an example of "online algorithm"

in $O(n)$

Negative item or subsequence cannot be a prefix of the subsequence we want.