计算机问题求解 – 论题2-3

- 分治法与递归

课程研讨

- TC第4章

问题1：maximum-subarray problem

```
FIND-MAXIMUM-SUBARRAY(A, low, high)
if high == low
    return (low, high, A[low])          // base case: only one element
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) =
        FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right-low, right-high, right-sum) =
        FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) =
        FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum and left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
left-sum = −∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left-sum
        left-sum = sum
        max-left = i
right-sum = −∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right-sum
        right-sum = sum
        max-right = j
return (max-left, max-right, left-sum + right-sum)
```

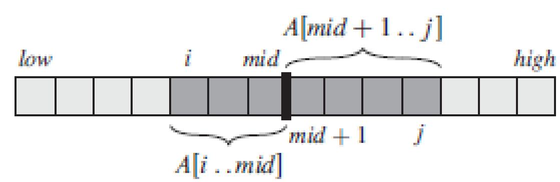
- divide、conquer、combine在这个算法中分别如何体现？
- 为什么这个divide-and-conquer比brute-force快？
  节约了哪些计算？
- 运行时间的递归式是什么？

问题1：maximum-subarray problem

```
FIND-MAXIMUM-SUBARRAY(A, low, high)
if high == low
    return (low, high, A[low])          // base case: only one element
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) =
        FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right-low, right-high, right-sum) =
        FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) =
        FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum and left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

```
FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
left-sum = −∞
sum = 0
for i = mid downto low
    sum = sum + A[i]
    if sum > left-sum
        left-sum = sum
        max-left = i
right-sum = −∞
sum = 0
for j = mid + 1 to high
    sum = sum + A[j]
    if sum > right-sum
        right-sum = sum
        max-right = j
return (max-left, max-right, left-sum + right-sum)
```

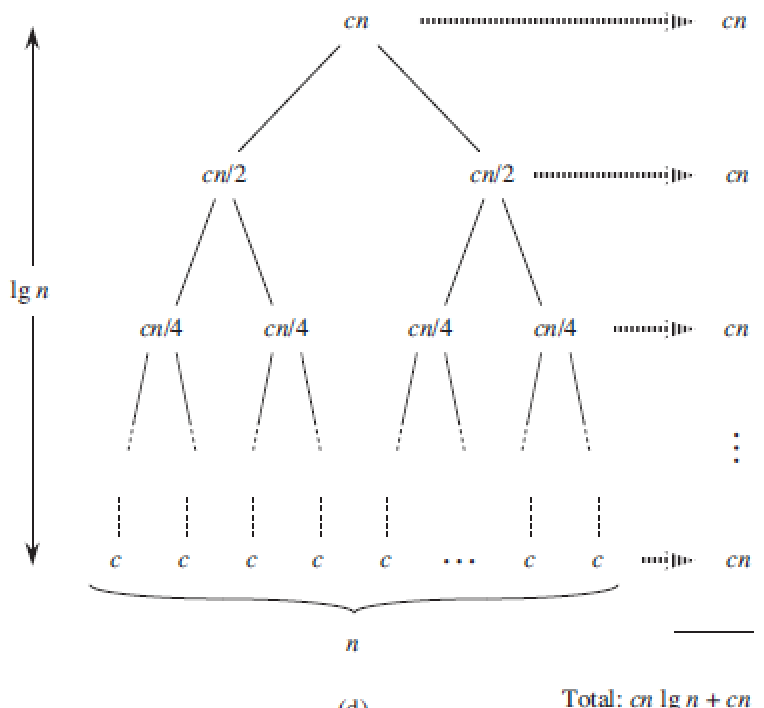
- divide、conquer、combine在这个算法中分别如何体现？
- 为什么这个divide-and-conquer比brute-force快？节约了哪些计算？
- 运行时间的递归式是什么？



问题1：maximum-subarray problem

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 . \end{cases}$$

- 你能画出递归树，并利用递归树来猜测递归式的解吗？



(d)

问题1：maximum-subarray problem (续)

$$T(n)=\begin{cases}\Theta(1) & \text{if } n=1\,,\\ 2T(n/2)+\Theta(n) & \text{if } n>1\,.\end{cases}$$

- 这段基于数学归纳法的证明，你能解释其中的红色标注吗？
  - 目标：$\exists c>0, T(n)\le cn\lg n$
  - 初始：
    - $T(1)=\Theta(1)\le c1\lg 1$ Oops!
    - $T(2)=2\Theta(1)+\Theta(2)\le c2\lg 2$
    - $T(3)=2\Theta(1)+\Theta(3)\le c3\lg 3$
  - 递推：
    - 假设：$T\left(\frac{n}{2}\right)\le c\frac{n}{2}\lg\frac{n}{2}$
    - 推导：$T(n)\le 2c\frac{n}{2}\lg\frac{n}{2}+\Theta(n)=cn\lg\frac{n}{2}+\Theta(n)=cn\lg n-cn\lg 2+\Theta(n)$

      $\le cn\lg n-cn+dn=cn\lg n-(c-d)n\le cn\lg n$

d是什么？ 最后一步的理由？

问题1：maximum-subarray problem (续)

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 , \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 . \end{cases}$$

- 你能利用主定理来解这个递归式吗？

Let $a \geq 1$ and $b > 1$ be constants, let $f(n)$ be a function, and let $T(n)$ be defined on the nonnegative integers by the recurrence

$$T(n) = aT(n/b) + f(n) ,$$

where we interpret n/b to mean either $\lfloor n/b \rfloor$ or $\lceil n/b \rceil$. Then $T(n)$ has the following asymptotic bounds:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \lg n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and if $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ■

A Linear Algorithm

ThisSum	0	0	0	4	10	2	0	2	5	4	2	11
MaxSum	0	0	0	4	10	10	10	10	10	10	10	11
the sequence		-2	-1	4	6	-8	-5	2	3	-1	-2	9

```
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;
```

j

This is an example of "online algorithm"

in $O(n)$

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

问题2：substitution method

- $T(n) = T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + 1$

- 尝试 $T(n) \le cn$

$$\begin{aligned} T(n) &\le c\lfloor n/2 \rfloor + c\lceil n/2 \rceil + 1 \\ &= cn + 1 , \end{aligned}$$

- 尝试 $T(n) \le cn - d$

$$\begin{aligned} T(n) &\le (c\lfloor n/2 \rfloor - d) + (c\lceil n/2 \rceil - d) + 1 \\ &= cn - 2d + 1 \\ &\le cn - d , \end{aligned}$$

- 教材希望通过这个例子教我们什么？你理解这段证明了吗？

问题2： substitution method (续)

- $T(n) = 2T\left(\lfloor\sqrt{n}\rfloor\right) + \lg n$

- $m = \lg n \quad \Rightarrow \quad T(2^m) = 2T(2^{m/2}) + m$
- $S(m) = T(2^m) \quad \Rightarrow \quad S(m) = 2S(m/2) + m$

  $\Rightarrow \quad S(m) = O(m \lg m)$

  $\Rightarrow \quad T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$

- 教材希望通过这个例子教我们什么？你理解这段证明了吗？

问题3：recursion-tree method

Argue that the solution to the recurrence $T(n) = T(n/3) + T(2n/3) + cn$, where c is a constant, is $\Omega(n \lg n)$ by appealing to a recursion tree.

问题4： master method

- 你能用主定理解这些递归式吗？

  a. $T(n) = 2T(n/4) + 1$.

  b. $T(n) = 2T(n/4) + \sqrt{n}$.

  c. $T(n) = 2T(n/4) + n$.

  d. $T(n) = 2T(n/4) + n^2$.

  e. $T(n) = 2T(n/4) + \sqrt{n} \lg n$

Gap in Master Theory

- T(n) = 9T(n/3) + O(n²)

$$E = \frac{\lg b}{\lg c} = \frac{\lg 9}{\lg 3} = 2 \qquad f(n) \in O(n^2)$$

Consider the worst case,

$$f(n) \in \Theta(n^2)$$

Case 2,

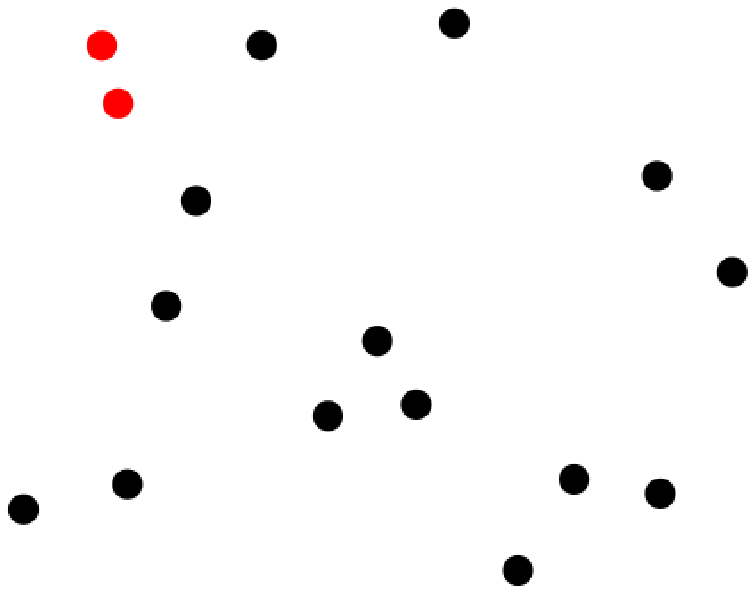
$$T(n) \in \Theta(n^2 \lg n)$$

Generally,

$$T(n) \in O(n^2 \lg n)$$

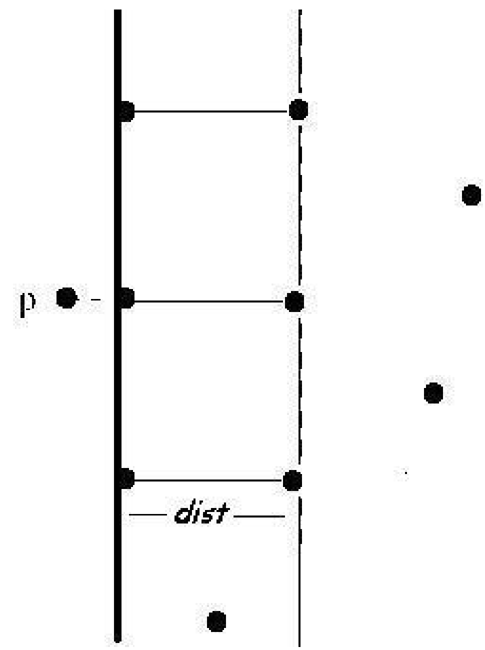
问题5： divide-and-conquer

- Closest pair of points problem



问题5： divide-and-conquer (续)

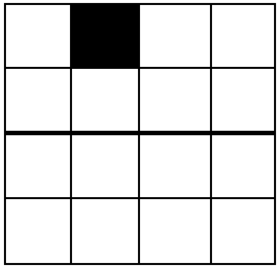
- For each point p to the left of the dividing line we have to compare the distances to the points that lie in the rectangle of dimensions (dist, 2·dist) to the right of the dividing line.
- This rectangle can contain at most six points with pairwise distances at least d_{Rmin}.
- Therefore, it is sufficient to compute at most 6n left-right distances.
- $T(n)=2T(n/2)+O(n)$

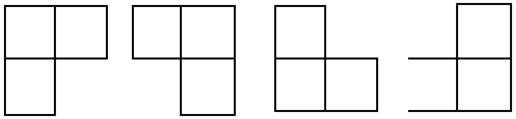


dist= $\min(d_{Lmin}, d_{Rmin})$

问题5：divide-and-conquer (续)

- 在一个2^k*2^k的棋盘中，有某个格子已被覆盖了，你能否设计一个分治算法，使用一些L型骨牌恰覆盖棋盘上的其它所有格子？
- 你能分析你给出的这个算法的运行时间吗？





问题5：divide-and-conquer (续)

- 在一个2^k*2^k的棋盘中，有某个格子已被覆盖了，你能否设计一个分治算法，使用一些L型骨牌恰覆盖棋盘上的其它所有格子？
- 你能分析你给出的这个算法的运行时间吗？