

第10讲 受限资源约束下的算法—排序问题求解示例

李玉华

华中科技大学计算机学院

ldcliyuhua@hust.edu.cn

第10讲-受限资源约束下的算法—排序问题求解示例

2

- 一、为什么要研究排序算法
- 二、内排序基本算法
- 三、PageRank算法--网页排序



为什么要研究排序算法

3

(1)什么是排序问题?

对一组对象按照某种规则进行有序排列。通常是把一组**对象**整理成按**关键字**递增(或递减)的排列，**关键字**是指对象的一个用于排序的特性。

例如：

- 对一组“**人**”，按“**年龄**”或“**身高**”排序；
- 对一组“**商品**”，按“**价格**”排序；
- 对一组“**网页**”，按“**重要度**”排序；
- 对一组“**词汇**”，按“**首字母**”字典序排序。
-

为什么要研究排序算法

4

(2)从结构化数据表查找看排序问题的重要性

未排序

学号	姓名	成绩
120300101	李鹏	88
120300105	张伟	66
120300107	闫宁	95
120300102	王刚	79
120300103	李宁	94
120300106	徐月	85
120300108	杜岩	44
120300104	赵凯	69
120300109	江海	77
120300110	周峰	73

数据表记录数: n

查找成绩为80分的所有同学?

【算法A：未排序数据查找算法】

Start of algorithm

Step 1. 从数据表的第1条记录开始，直到其最后一条记录为止，读取每一条记录，做**Step 2**。

Step 2. 对每一条记录，判断成绩是否等于给定的分数：如果是，则输出；如果不是，则不输出。

End of algorithm

算法效率：读取并处理所有记录，即 n 条记录

为什么要研究排序算法

5

(2)从结构化数据表查找看排序问题的重要性

查找成绩为80分的所有同学？

已排序

学号	姓名	成绩
120300107	闫宁	95
120300103	李宁	94
120300101	李鹏	88
120300106	徐月	85
120300102	王刚	79
120300109	江海	77
120300110	周峰	73
120300104	赵凯	69
120300105	张伟	66
120300108	杜岩	44

数据表记录数: n

【算法B：已排序数据查找算法】

Start of algorithm

Step 1. 从数据表的第1条记录开始，直到其最后一条记录为止，读取每一条记录，做**Step 2**和**Step 3**步。

Step 2. 对每一条记录，判断成绩是否等于给定的分数：如果等于，则输出；如果不等于，则不输出。

Step 3. 判断该条记录的成绩是否小于给定的分数：如果不是，则继续；否则，退出循环，算法结束。

End of algorithm

算法效率：读取并处理部分记录，即 $\leq n$ 条记录

为什么要研究排序算法

6

(2)从结构化数据表查找看排序问题的重要性

已排序

学号	姓名	成绩
120300107	闫宁	95
120300103	李宁	94
120300101	李鹏	88
120300106	徐月	85
120300102	王刚	79
120300109	江海	77
120300110	周峰	73
120300104	赵凯	69
120300105	张伟	66
120300108	杜岩	44

数据表记录数: n

查找成绩为80分的所有同学?

【算法C：已排序数据查找算法】

Start of algorithm

Step 1. 假设数据表的最大记录数是 n ，待查询区间的起始记录位置Start为1，终止记录位置Finish为 n ；

Step 2. 计算中间记录位置 $I=(Start+Finish)/2$ ，读取第 I 条记录。

Step 3. 判断第 I 条记录成绩是否大于给定查找分数：

(1)如果是小于，调整 $Finish = I-1$ ，如果 $Start > Finish$ 则结束，否则继续做

Step 2；(2)如果是大于，调整 $Start = I+1$ ，如果 $Start > Finish$ 则结束，否则继续做Step 2；(3)如果是等于，则输出，继续读取 I 周围所有的成绩与给定查找条件相等的记录并输出，直到所有相等记录查询输出完毕则算法结束。

End of algorithm

•算法效率：除极端情况外读取并处理 $\leq n/2$ 条记录

为什么要研究排序算法

7

(3)从结构化数据表的统计看排序问题的重要性

学号	姓名	成绩
120300107	闫宁	95
120300103	李宁	94
120300101	李鹏	88
120300106	徐月	85
120300102	王刚	79
120300109	江海	77
120300110	周峰	73
120300104	赵凯	69
120300105	张伟	66
120300108	杜岩	44

???

- 统计各个分数段的人数
- 统计每个同学的平均分数
- 统计每门课的平均分数

学号	姓名	成绩
120300101	李鹏	88
120300105	张伟	66
120300107	闫宁	95
120300102	王刚	79
120300103	李宁	94
120300106	徐月	85
120300108	杜岩	44
120300104	赵凯	69
120300109	江海	77
120300110	周峰	73

•算法效率：需要读取并处理???条记录才能完成呢？



为什么要研究排序算法

8

(4)排序算法的重要性

学号	姓名	成绩
120300107	闫宁	88
120300103	李宁	66
120300101	李鹏	95
120300106	徐月	79
120300102	王刚	94
120300109	江海	85
120300110	周峰	44
120300104	赵凯	69
120300105	张伟	77
120300108	杜岩	73

排序算法是计算学科中很重要的算法

当对数据集合需要多遍处理时，先排序-好

排序算法是很多复杂算法的基础

•算法效率：需要读取并处理???条记录才能完成呢？

为什么要研究排序算法

9

(5)非结构化文档检索



关键词查询

包含<关键词>的文档
是哪一个? 有多少个?

怎样找?

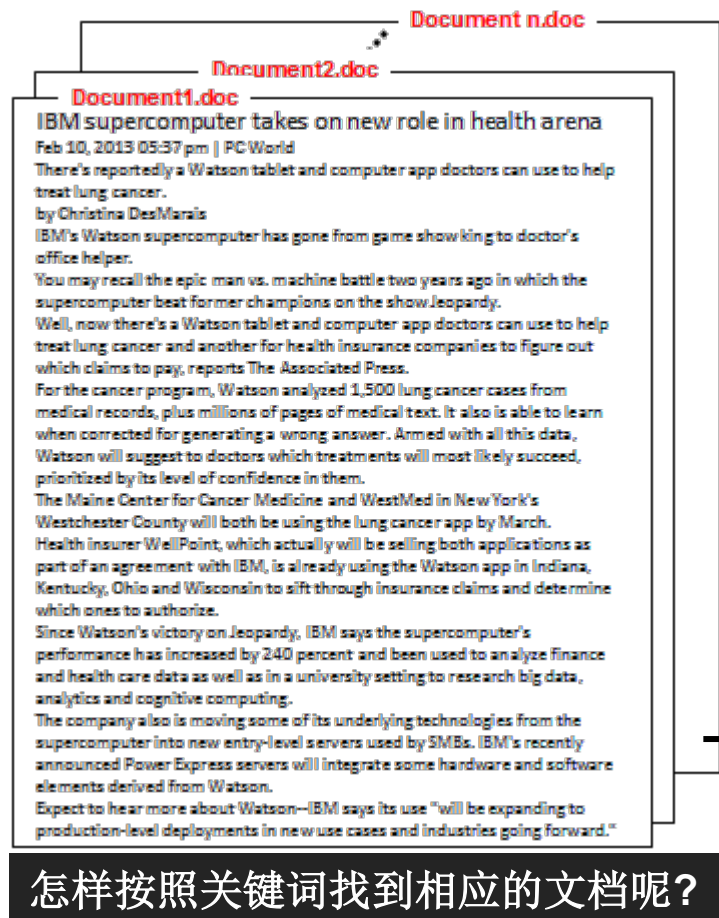
怎样快
速地找?

怎样按照关键词找到相应的文档呢?

为什么要研究排序算法

10

(6)索引与倒排索引--需要排序?



正排：一个文档包含了哪些词汇?

#Doc1, { Word1, Word2, ... }

倒排：一个词汇包含在哪些文档中

Word1, { #Doc1, #Doc2, ... }

怎样建立索引?

关键词查询

关键词索引表---倒排索引

关键词, (#所在文档, 出现次数, <出现位置,...>)

health, (#Document1.doc, 1次, <8>),

(#Document3.doc, 3次, <10,62, 182>)

IBM, (#Document1.doc, 2次, <1,10>),

(#Document2.doc, 5次, <1,10,100,24>)

Supercomputer, (#Document1.doc, 3次, <1,10,100>)

Watson, (#Document1.doc, 1次, <11>)

(#Document4.doc, 3次, <15,8>)

排序 or 不排序?

怎样利用索引快速地找?

为什么要研究排序算法

11

(7)关键词的提取--需要排序?

能否自动找出文档中的关键词?

哪些是关键词?

排序 or 不排序?

IBM supercomputer takes on new role in health arena
...
IBM's Watson supercomputer has gone from game show king to doctor's office helper.
You may recall the epic man vs. machine battle two years ago in which the supercomputer beat former champions on the show Jeopardy.
...

文档

关键词, 出现次数, <出现位置,...>

Supercomputer, 3次, {2, 12, 38}

IBM, 2次, {1, 10}

health, 1次, {8}

Watson, 1次, {11}

...

关键词, (#所在文档, 出现次数, <出现位置,...>)

health, (#Document1.doc, 1次, <8>),
(#Document3.doc, 3次, <10, 62, 182>)

IBM, (#Document1.doc, 2次, <1, 10>),
(#Document2.doc, 5次, <1, 10, 100, 240, 500>)

Supercomputer, (#Document1.doc, 3次, <2, 12, 38>)

Watson, (#Document1.doc, 1次, <11>),
(#Document4.doc, 3次, <15, 81, 202>)

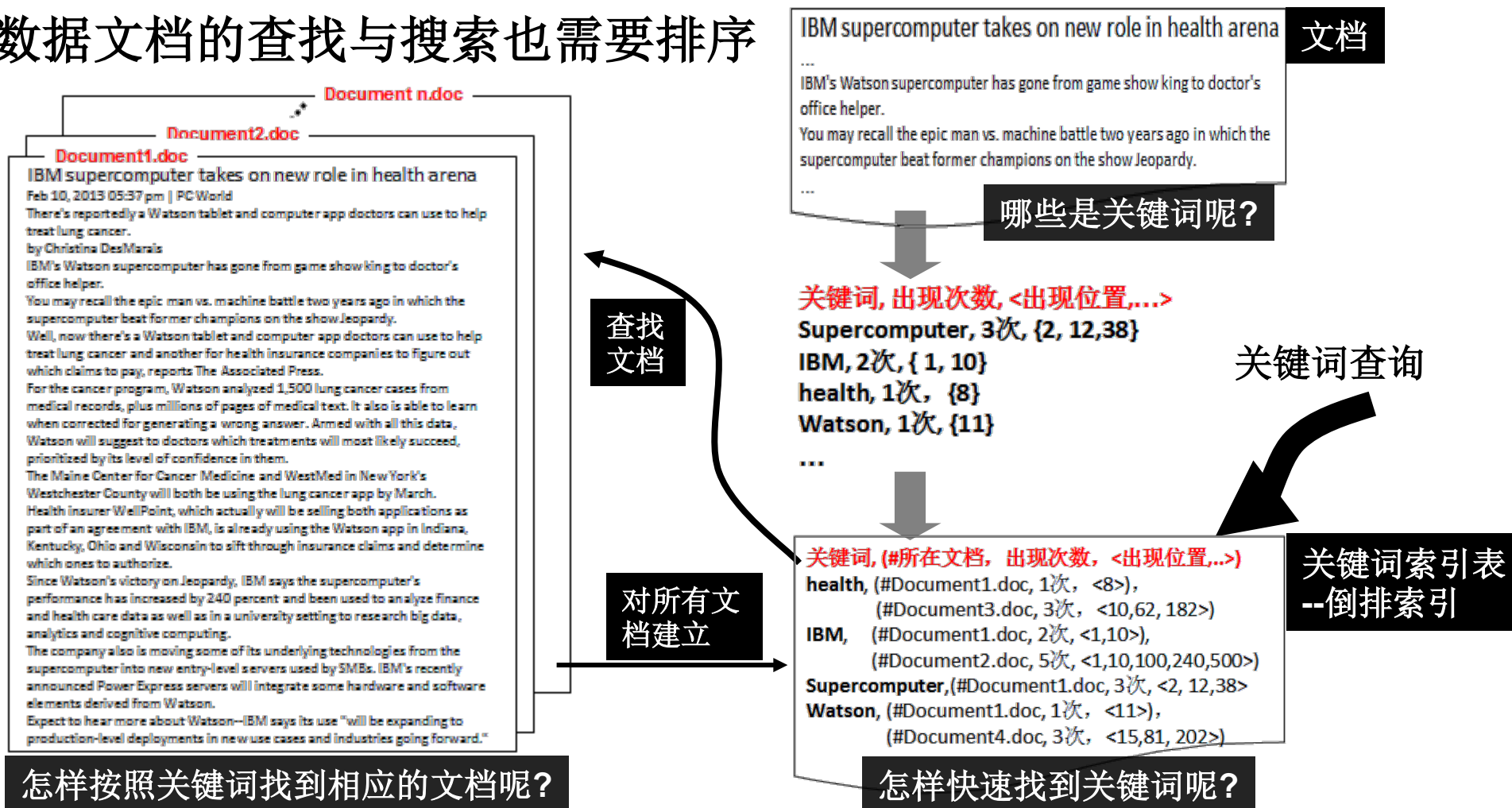
关键词索引表
--倒排索引

为什么要研究排序算法

12

(8)从非结构化文档检索看排序问题的重要性

非结构化数据文档的查找与搜索也需要排序



为什么要研究排序算法

13

(8)从非结构化文档检索看排序问题的重要性

非结构化数据文档的查找与搜索也需要排序



第10讲-受限资源约束下的算法—排序问题求解示例

14

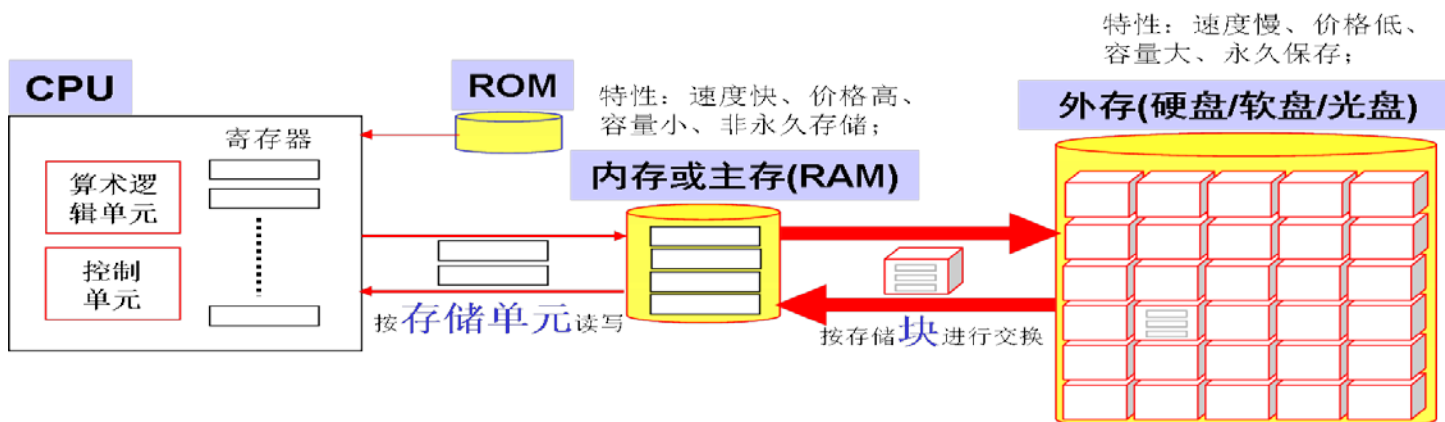
- 一、为什么要研究排序算法
- 二、内排序基本算法
- 三、PageRank算法--网页排序

内排序基本算法

15

受限资源约束下的算法--内排序与外排序问题

- 内排序问题**:待排序的数据可一次性地装入内存中，即排序者可以完整地看到和操纵所有数据，使用数组或其他数据结构便可进行统一的排序处理的排序问题；
- 外排序问题**:待排序的数据保存在磁盘上，不能一次性装入内存，即排序者不能一次完整地看到和操纵所有数据，需要将数据分批装入内存分批处理的排序问题；



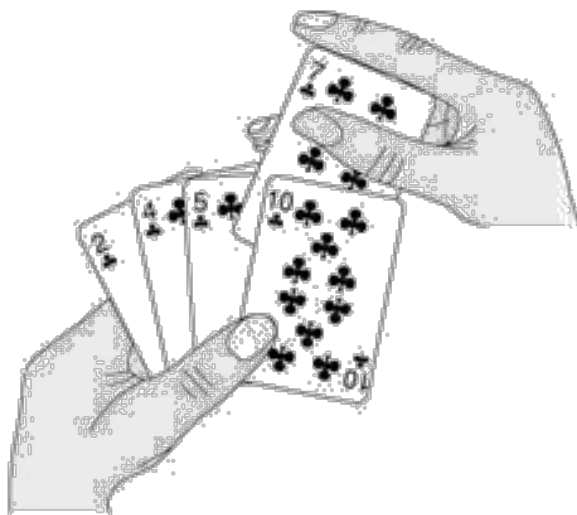
问题类比：某教师要对学生按身高排序。教师只能在房间(相当于内存)中对学生进行排序，假设房间仅能容纳100人，那么对于小于100人的学生排序便属于内排序问题。而对于大于100人，如1000人的学生排序，学生并不能都进入房间，而只能在操场(相当于磁盘)等候，轮流进入房间，这样的排序便属于外排序问题。

内排序基本算法

16

(1)【插入排序法】：基本思想

类似于打扑克牌时，一边抓牌，一边理牌的过程：
每抓一张牌就把它插入到适当的位置；
牌抓完了，也理完了。
---这种策略被称为插入排序

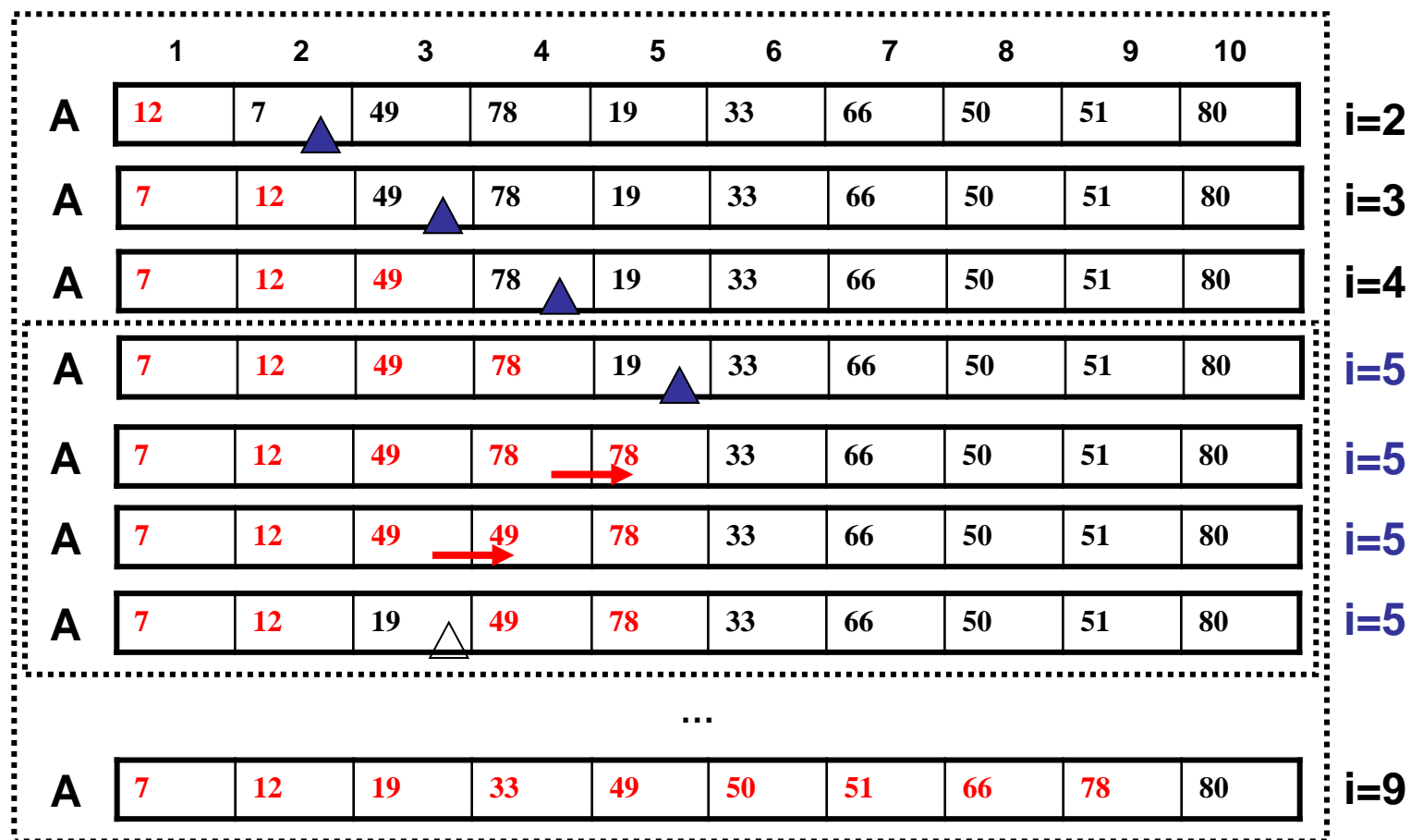




内排序基本算法

17

(1)【插入排序法】：过程模拟



插入排序:递增排序示意. 其中三角形左侧为已排好序的元素, 其右侧为未排序的元素, 实心三角形本身为待插入的元素. 图中示意了为待排序元素19腾挪空间的过程, 由箭头示意. 空心三角形表示新插入的元素

内排序基本算法

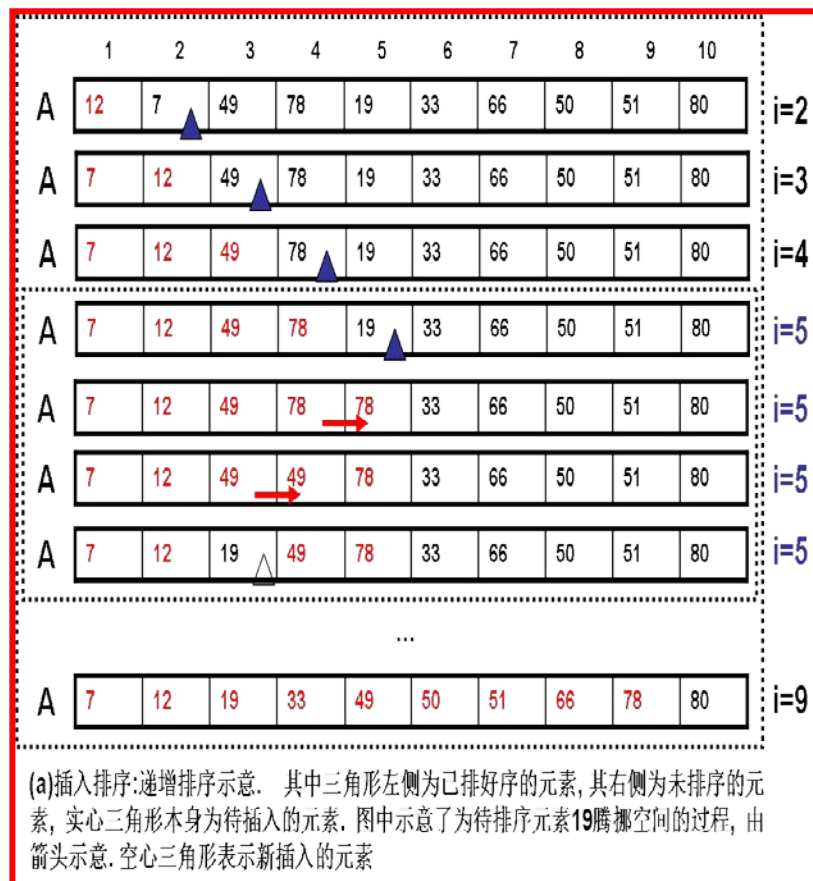
18

(1)【插入排序法】：算法表达

INSERTION-SORT(A)

1. *for* $i=2$ *to* N
2. { $key = A[i]$;
3. $j = i-1$;
4. *While* ($j > 0$ and $A[j] > key$) *do*
5. { $A[j+1] = A[j]$;
6. $j = j-1$; }
7. $A[j+1] = key$;
8. }

$O(N^2)$





内排序基本算法

19

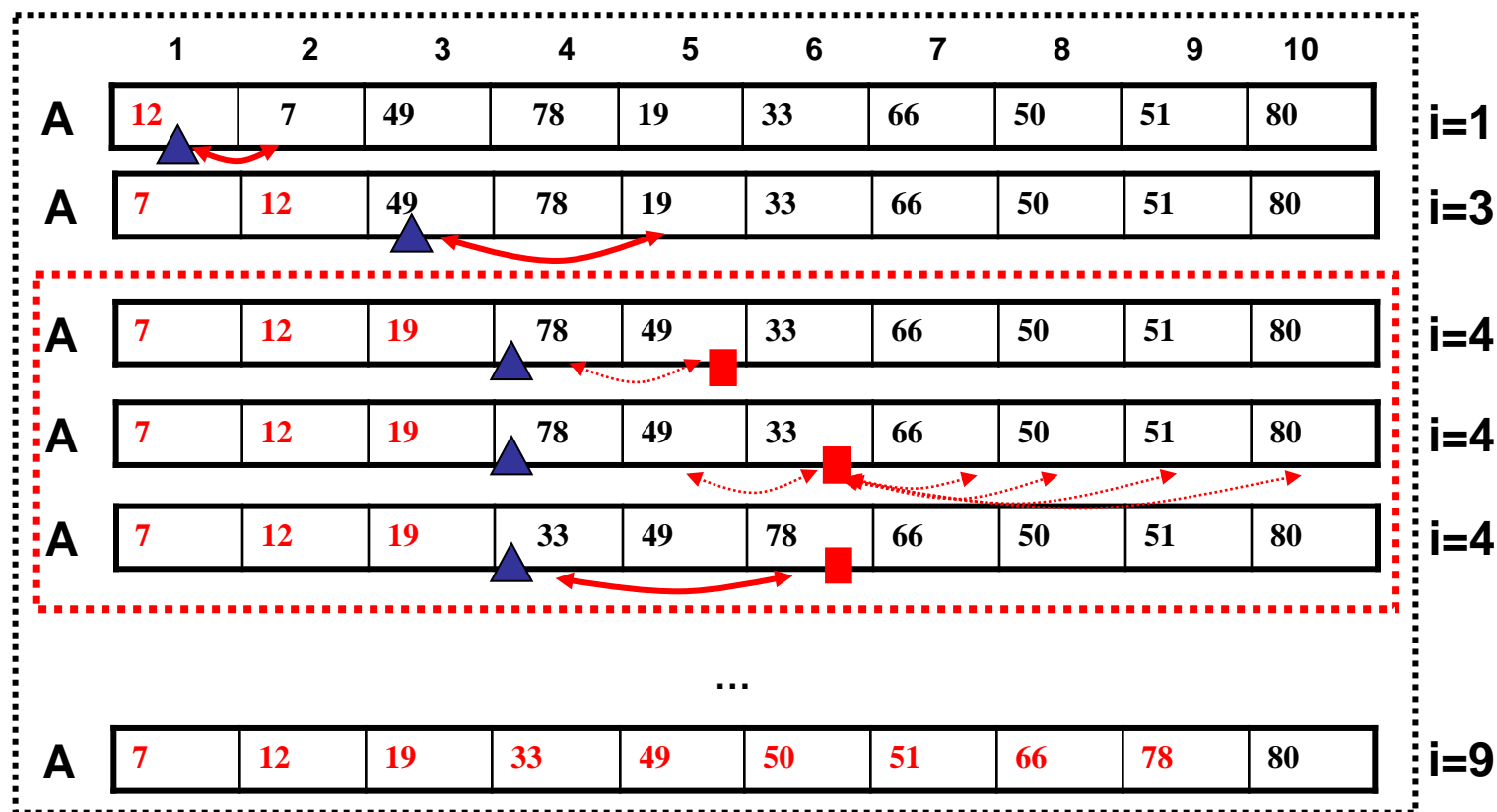
(2)【简单选择法】：基本思想

首先在所有数组元素中找出最小值的元素，放在A[1]中；
接着在不包含A[1]的余下数组元素中再找出最小值元素，放置在A[2]中；
如此下去，一直到最后一个元素。
这一排序策略被称为简单选择法排序。

内排序基本算法

20

(2)【简单选择法】：过程模拟



(b)选择排序:递增排序示意.

其中三角形代表本轮要找的最小元素应在的位置, 方形代表本轮次至今为止所找到的最小元素所在位置, 三角形左侧为已排好序的元素, 三角形右侧的每一元素依次和方形位置元素比较. 实线双向箭头代表两个元素交换. 虚线双向箭头代表两个元素需要比较

内排序基本算法

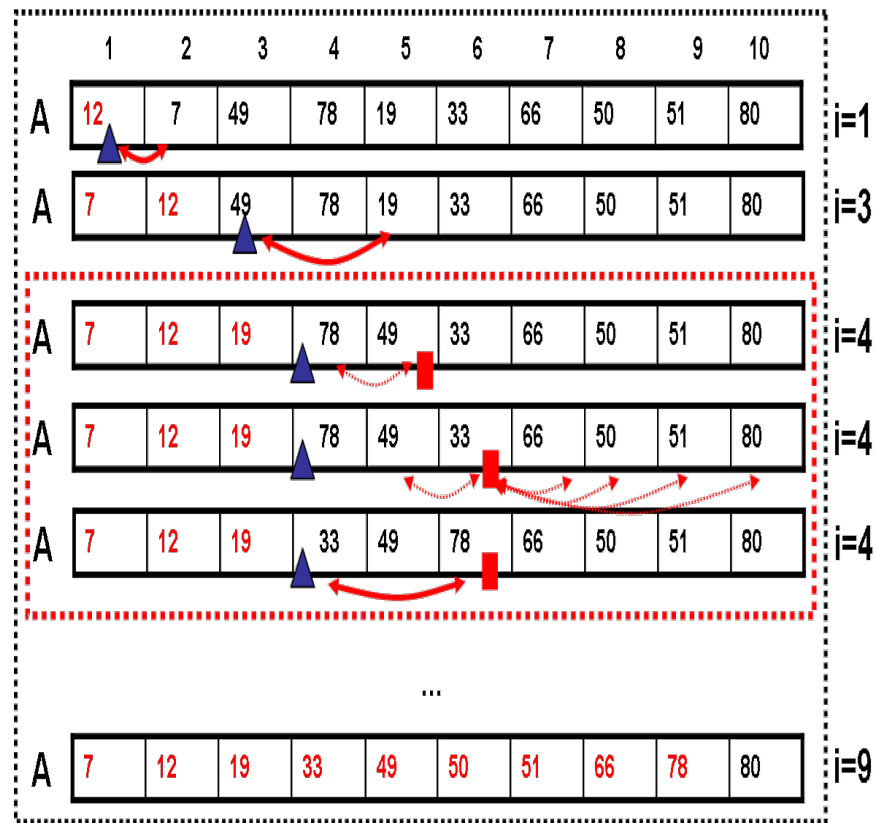
(2) 【简单选择法】：算法表达

SELECTION-SORT(A)

```
1. for  $i=1$  to  $N-1$ 
2. {    $k=i$ ;
3.     for  $j=i+1$  to  $N$ 
4.       { if  $A[j]<A[k]$  then  $k=j$ ; }
5.     if  $k \neq i$  then
6.       {
7.          $temp=A[k]$ ;
8.          $A[k]=A[i]$ ;
9.          $A[i]=temp$ ;
10.      }
11. }
```

K：本轮要找的最小元素的位置序号

$O(N^2)$



(b)选择排序:递增排序示意.

其中三角形代表本轮要找的最小元素应在的位置,方形代表本轮次至今为止所找到的最小元素所在位置,三角形左侧为已排好序的元素,三角形右侧的每一元素依次和方形位置元素比较.实线双向箭头代表两个元素交换.虚线双向箭头代表两个元素需要比较

内排序基本算法

22

(3)【冒泡排序法】：基本思想？

一个轮次一个轮次的处理。

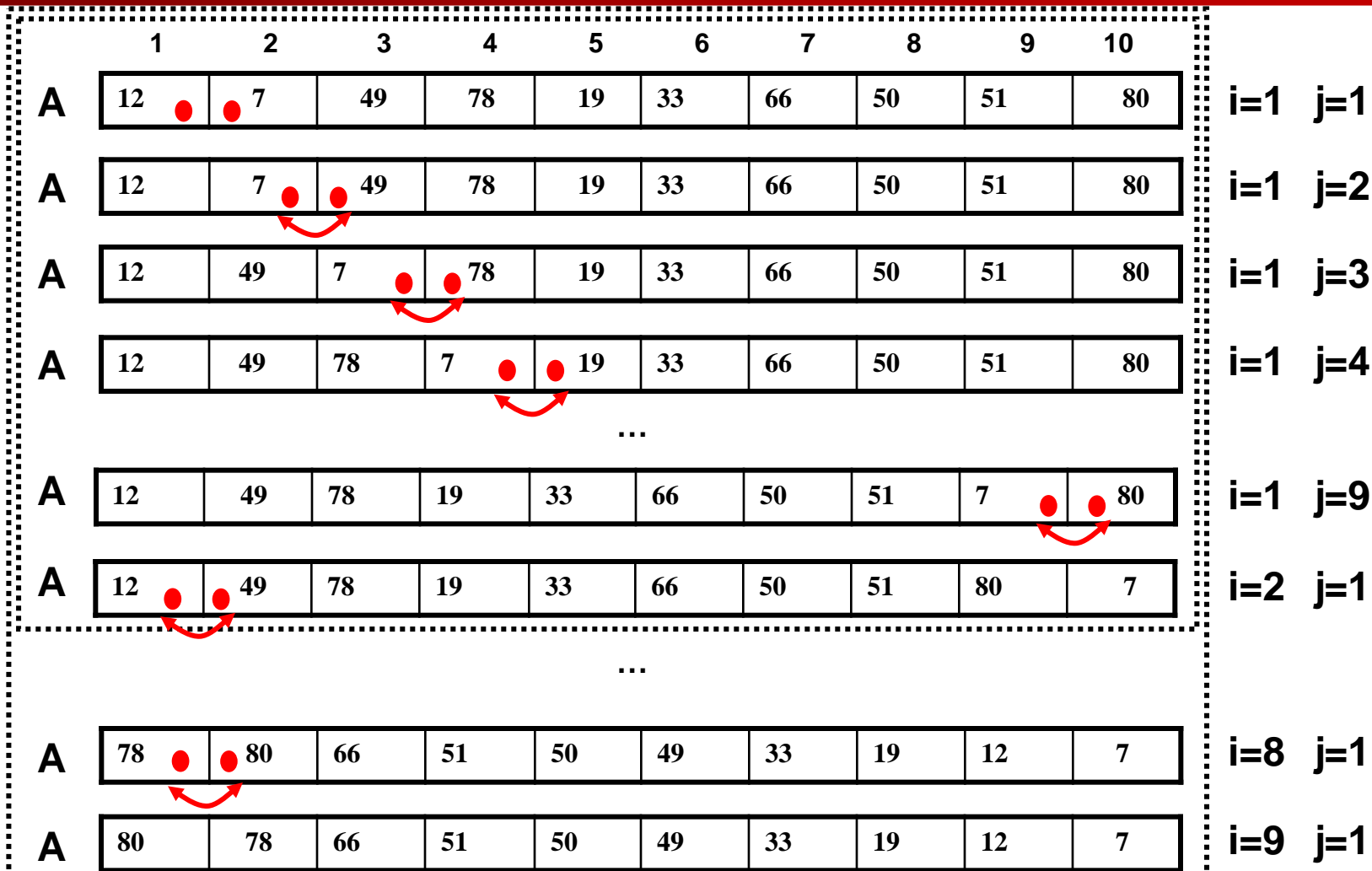
在每一轮次中依次对待排序数组元素中相邻的两个元素进行比较，将大的放前，小的放后--递减排序(或者将小的放前，大的放后--递增排序)。当没有交换时，则数据已被排好序。



内排序基本算法

23

(3) 【冒泡排序法】：过程模拟



(c)冒泡排序:递减
排序示意, 其中
小圆点代表本轮
本次比较的两个
元素, 双向弧线箭
头代表两个元素
要相互交换

内排序基本算法

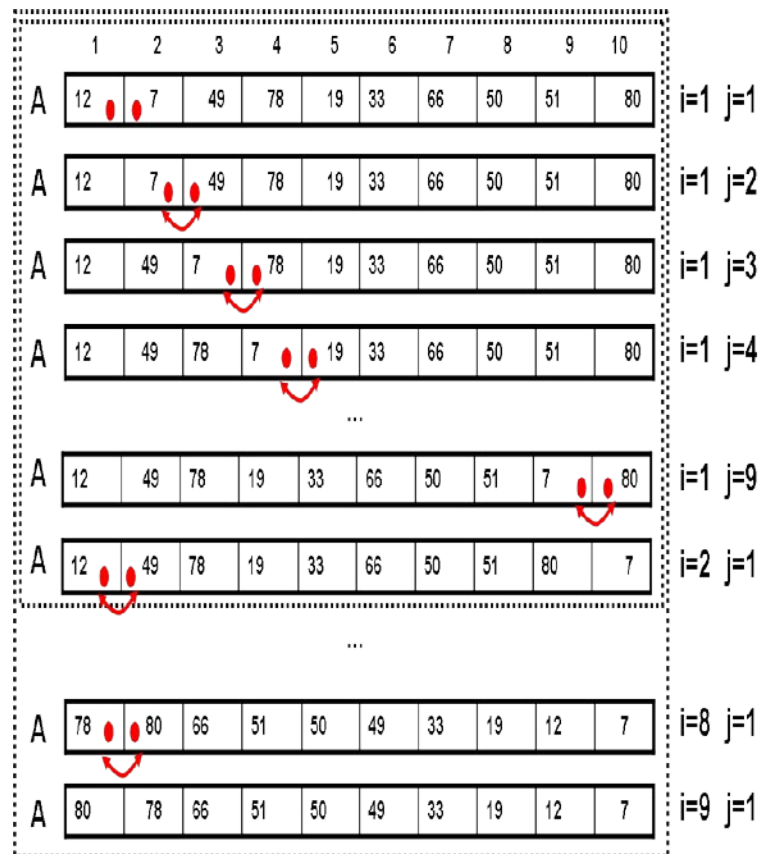
24

(3) 【冒泡排序法】：算法表达

BUBBLE-SORT(A)

1. **for** $i=1$ **to** $N-1$
2. { $haschange=false$;
3. **for** $j=1$ **to** $N-i$
4. { **if** $A[j]>A[j+1]$ **then**
5. $temp=A[j]$;
6. $A[j]=A[j+1]$;
7. $A[j+1]=temp$;
8. $haschange=true$;
9. }
10. }
11. **if** ($haschange == false$) **then break**;
12. }

$O(N^2)$



(c)冒泡排序:递减排序示意

其中小圆点代表本轮本次比较的两个元素,双向弧线箭头代表两个元素要相互交换

三种内排序算法的比较


- 时间复杂度: $O(N^2)$
- 空间复杂度: $O(1)$
- 执行时间上有差异, 比较、交换次数越少的算法越快

算法实例：二分查找

26

- 二分查找算法：对于一个已经排序好的序列(比如升序)在查找所要查找的元素 k 时,首先与序列中间的元素进行比较,如果 k 大于这个元素,就在当前序列的后半部分继续查找,如果 k 小于这个元素,就在当前序列的前半部分继续查找,直到找到相同的元素,或者所查找的序列范围为空为止。
- 定义数组 $A[i]$, $i=1, \dots, n$, 给定了 k 值, 另外设置变量 low , $high$, mid

1	2	3		12
10	20	30	...	120

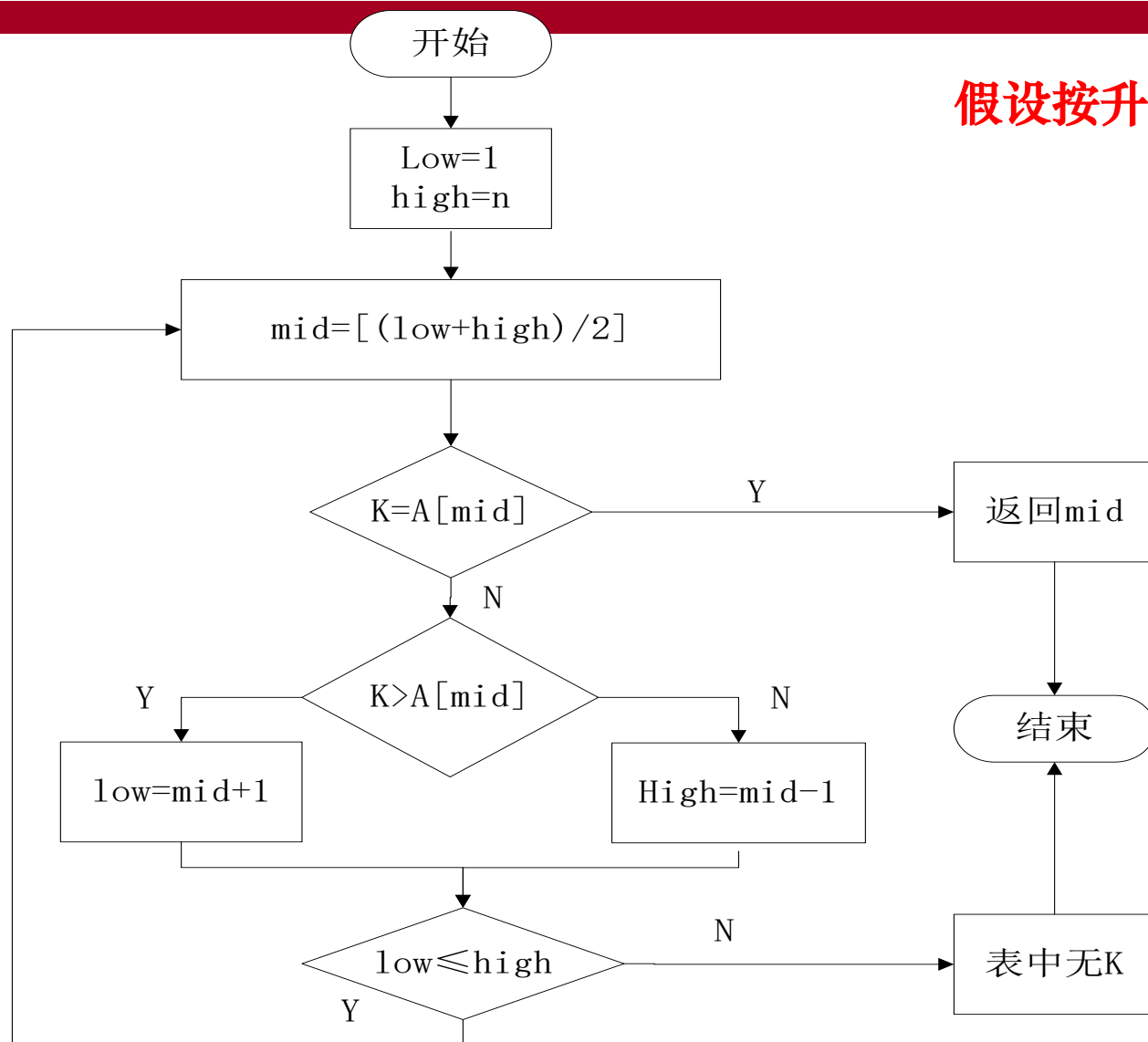


$i=n/2=6$ $A[i]=60$
则 $k=50$ 是在左边or右边?

算法实例：二分查找程序流程图

28

假设按升序排好序





二分查找实例

29

1	2	3	4	5	6	7	8	9	10	11
5	13	19	21	37	56	64	75	80	88	92

现在 执行k=21， k=85的过程， 给出结论及各变量的值

low	high	mid
-----	------	-----

查找k=21过程

low	high	mid
1	11	6
1	5	3
4	5	4

查找k=85过程

low	high	mid
1	11	6
7	11	9
10	11	10
10	9	

二分折半查找过程实例

[05 13 19 21 37 56 64 75 80 88 92]

↑

[05 13 19 21 37] 56 64 75 80 88 92

↑

05 13 19 [21 37] 56 64 75 80 88 92

↑

(a) 查找K=21的过程 (3次比较后查找成功)

[05 13 19 21 37 56 64 75 80 88 92]

↑

05 13 19 21 37 56 [64 75 80 88 92]

↑

05 13 19 21 37 56 64 75 80 [88 92]

↑

05 13 19 21 37 56 64 75 80] [88 92

(b) 查找K=85的过程 (3次比较后查找失败)

第10讲-受限资源约束下的算法—排序问题求解示例

31

- 一、为什么要研究排序算法
- 二、内排序基本算法
- 三、PageRank算法--网页排序

PageRank算法--网页排序问题及思想

32

(1)网页排序问题：搜索引擎？



4,540,000条检索记录



1,210,000条检索记录

怎样把最重要的检索记录显示给用户？

PageRank算法--网页排序问题及思想

33

(2)PageRank是什么？网页又是什么？

问题背景--网页

●PageRank是计算网页重要度的一种方法



网页重要吗?---网页重要度

<标记>文本</标记>

Our Product Information

Our Product Information

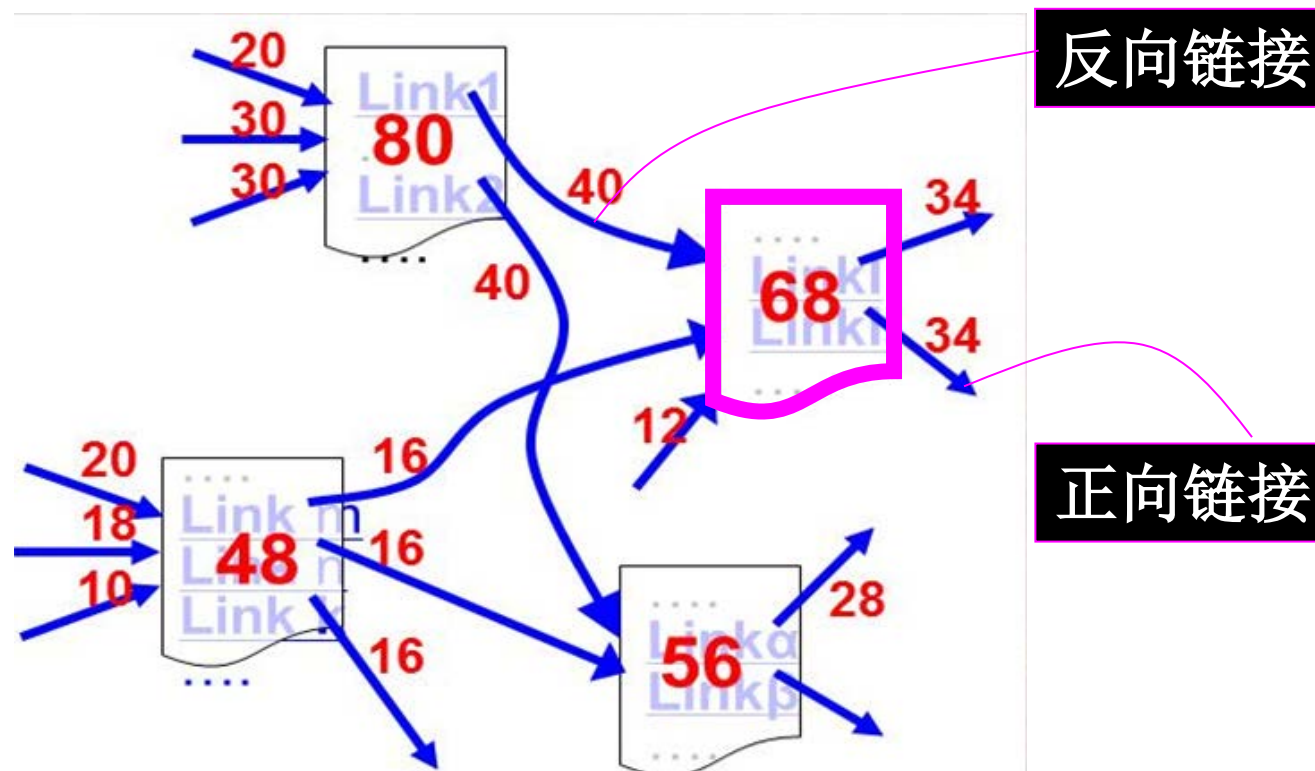
PageRank算法--网页排序问题及思想

34

(3)正向链接与反向链接?

网页重要度问题的抽象

基于这些信息，
如何计算网页
重要度？

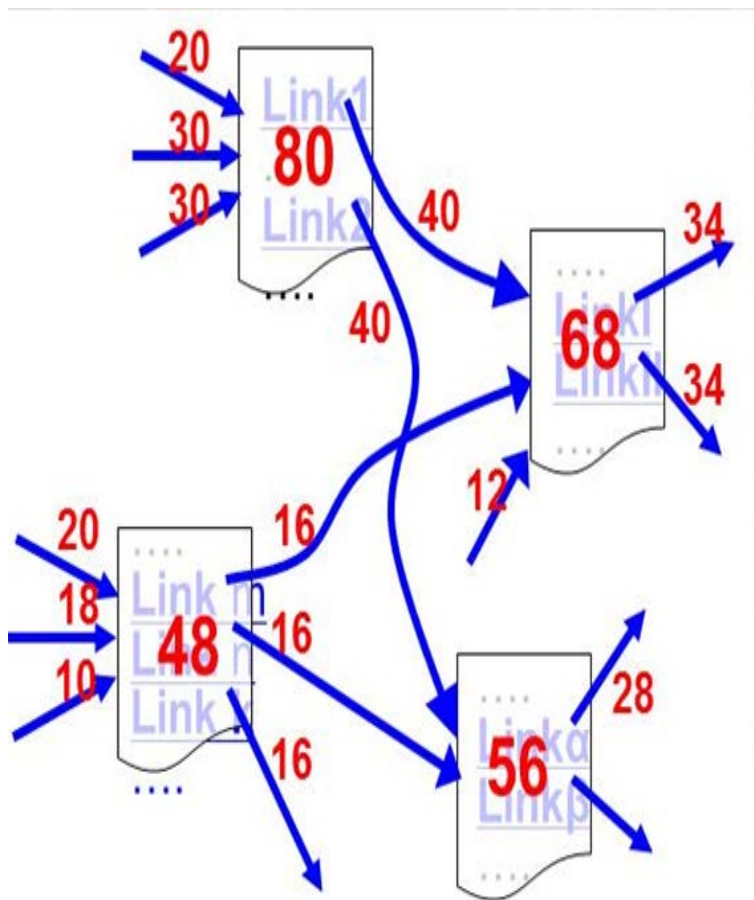


一个网页的正向链接是另一个网页的反向链接

PageRank算法--网页排序问题及思想

35

(4)PageRank的基本思想?



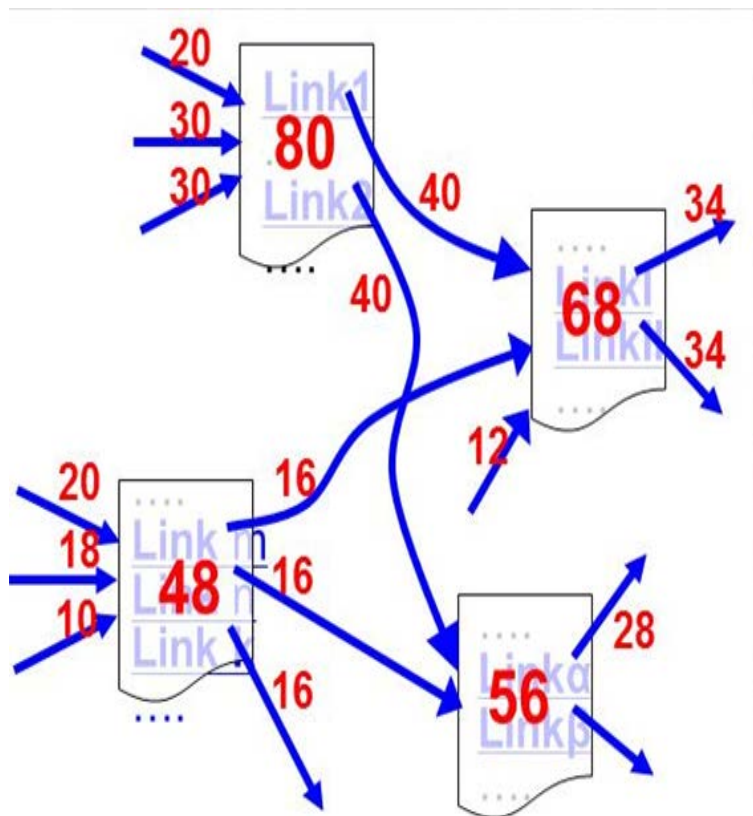
关于网页的基本观点

- 网页的反向链接数越多是否越重要呢?
- 重要度越高的反向链接是否越重要呢?
- 正向链接数越多, 是否其对链接的网页而言, 重要度会降低呢?

PageRank算法--网页排序问题及思想

36

(4)PageRank的基本思想?



- 一个网页的重要度等于其所有反向链接的加权和，即：反向链接权值 z_i ，网页重要度 R ，则

$$R = \sum z_i \text{ (for 所有反向链接 } i \text{)}。$$

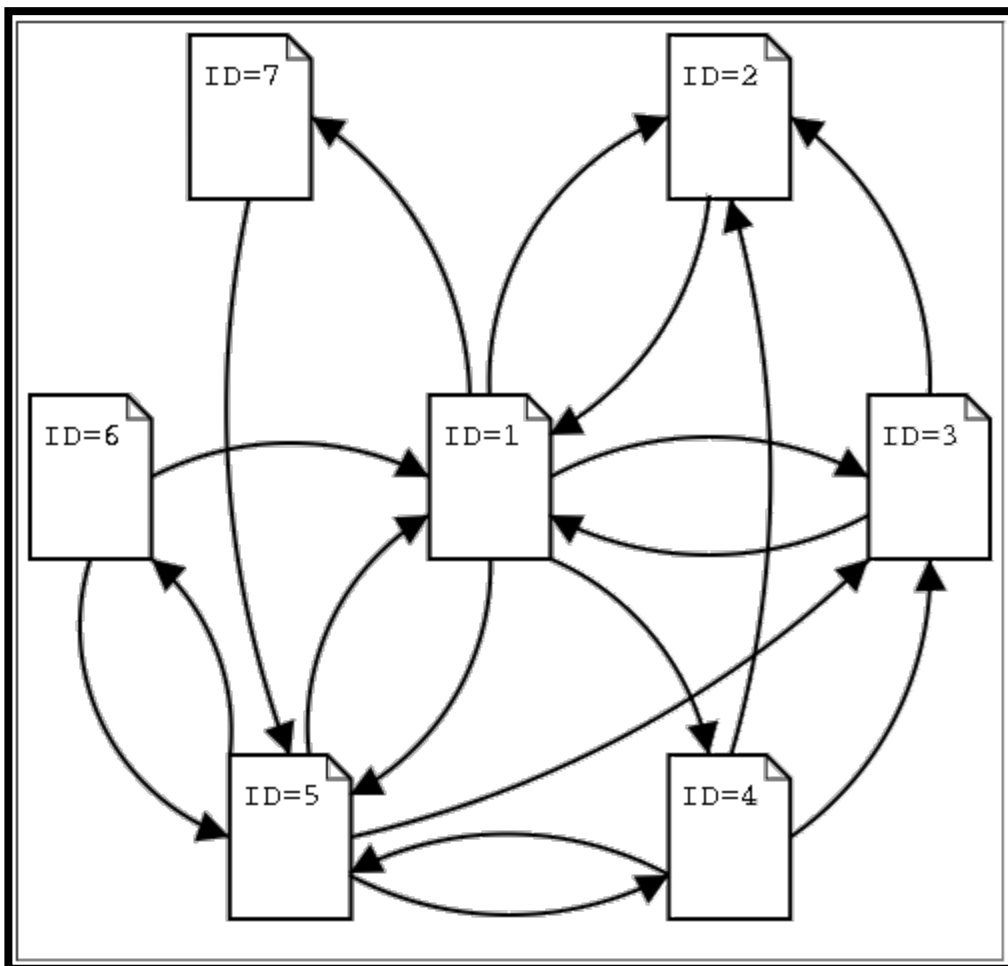
- 一个正向链接的权值等于网页的重要度除以其正向链接数，即：网页重要度 R ，其正向链接数 m ，则其每一个正向链接的权值 $z = R/m$ 。

怎样计算网页的重要度呢？

PageRank算法--网页排序问题的表达与建模

37

(1)问题的数学建模：抽象



链接源页面 ID	链接目标页面 ID
1	2, 3, 4, 5, 7
2	1
3	1, 2
4	2, 3, 5
5	1, 3, 4, 6
6	1, 5
7	5

PageRank算法--网页排序问题的表达与建模

38

(2)问题的数学建模：邻接矩阵

行*i*, 列*j* 均是网页编号

$$a_{ij} = \begin{cases} 1 & \text{if (网页} i \text{存在有指向网页} j \text{的链接)} \\ 0 & \text{if (网页} i \text{没有指向网页} j \text{的链接)} \end{cases}$$

正向链接

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

反向链接

反向链接

$$A^T = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

正向链接

链接源页面 ID	链接目标页面 ID
1	2,3,4,5,7
2	1
3	1,2
4	2,3,5
5	1,3,4,6
6	1,5
7	5

PageRank算法--网页排序问题的表达与建模

39

(3)正向链接的权值矩阵---转移概率?

$$A^T = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

反向链接

邻接矩阵

$$M = \begin{bmatrix} 0 & 1 & 1/2 & 0 & 1/4 & 1/2 & 0 \\ 1/5 & 0 & 1/2 & 1/3 & 0 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

反向链接的权值

转移概率矩阵

$$R^T = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{bmatrix}$$

网页重要度向量

PageRank算法--网页排序问题的表达与建模

40

(3)矩阵乘法与反向链接的加权和?

网页*i*的重要度为 R_i ，各网页重要度的向量 R ，记为： $R = (R_1, R_2, \dots, R_n)^T$

转移概率矩阵M

第n-1次
的网页
重要度

第n次
的网页
重要度

$$\begin{bmatrix} 0 & 1 & 1/2 & 0 & 1/4 & 1/2 & 0 \\ 1/5 & 0 & 1/2 & 1/3 & 0 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{bmatrix}^{(n-1)} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{bmatrix}^{(n)}$$

矩阵乘法

$$R_i^{(n)} = \sum_j (M[i][j] * R_j^{(n-1)})$$

PageRank算法--网页重要度的迭代计算方法

41

(1)网页重要度的迭代计算方法?

网页*i*的重要度为 R_i ，各网页重要度的向量 R ，记为： $R = (R_1, R_2, \dots, R_n)^T$

迭代计算

$$R_i^{(1)} = \sum_j (M[i][j] * R_j^{(0)})$$

$$R_i^{(2)} = \sum_j (M[i][j] * R_j^{(1)})$$

... ..

$$R_i^{(n)} = \sum_j (M[i][j] * R_j^{(n-1)})$$

$$R_i^{(n)} = R_i^{(n-1)} \quad ???$$

转移概率矩阵M

0	1	1/2	0	1/4	1/2	0
1/5	0	1/2	1/3	0	0	0
1/5	0	0	1/3	1/4	0	0
1/5	0	0	0	1/4	0	0
1/5	0	0	1/3	0	1/2	1
0	0	0	0	1/4	0	0
1/5	0	0	0	0	0	0

$\begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{bmatrix}^{(n-1)}$

\times

$\begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{bmatrix}^{(n)}$

第n-1次的网页重要度

第n次的网页重要度

矩阵乘法

R的初始值是多少呢？从哪一个 R_i 开始计算呢？

PageRank算法--网页重要度的迭代计算方法

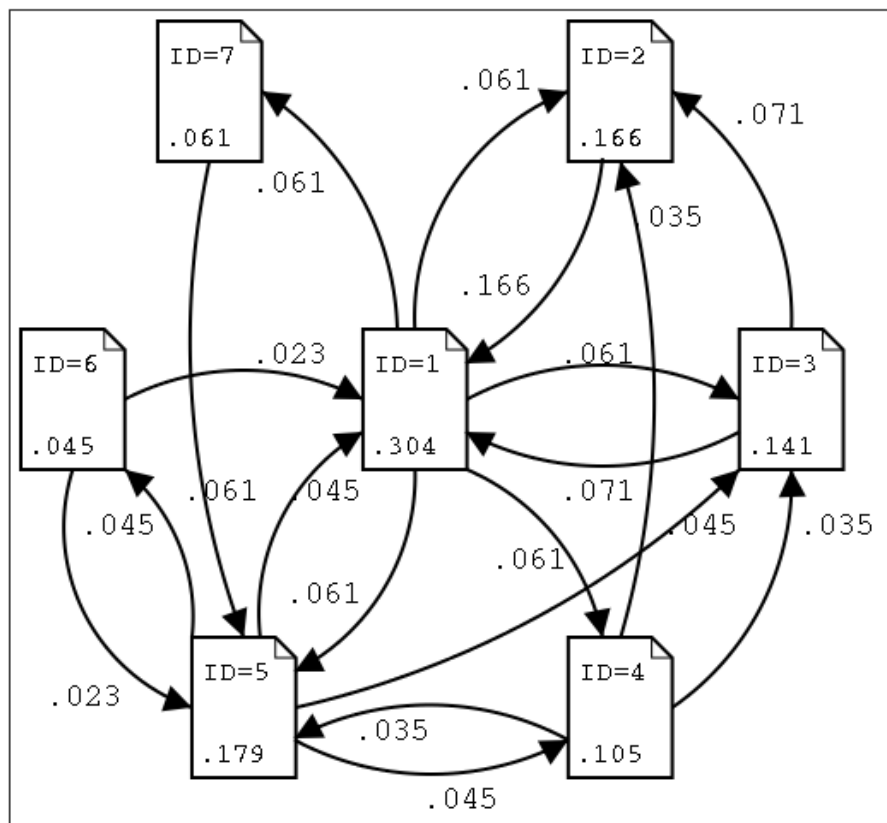
(2)PageRank的计算结果分析?

1号 vs. 5号

2号 vs. 3号

6号 vs. 7号

名次	PageRank	所评价的文件 ID	发出链接 ID (正向链接)	被链接 ID (反向链接)
1	0.304	1	2,3,4,5,7	2,3,5,6
2	0.179	5	1,3,4,6	1,4,6,7
3	0.166	2	1	1,3,4
4	0.141	3	1,2	1,4,5
5	0.105	4	2,3,5	1,5
6	0.061	7	5	1
7	0.045	6	1,5	5



PageRank算法--PageRank与数学及算法总结

43

(1)PageRank计算 vs. 数学的特征方程?

网页重要度的迭代计算

$$\begin{bmatrix} 0 & 1 & 1/2 & 0 & 1/4 & 1/2 & 0 \\ 1/5 & 0 & 1/2 & 1/3 & 0 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \times \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{bmatrix}^{(n-1)} = \begin{bmatrix} R_1 \\ R_2 \\ R_3 \\ R_4 \\ R_5 \\ R_6 \\ R_7 \end{bmatrix}^{(n)}$$

迭代计算网页重要度

$$R^{(0)} = (R_1^{(0)}, R_2^{(0)}, \dots, R_n^{(0)})^T$$

$$R^{(1)} = cMR^{(0)}$$

$$R^{(2)} = cMR^{(1)}$$

...

$R = R^{(n)} = R^{(n-1)}$, 当R不发生变化时,
即收敛时则为所求

$$MR = \frac{1}{c} R$$

对 N 阶方阵A(转移概率矩阵),
满足:

$$Ax = \lambda x$$

的数 λ 称为 A 的特征值, 称 x
为属于 λ 的特征向量。

通过数学学习求解方法

PageRank算法--PageRank与数学及算法总结

44

数学的语义：特征方程

求解思想：求稳定性

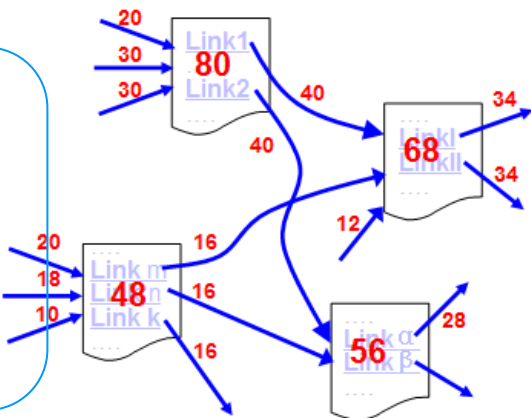
表达成数学：0,1矩阵
权值矩阵—转移概率矩阵

从问题语义挖掘求解思想：
反向链接数越多越重要；
反向链接有权值；
反向链接的权值确定：网页
重要度按其正向链接的个数
进行分配。

网页链接：正向链接与反向链接

网页排序：网
页重要度计算

$$Ax = \lambda x$$



网页重要度计算：PageRank

网页 i 的重要度为 R_i ，各网页重要度的向量 R ，记为：

$$R = (R_1, R_2, \dots, R_n)^T$$

需要迭代计算，第 j 次迭代计算得到的 R 的结果记为 $R^{(j)}$ 。

R 的初始可设置为任意的值，记为： $R^{(0)} = (R_1^{(0)}, R_2^{(0)}, \dots, R_n^{(0)})^T$

$$R^{(1)} = cMR^{(0)}$$

$$R^{(2)} = cMR^{(1)}$$

... ..

$$R^{(n)} = cMR^{(n-1)}$$

$R = R^{(n)} = R^{(n-1)}$ ----收敛状态暨稳定状态

$$M = \begin{bmatrix} 0 & 1 & 1/2 & 0 & 1/4 & 1/2 & 0 \\ 1/5 & 0 & 1/2 & 1/3 & 0 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 1/3 & 0 & 1/2 & 1 \\ 0 & 0 & 0 & 0 & 1/4 & 0 & 0 \\ 1/5 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \text{PageRank} = \begin{bmatrix} 0.303514 \\ 0.166134 \\ 0.140575 \\ 0.105431 \\ 0.178914 \\ 0.044728 \\ 0.060703 \end{bmatrix}$$

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}, \quad A^T = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$