第四章　算法与复杂性

4.1 排序问题及其算法

哈尔滨工业大学 （深圳）
计算机学院

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。
- …… ……

4.1.1 排序问题--结构化数据表查找问题
(2)为什么要研究排序问题？

结构化数据表的查找问题

查找成绩为80分的所有同学?

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

【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

4.1.1 排序问题--结构化数据表查找问题

(2)为什么要研究排序问题？

结构化数据表的查找与统计需要排序

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<=n条记录

4.1.1 排序问题--结构化数据表查找问题
(2)为什么要研究排序问题？

结构化数据表的查找与统计需要排序

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<=n/2条记录

4.1.1 排序问题--结构化数据表查找问题

(2)为什么要研究排序问题？

结构化数据表的查找与统计需要排序

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

???

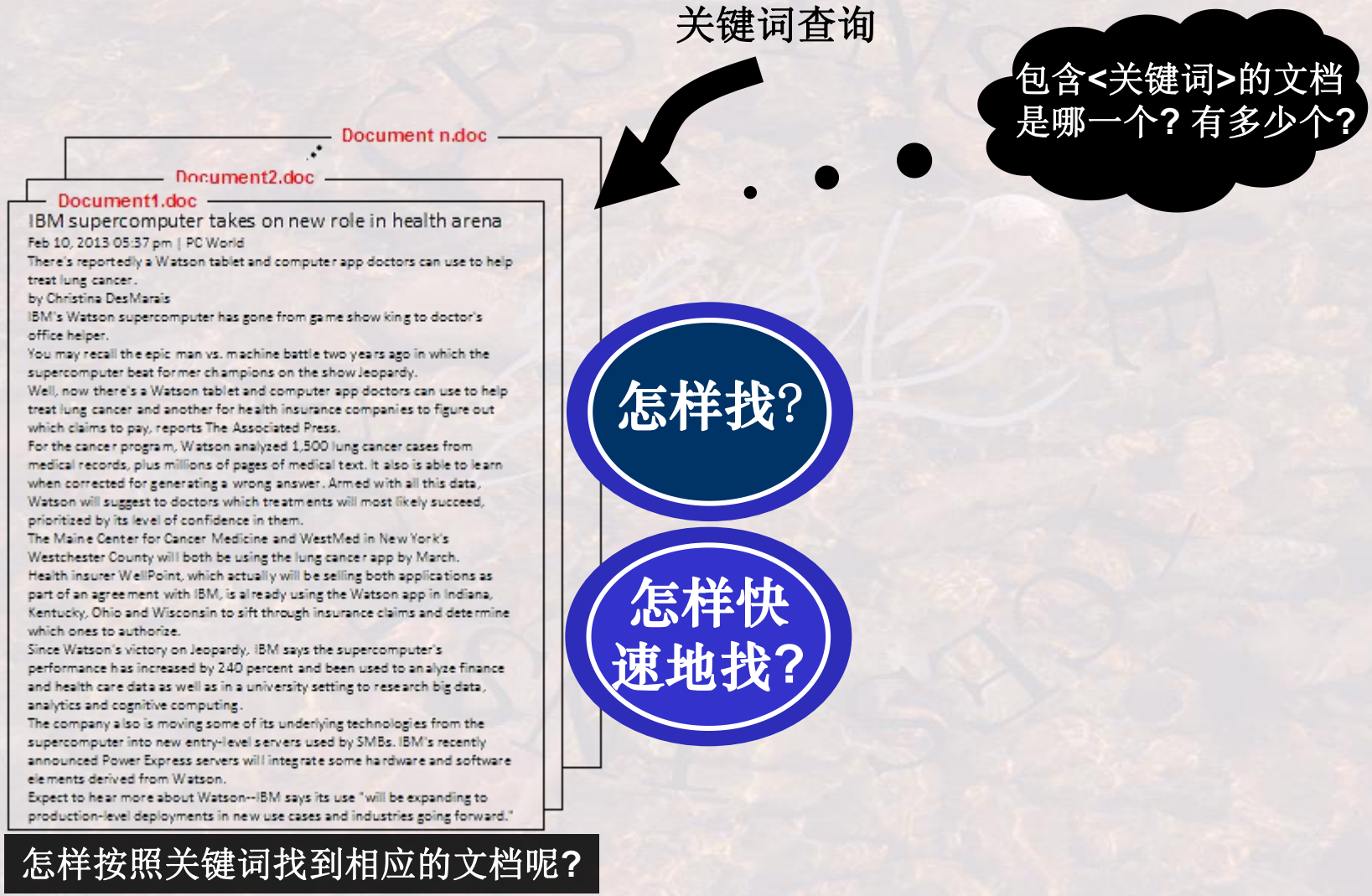
- 统计各个分数段的人数
- 统计每个同学的平均分数
- 统计每门课的平均分数

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？

4.1.1 排序问题--非结构化的数据文档查找问题

(1)非结构化数据(文档)的查找问题



4.1.1 排序问题--非结构化的数据文档查找问题
(2)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?

Document n.doc
Document2.doc
Document1.doc

IBM supercomputer takes on new role in health arena

Feb 10, 2013 05:37 pm | PC World

There's reportedly a Watson tablet and computer app doctors can use to help treat lung cancer.

by Christina DesMarais

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.

Well, now there's a Watson tablet and computer app doctors can use to help treat lung cancer and another for health insurance companies to figure out which claims to pay, reports The Associated Press.

For the cancer program, Watson analyzed 1,500 lung cancer cases from medical records, plus millions of pages of medical text. It also is able to learn when corrected for generating a wrong answer. Armed with all this data, Watson will suggest to doctors which treatments will most likely succeed, prioritized by its level of confidence in them.

The Maine Center for Cancer Medicine and WestMed in New York's Westchester County will both be using the lung cancer app by March.

Health insurer WellPoint, which actually will be selling both applications as part of an agreement with IBM, is already using the Watson app in Indiana, Kentucky, Ohio and Wisconsin to sift through insurance claims and determine which ones to authorize.

Since Watson's victory on Jeopardy, IBM says the supercomputer's performance has increased by 240 percent and been used to analyze finance and health care data as well as in a university setting to research big data, analytics and cognitive computing.

The company also is moving some of its underlying technologies from the supercomputer into new entry-level servers used by SMBs. IBM's recently announced Power Express servers will integrate some hardware and software elements derived from Watson.

Expect to hear more about Watson--IBM says its use "will be expanding to production-level deployments in new use cases and industries going forward."

怎样按照关键词找到相应的文档呢?

查找文档

对所有文档建立

关键词查询

关键词索引表---倒排索引

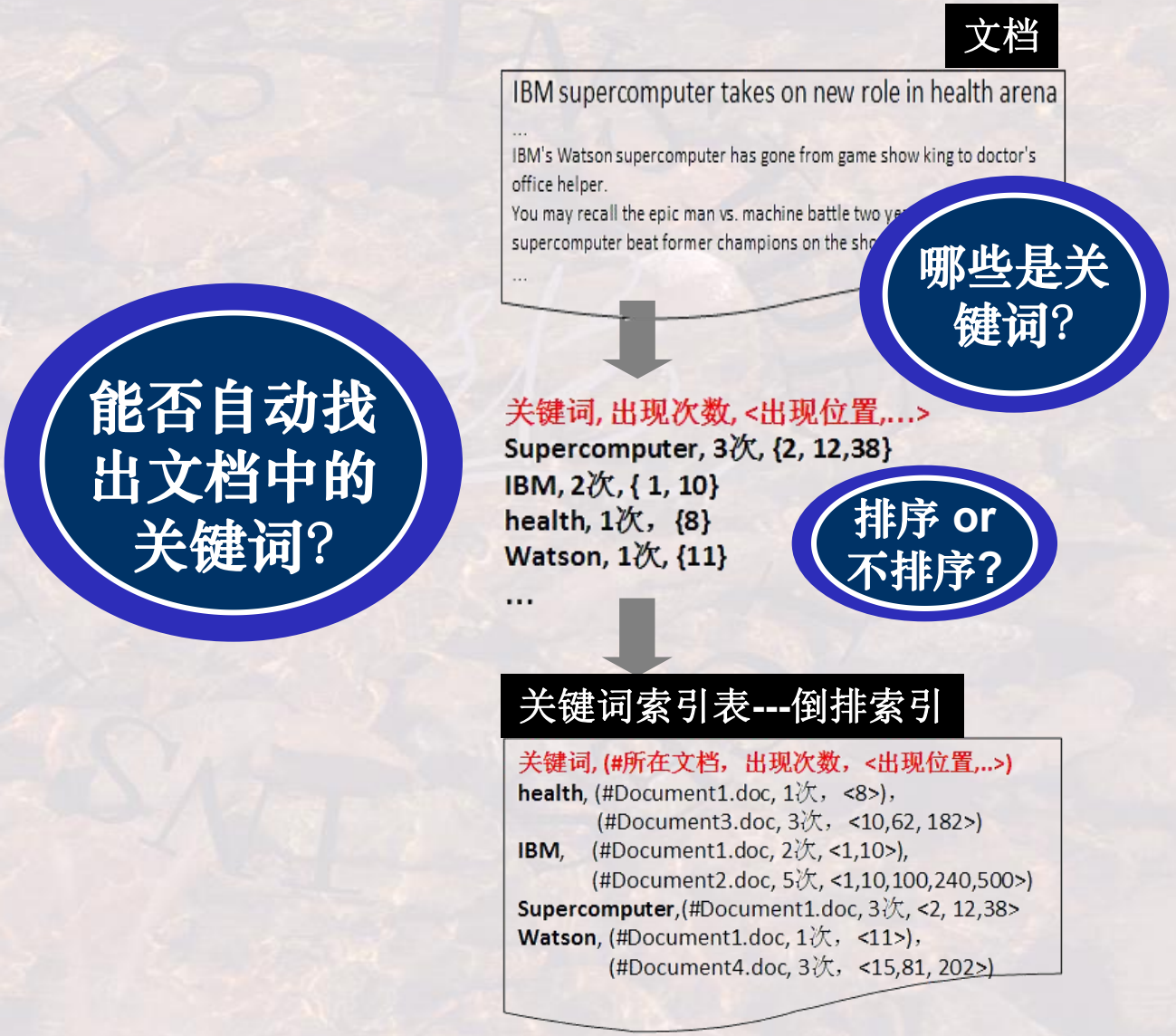
关键词, (#所在文档，出现次数，<出现位置,..>)
health, (#Document1.doc, 1次，<8>),
(#Document3.doc, 3次，<10,62, 182>)
IBM, (#Document1.doc, 2次, <1,10>),
(#Document2.doc, 5次, <1,10,100,24
Supercomputer,(#Document1.doc, 3次, <
Watson, (#Document1.doc, 1次，<11>)
(#Document4.doc, 3次，<15,8

排序 or 不排序?

怎样利用索引快速地找?

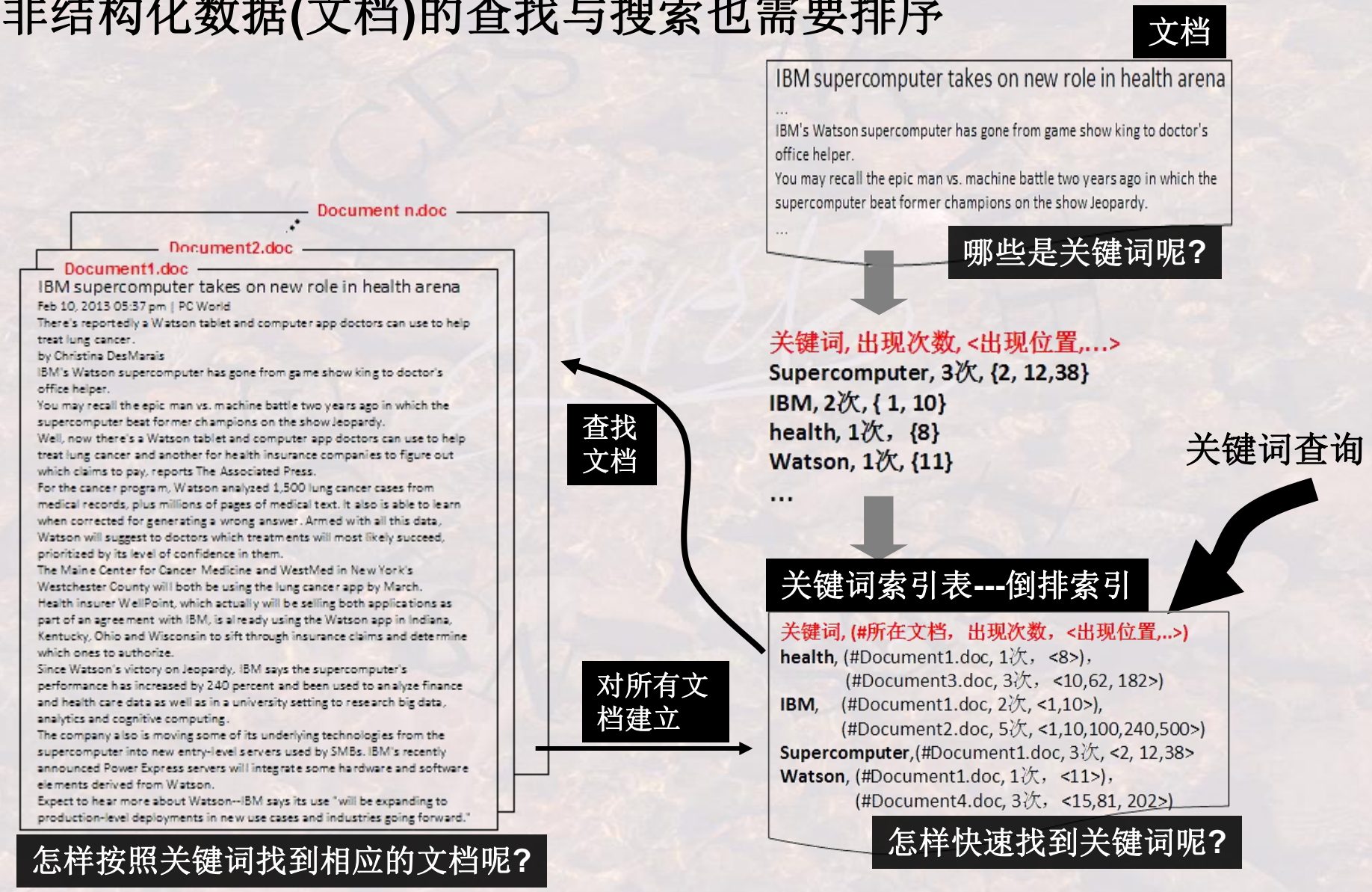
4.1.1 排序问题--非结构化的数据文档查找问题

(3)关键词的提取--需要排序



4.1.1 排序问题--非结构化的数据文档查找问题
(4)小结

非结构化数据(文档)的查找与搜索也需要排序



单选题 1分

设置

排序算法是最基本的算法，很多复杂算法都是以排序为基础进行构造的。关于排序算法，下列说法不正确的是_____

A 大规模数据集合中查找有无某些元素的问题，有序数据集合比无序数据集合的查找要快得多；

B 大规模数据集合中按元素分组进行计算的问题，有序数据集合比无序数据集合的计算要快得多；

C 对无序数据集合，两个算法 X和Y：X采用无序数据处理，Y采用先将无序数据排序成有序数据，然后进行处理；则对前述(A)、(B)两类问题，Y算法一定比X算法慢；

D 上述说法有不正确的；

提交

4.1.2 基本排序算法

4.1.2 基本排序算法--内排序算法：插入排序

(1)插入排序的思想

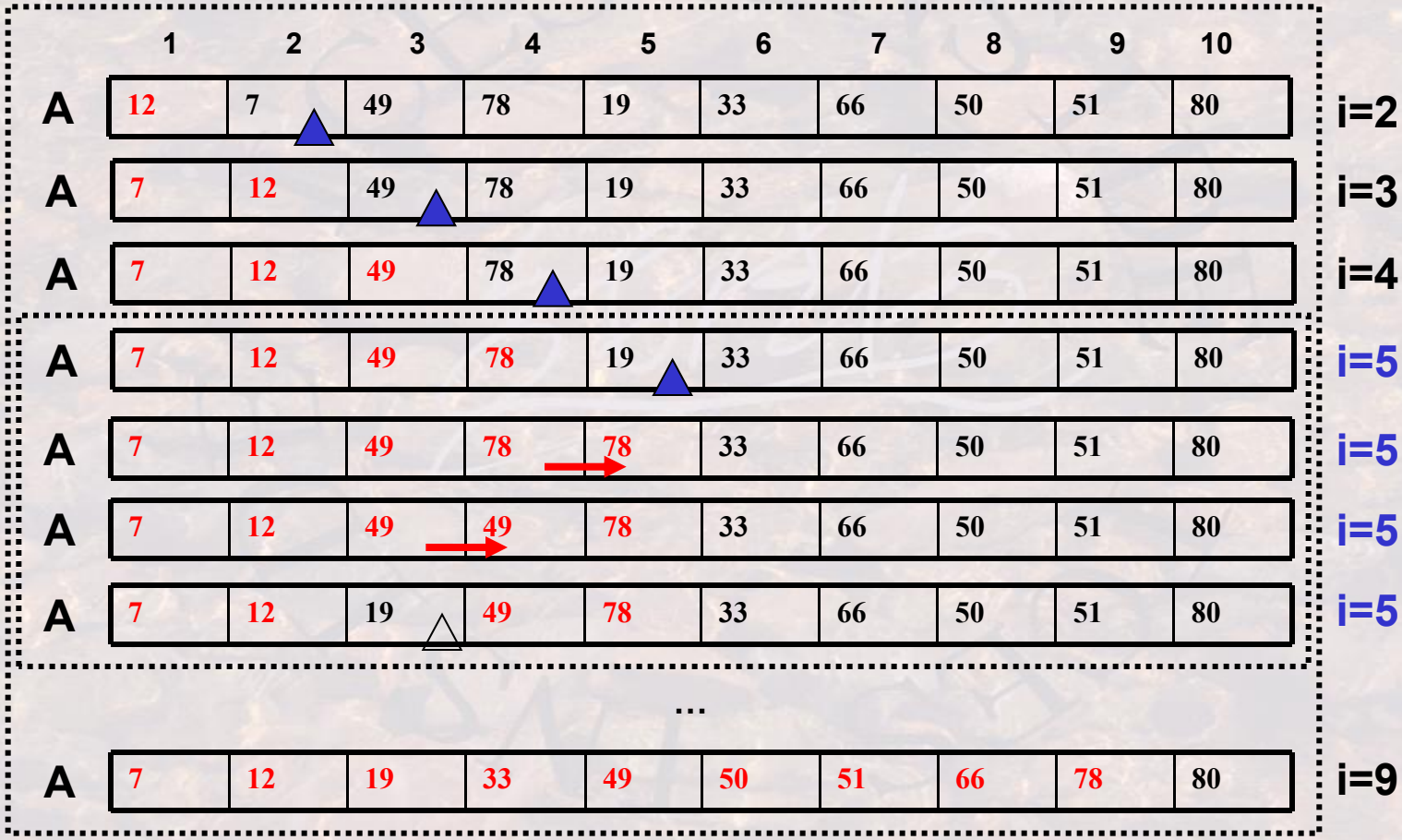
插入法排序

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

4.1.2 基本排序算法--内排序算法：插入排序

(2)插入排序的过程模拟

插入法排序



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

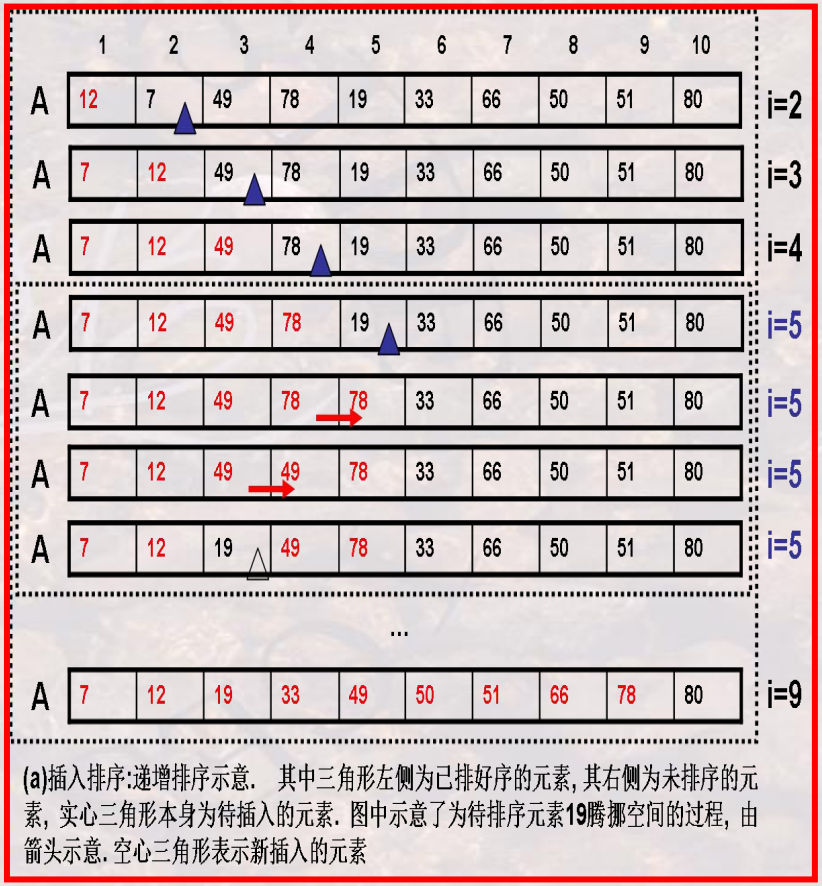
4.1.2 基本排序算法--内排序算法：插入排序

(3)插入排序的算法表达

插入法排序

```
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.  }
```

$O(N^2)$



(a)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

4.1.2 基本排序算法--内排序算法：简单选择法排序

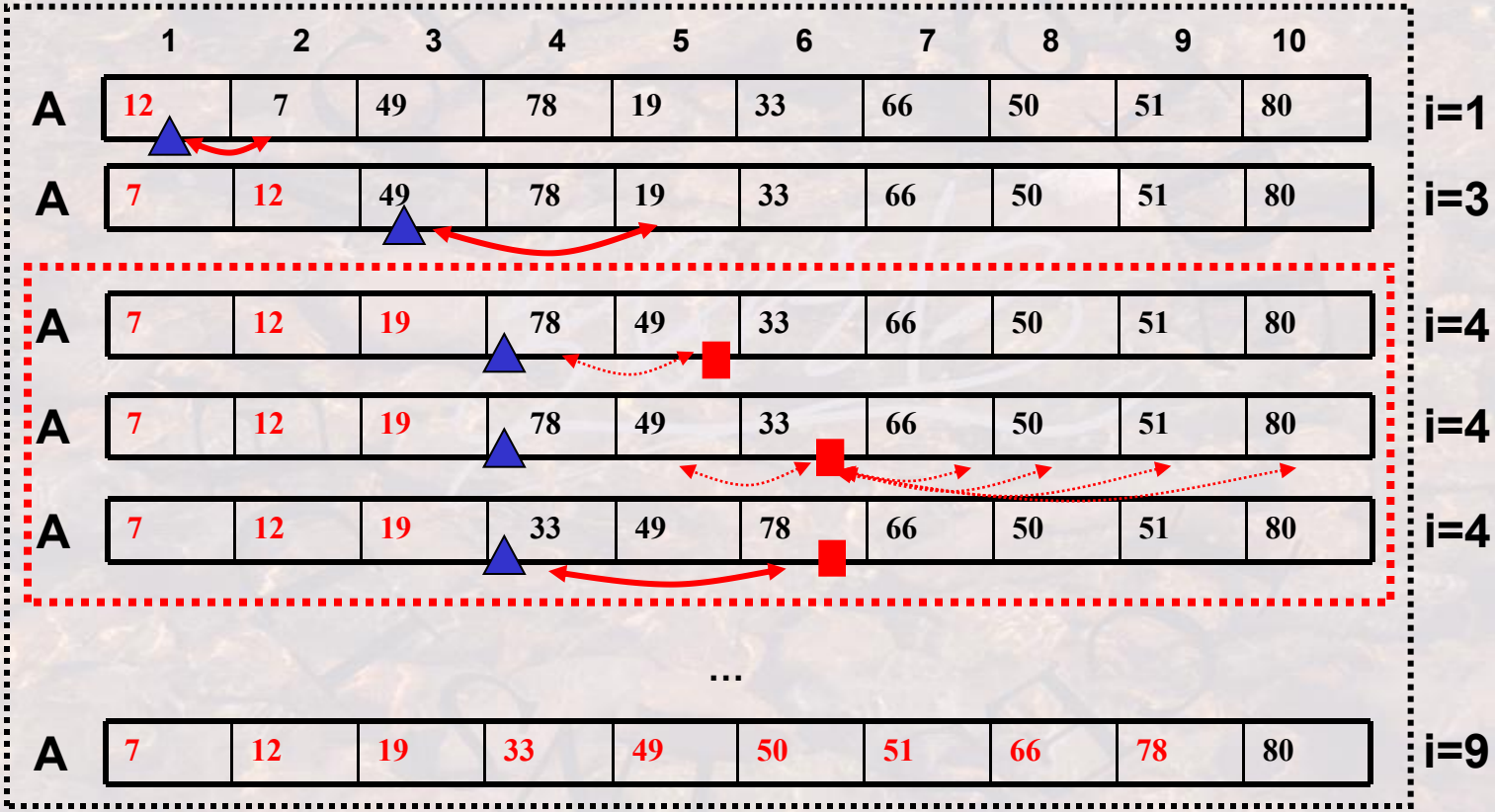
(1)简单选择排序的思想

简单选择法排序

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]的余下的数组元素中再找出最小值的元素，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排序。

4.1.2 基本排序算法--内排序算法：简单选择法排序
(2)简单选择排序的过程模拟

简单选择法排序



(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

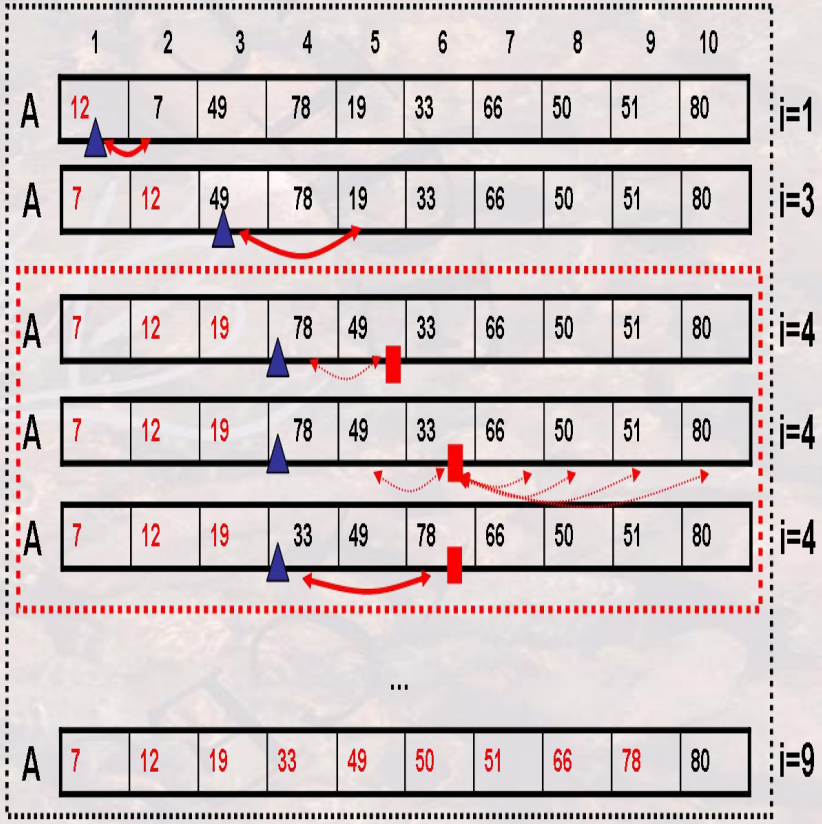
4.1.2 基本排序算法--内排序算法：简单选择法排序

(3)简单选择排序的算法表达

简单选择法排序

```
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<>i then
6.          {
7.             temp =A[k];
8.            A[k]=A[i];
9.            A[i]=temp;
10.         }
11. }
```

$O(N^2)$



(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

4.1.2 基本排序算法--内排序算法：冒泡法排序

(1)冒泡排序的基本思想

冒泡法排序

一个轮次一个轮次的处理。

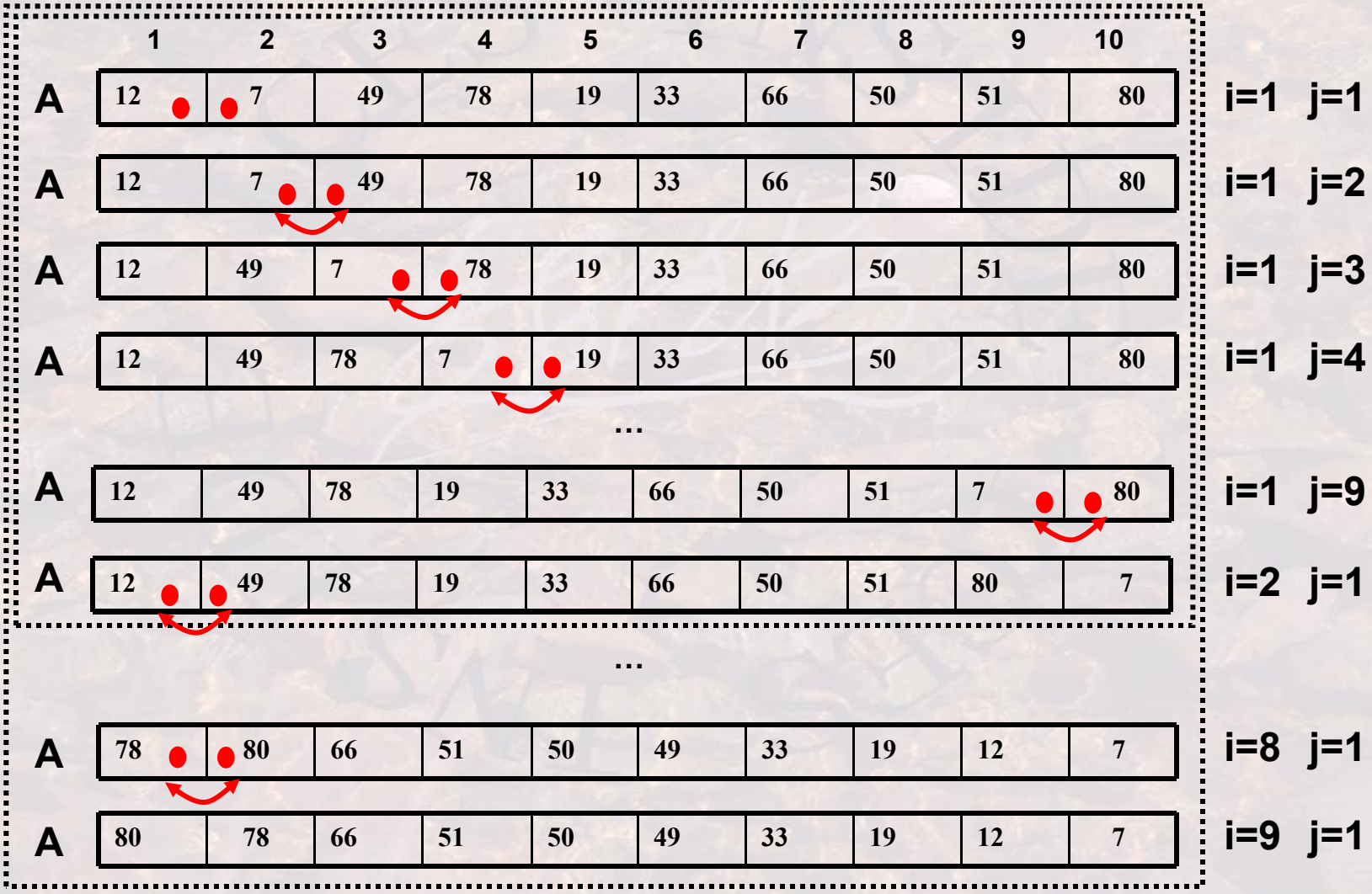
在每一轮次中依次对待排序数组元素中相邻的两个元素进行比较，将大的放前，小的放后--递减排序(或者是将小的放前，大的放后--递增排序)。

当没有交换时，则数据已被排好序。

4.1.2 基本排序算法--内排序算法：冒泡法排序

(2)冒泡排序的过程模拟

冒泡法排序



(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

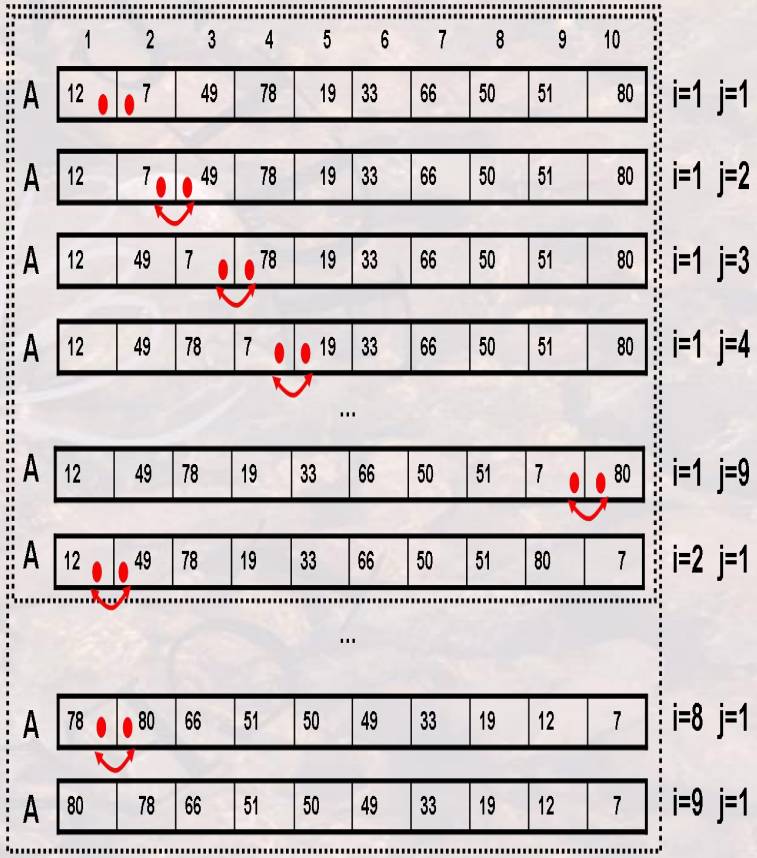
4.1.2 基本排序算法--内排序算法：冒泡法排序

(3)冒泡排序的算法表达

冒泡法排序

```
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(N²)



(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

其他排序算法

快速排序

从待排序列中任取一个元素 (例如取第一个) 作为中心，所有比它小的元素放在左侧，所有比它大的元素放在右侧，形成左右两个子序列；
然后再对各子序列重新选择中心元素并依此规则调整，直到每个子序列的元素只剩一个，此时便为有序序列了。

同学自己能否写出其算法--这里将用到递归—(略)

单选题 1分

设置

关于三种排序算法，下列说法正确的是_____。

A 三种算法的时间复杂度都为O(n^2)，所以三种算法的执行效率是一样的；

B 尽管三种算法的时间复杂度都为O(n^2)，但细致比较还是有差别的，例如冒泡法排序比选择法排序要快一些；

C 尽管细致比较三种算法的执行时间是有差别的，但这种差别对内排序问题而言是可以忽略不计的

D 尽管细致比较三种算法的执行时间是有差别的，这种差别对内排序问题而言是重要的，因为内排序算法可能要被频繁的执行

提交

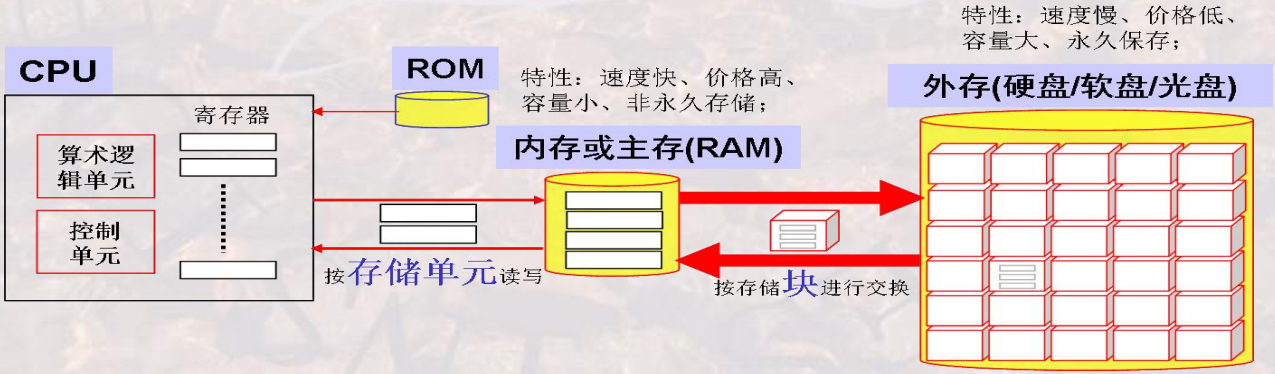
受限资源约束下的算法
--外排序算法

4.1.2 基本排序算法—外排序算法

(1)排序问题的复杂性在哪里？

●**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；

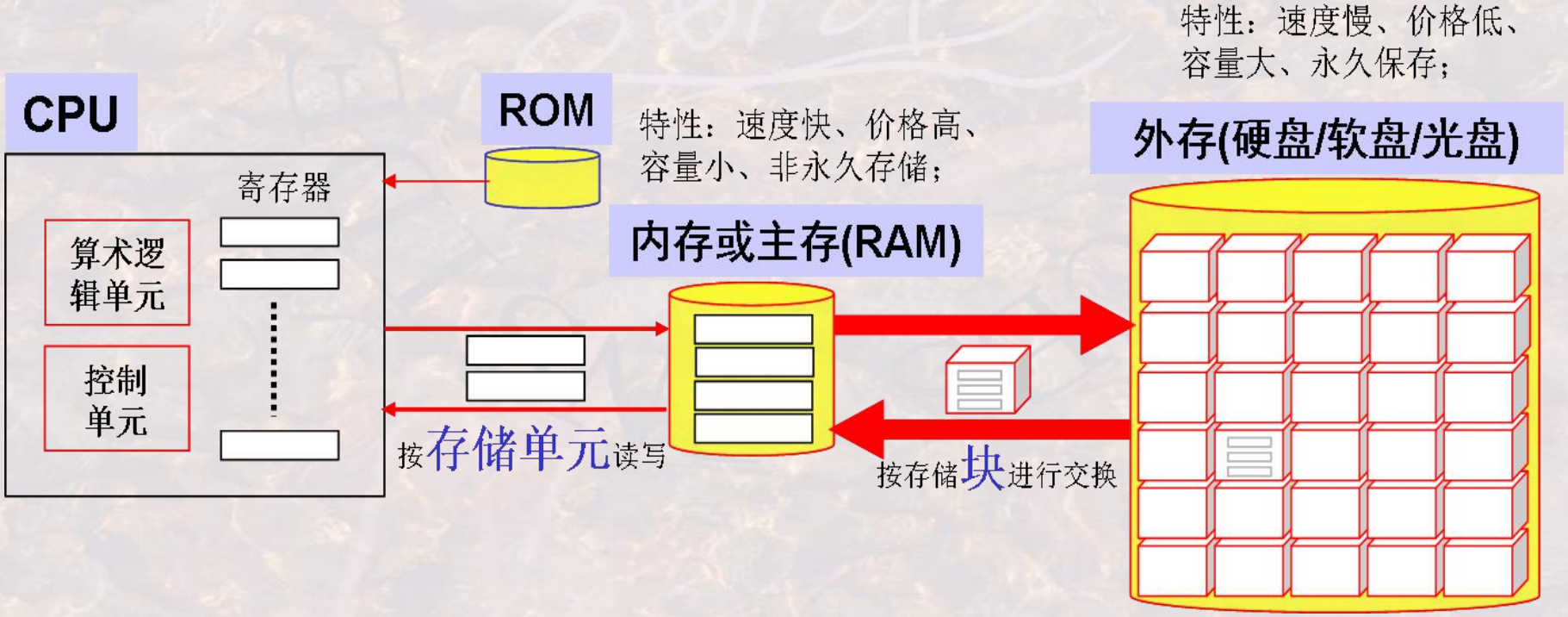


问题类比：某教师要对学生按身高排序。教师只能在房间(相当于内存)中对学生进行排序，假设房间仅能容纳100人，那么对于小于100人的学生排序便属于内排序问题。而对于大于100人，如1000人的学生排序，学生并不能都进入房间，而只能在操场(相当于磁盘)等候，轮流进入房间，这样的排序便属于外排序问题。

4.1.2 基本排序算法—外排序算法：
(2)外排序环境与问题示例

- 内存: 2GB
- 待排序数据: 7GB, 10GB, 100GB?--大数据集合

•这种需要使用硬盘等外部存储设备进行大数据集合排序的过程/算法称为外排序(External sorting) 。

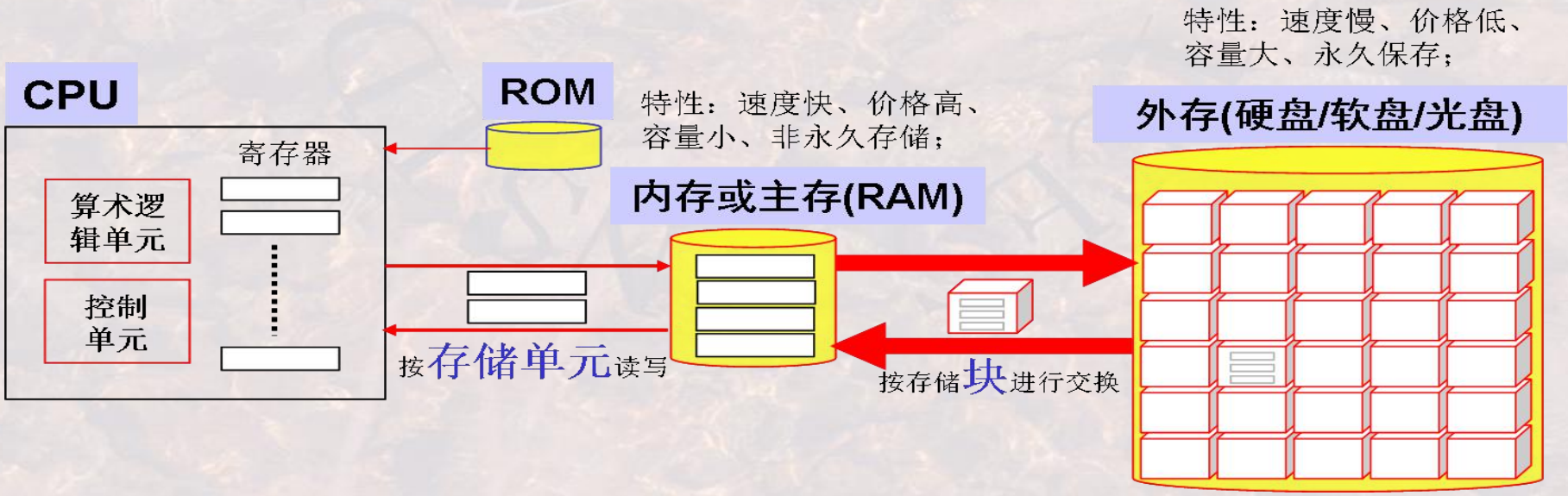


4.1.2 基本排序算法—外排序算法
(2)外排序环境与问题示例

外排序算法的环境/资源(仅介绍思想,忽略一些细节), 假设:

- 读写磁盘块函数: ReadBlock, WriteBlock
- 内存大小: 共B_{memory}=6块, 每块可装载R_{block}=5个元素
- 待排序数据: $R_{problem}$=60个元素, 共占用$B_{problem}$=12块

问题: $B_{problem}$块的数据怎样利用B_{memory}块的内存进行排序?



4.1.2 基本排序算法—外排序算法
(3)外排序的基本处理策略

基本排序策略

●$B_{problem}$块数据可划分为N个子集合，使每个子集合的块数小于内存可用块数，即：$B_{problem}/N < B_{memory}$。每个子集合都可装入内存并采用内排序算法排好序并重新写回磁盘。

问题转化为: N个已排序子集合的数据怎样利用内存进行总排序?

子集合1	90	86	82	80	75	70	60	58	43	32	28	15	11	08	05
子集合2	55	45	35	30	27	24	20	18	10	09	08	08	06	04	03
子集合3	80	78	72	62	52	50	42	38	34	31	29	19	16	13	10
子集合4	72	70	68	64	62	60	45	42	38	35	25	20	15	09	08

B_{memory} =6块,

R_{block} =5个元素

$R_{problem}$=60个元素

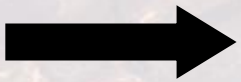
$B_{problem}$=12块

基本排序算法
--多路归并排序

4.1.2 基本排序算法—外排序算法：多路归并排序

(1)外排序的问题？

子集合1	90	86	82	80	75	70	60	58	43	32	28	15	11	08	05
子集合2	55	45	35	30	27	24	20	18	10	09	08	08	06	04	03
子集合3	80	78	72	62	52	50	42	38	34	31	29	19	16	13	10
子集合4	72	70	68	64	62	60	45	42	38	35	25	20	15	09	08

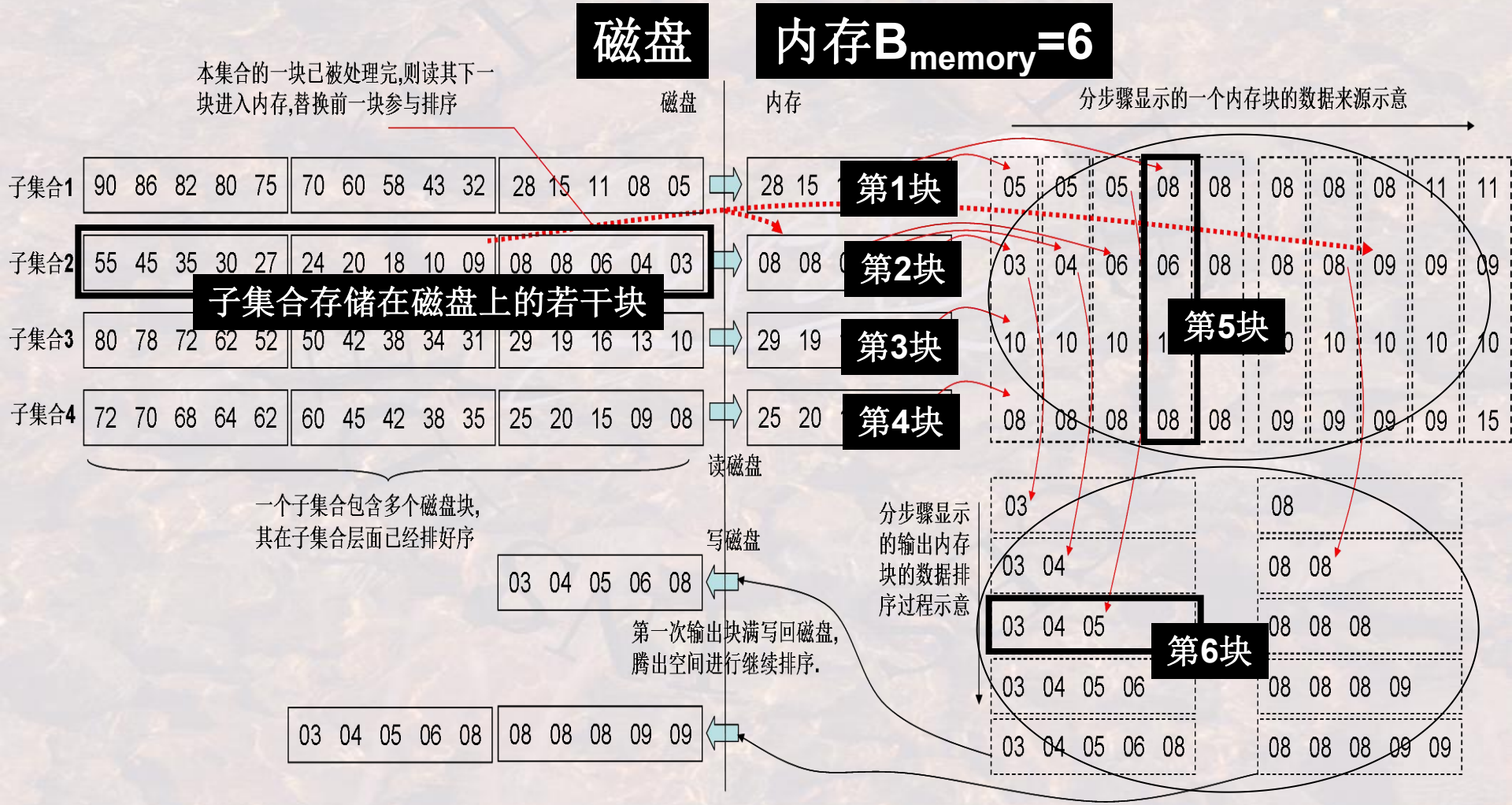


子集合已排好序，如何进行总排序

•内存不能装下所有子集合

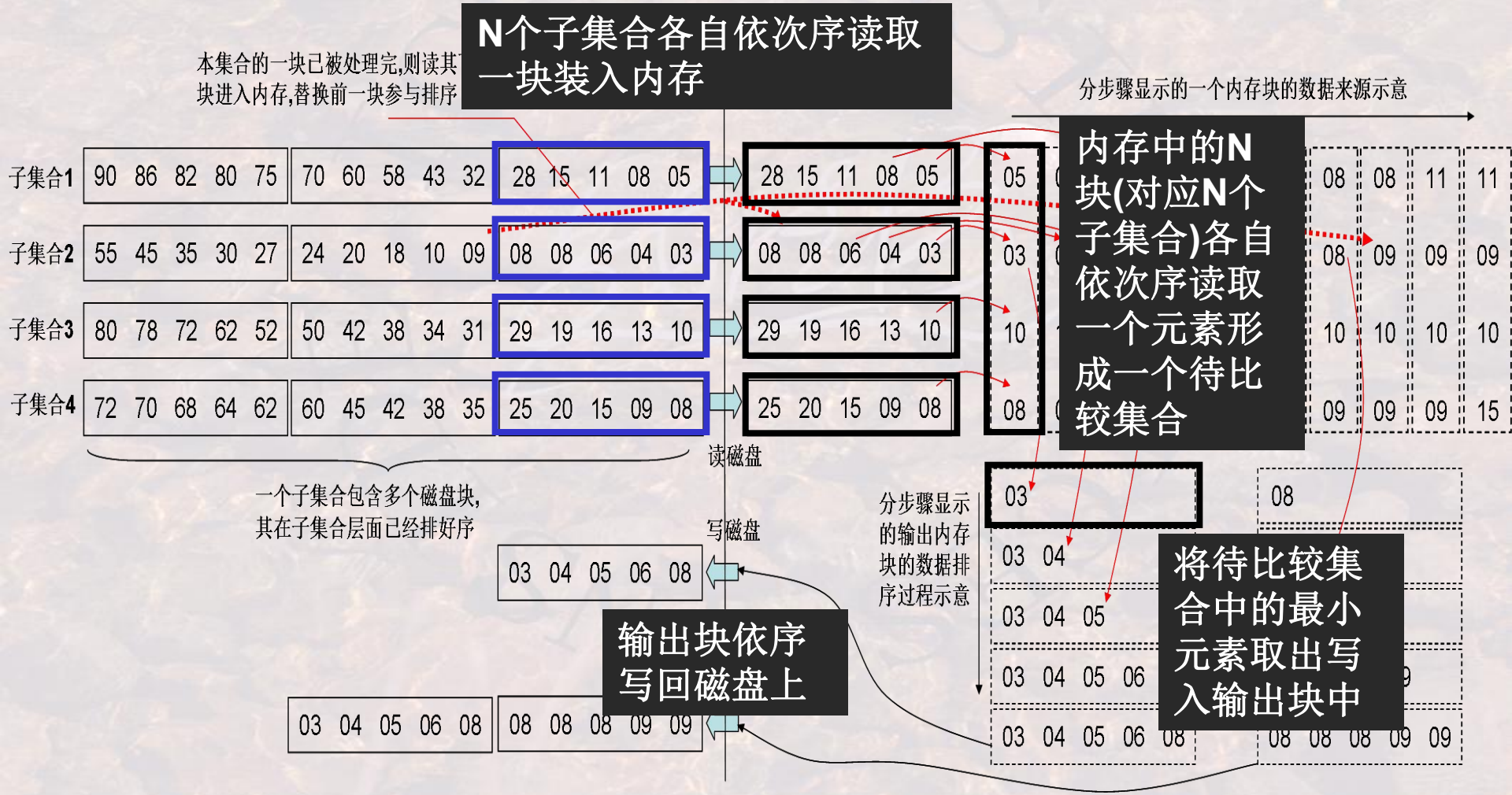
4.1.2 基本排序算法—外排序算法：多路归并排序

(2)内存资源的使用分配



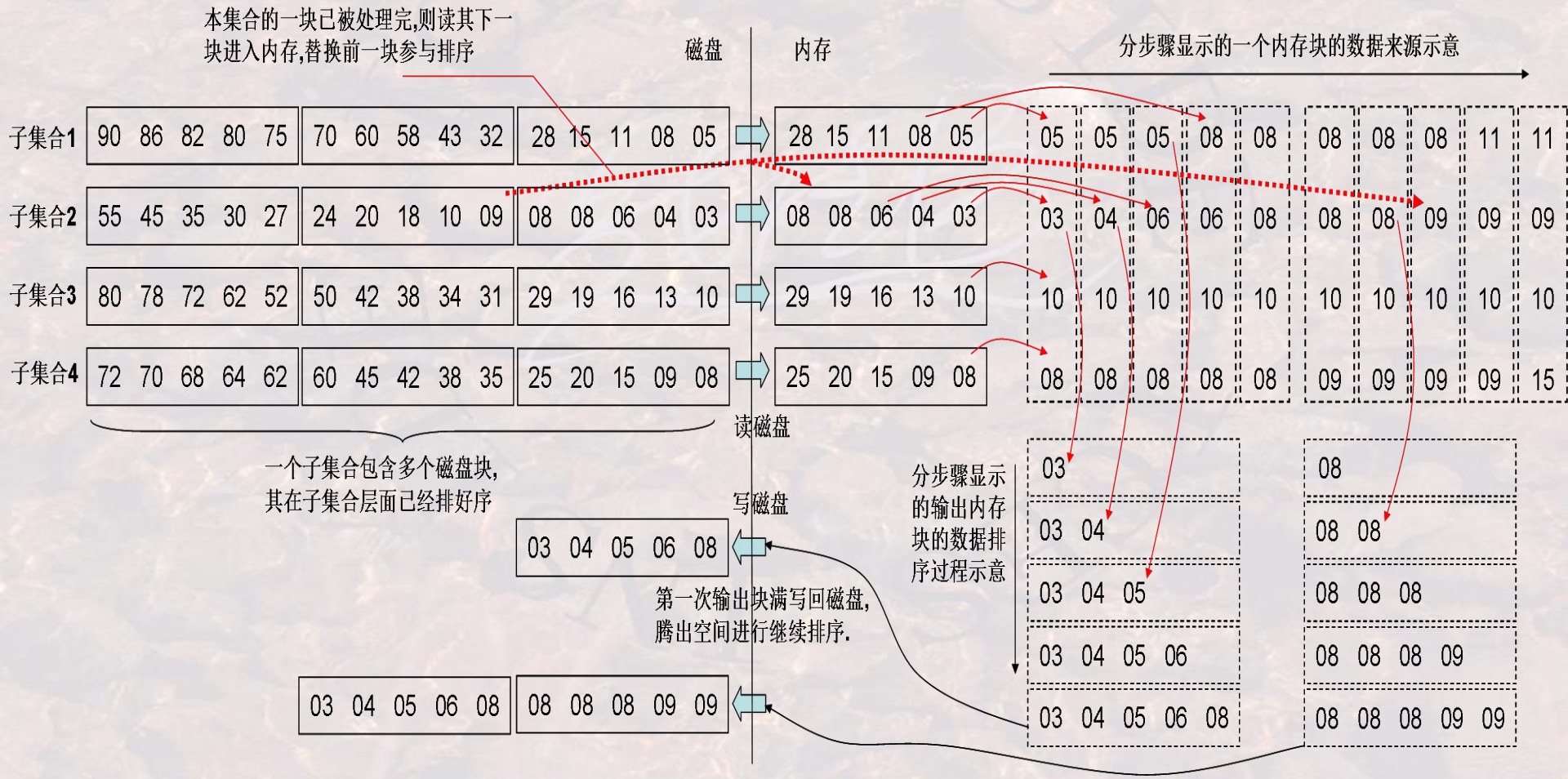
4.1.2 基本排序算法—外排序算法：多路归并排序

(3)基本归并动作



4.1.2 基本排序算法—外排序算法：多路归并排序
(4)过程模拟

归并排序--过程模拟(详细介绍参见另一部份：过程模拟)



4.1.2 基本排序算法—外排序算法：多路归并排序
(4)外排序暨多路归并排序的算法

归并排序--算法描述
(同学自己阅读)

已知：$S_{problem}$为待排序元素集合，$R_{problem}$–待排序集合中的元素个数，R_{block}-磁盘块或内存块能存储的元素个数，B_{memory}-可用内存块的个数，R(S)为求集合S的元素个数的函数，M_i为内存的第i块，P_{output}为输出块内存中当前元素的指针。

1. 将待排序集合 $S_{problem}$划分为 m 个子集合 S_1, S_2,...,S_m, 其中 $S_{problem}=\cup_{i=1,...,m}S_i$，且 $R_{problem}=\Sigma_{i=1,...,m}R(S_i)$, $R(S_i)<=B_{memory}*R_{block}$, i=1,...,m(注:每个 S_i的元素个数小于内存所能装载的元素个数).
2. for i=1 to m
3. {将 S_i装入内存，并采用一种内排序算法进行排序，排序后再存回相应的外存中}
/*步骤 2 和 3 完成子集合的排序。接下来要进行归并，M_1,...,M_m用于分别装载 S_1,...,S_m的一块*/;
4. for i=1 to m
5. { 调用 read block 函数，读 S_i的第一块存入 M_i中，同时将其第一个元素存入 $M_{compare}$的第 i_{th}个位置; }
6. 设置 P_{output}为输出内存块的起始位置;
7. 求 $M_{compare}$中 m 个元素的最小值及其位置 i。
8. If (找到最小值及其位置 i) then
9. { 将第 i_{th}个位置的元素存入 M_{output}中的 P_{output}位置，P_{output}指针按次序指向下一位置；
10. If (P_{output}指向结束位置) then
11. { 调用 Write Block 按次序将 M_{output}写回磁盘；置 P_{output}为输出内存块的起始位置；继续进行; }
12. 获取 M_i的下一个元素.
13. If (M_i有下一个元素)
14. { 将 M_i下一个元素存入 $M_{compare}$的第 i_{th}个位置。转步骤 7 继续执行。 }
15. Else { 调用 read block 按次序读 S_i的下一块并存入 M_i;
16. If (S_i有下一块)
{ 将其第一个元素存入 $M_{compare}$的第 i_{th}个位置。转步骤 7 继续执行。 }
17. ELSE { 返回一个特殊值如 Finished，以示 S_i子集合处理完毕，M_i为空，且使 $M_{compare}$中的第 i_{th}位置为该特殊值，表明该元素不参与 $M_{compare}$的比较操作。转步骤 7 继续执行。 }
18. }
19. } /*若 $M_{compare}$的所有元素都是特殊值 Finished，即没有最小值，则算法结束*/

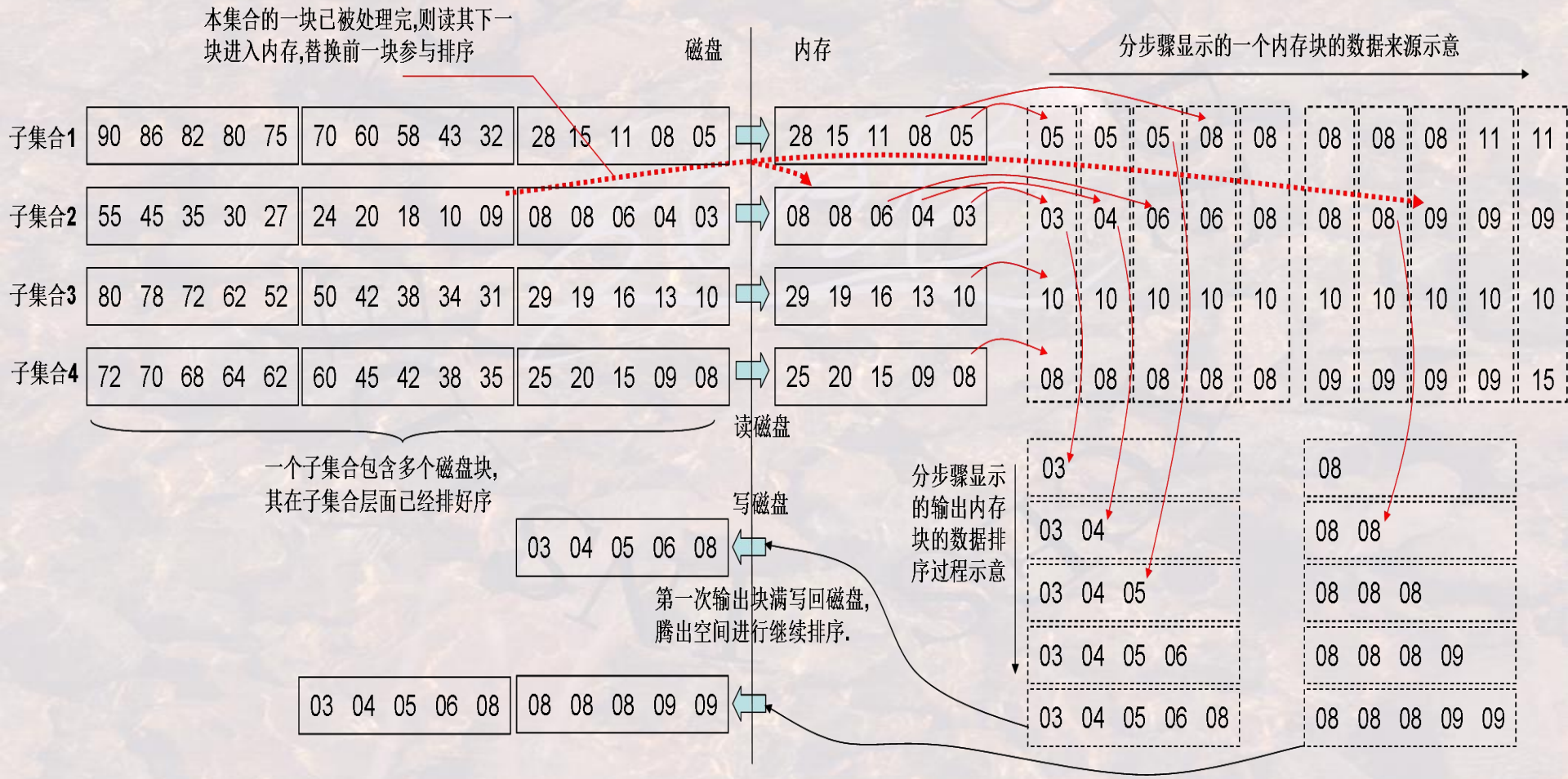
基本排序算法
--多路归并排序的过程模拟

基本排序算法IV--外排序之多路归并排序的过程模拟

(1)过程模拟-1

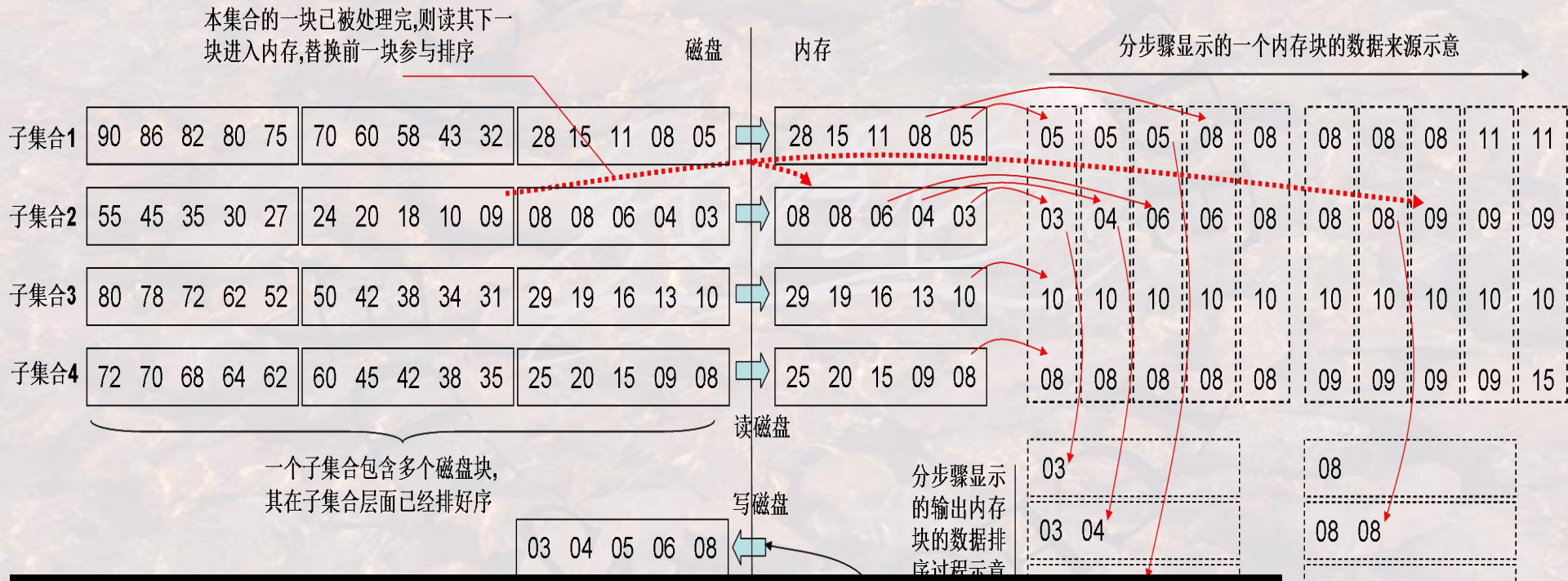
基本排序算法IV--外排序之多路归并排序的过程模拟
(2)过程模拟小结

归并排序--过程模拟



基本排序算法-续
--多路归并排序-讨论

归并排序--讨论



算法的效率：读写磁盘块的次数，即I/O数= 4 $B_{problem}$

子集合排序阶段读一遍写一遍 2 $B_{problem}$

归并阶段读一遍写一遍 2 $B_{problem}$

基本排序算法续--外排序之多路归并排序-讨论
(2)算法在任何情况下都可以应用吗?

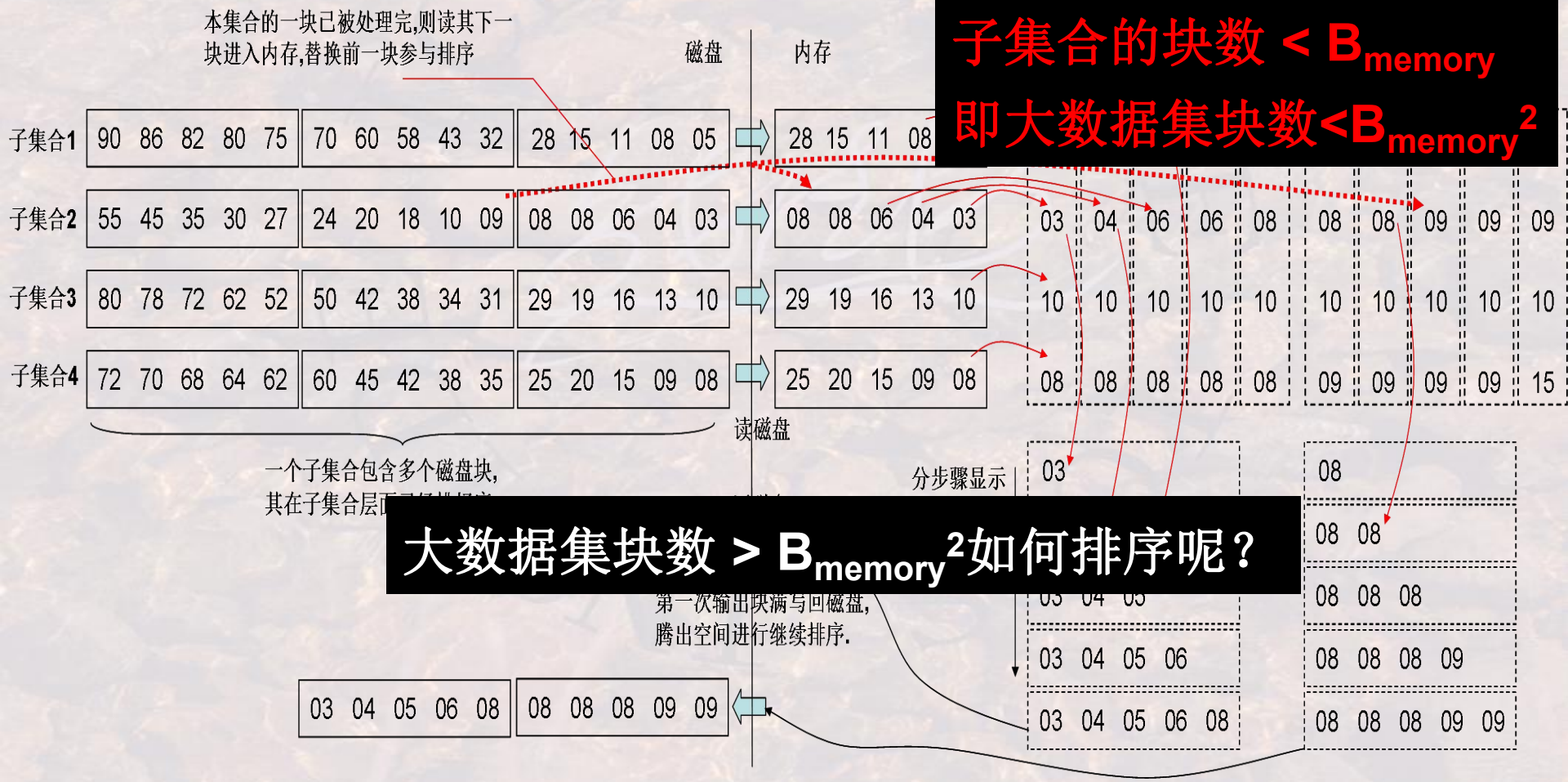
归并排序--讨论

算法的应用条件:

子集合数< B_{memory}

子集合的块数 < B_{memory}

即大数据集块数<B_{memory}^2



大数据集块数 > B_{memory}^2如何排序呢?

基本排序算法续--外排序之多路归并排序-讨论
(3)当更大规模的数据需要排序时怎么办?

归并排序--讨论

- 内存大小: 共B_{memory} =3块
- 待排序数据: 共占用$B_{problem}$=30块

基本策略:

- 30块的数据集 ➔ 10个子集合，每个子集合3块，排序并存储。
- 10个已排序子集合分成5个组：每个组2个子集合, 分别进行二路归并，则可得到5个排好序的集合；
- 5个集合再分成3个组：每个组2个子集，剩余一个单独1组，分别进行二路归并，可得3个排好序的集合；再分组，再归并得到2个排好序的集合；再归并便可完成最终的排序。

基本排序算法续--外排序之多路归并排序-讨论
(4)思考一下下列情况排序，应该怎么办？

归并排序--思考

假如内存共有8块，问其如何排序有70块的数据集呢？你是采用二路归并、三路归并、…、七路归并？你设计的具体算法，磁盘读写次数是多少呢？磁盘读写次数最少的应是几路归并？

4.1.3 PageRank排序
--排序问题的不同思考方法

PageRank网页排序算法I--网页排序问题及思想

PageRank网页排序算法I--网页排序问题及思想

(1)网页排序问题?

问题背景---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网页排序算法I--网页排序问题及思想
(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

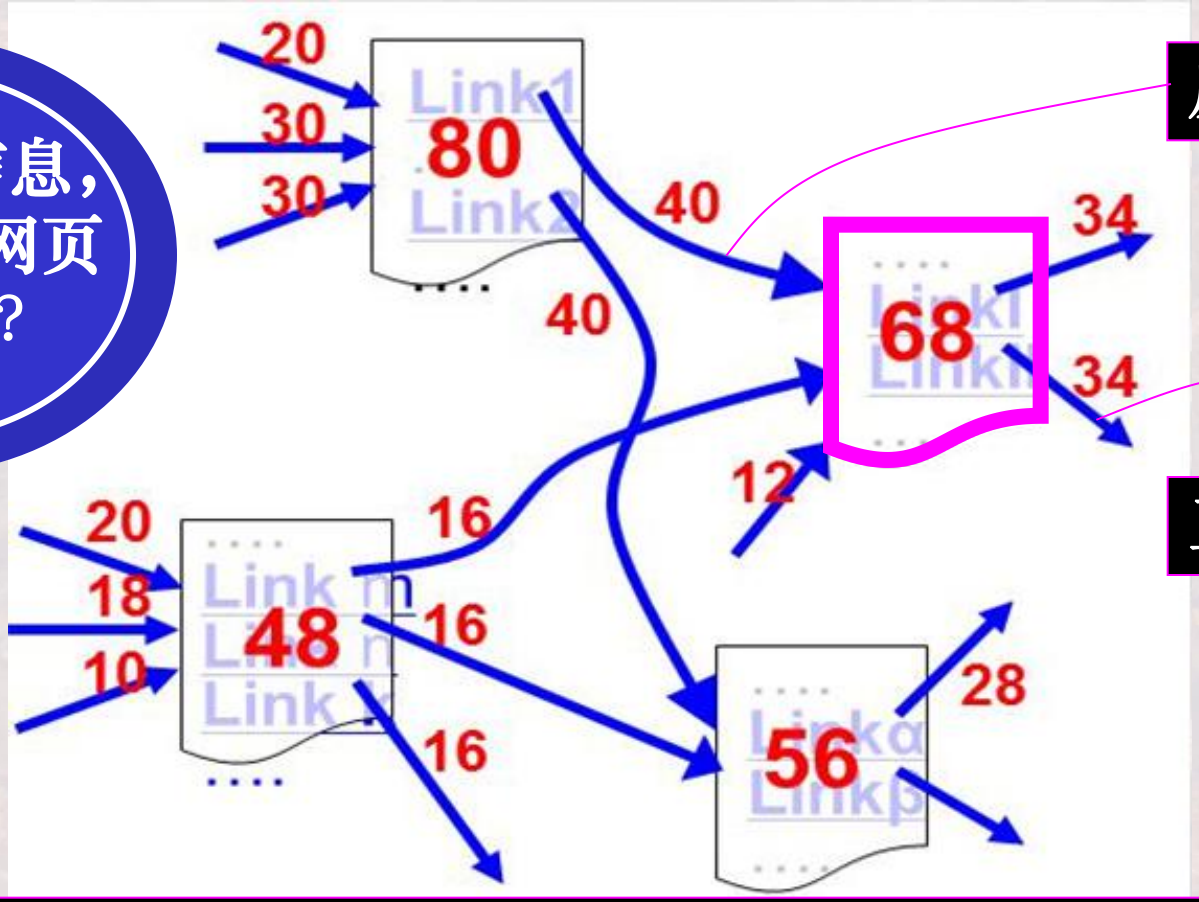
<标记>文本</标记>

<A HREF=“http://www.bignet.net/realcorp/products.html”>Our Product Information</A>

<A HREF=“链接到另一个网页的地址”>Our Product Information</A>

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?



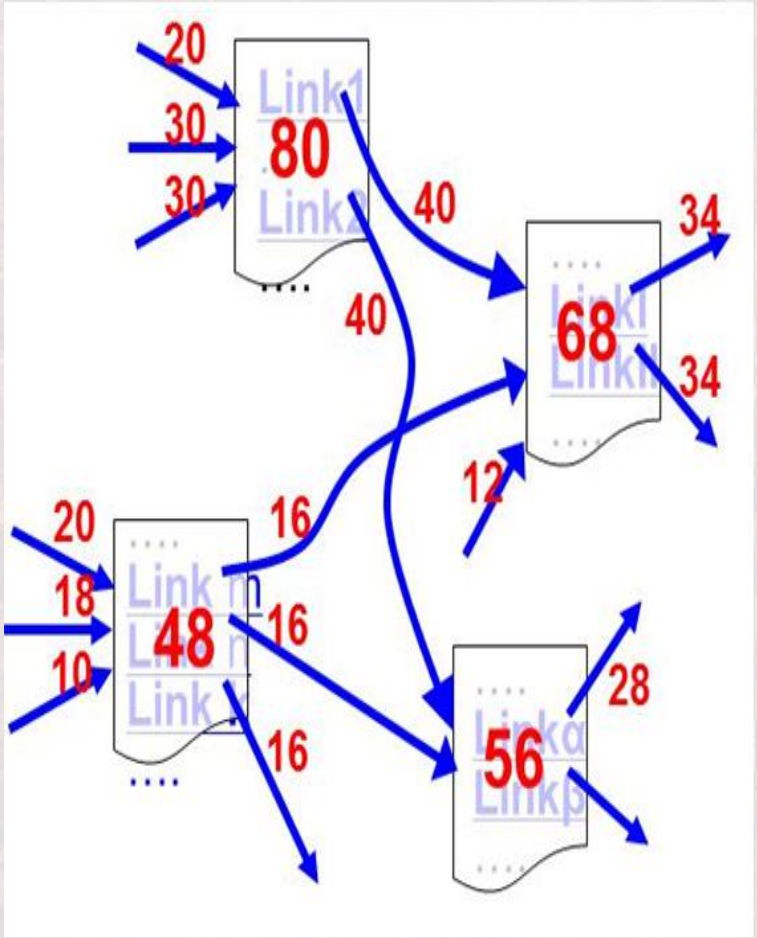
反向链接

正向链接

一个网页的正向链接是另一个网页的反向链接

PageRank网页排序算法I--网页排序问题及思想
(4)PageRank的基本思想

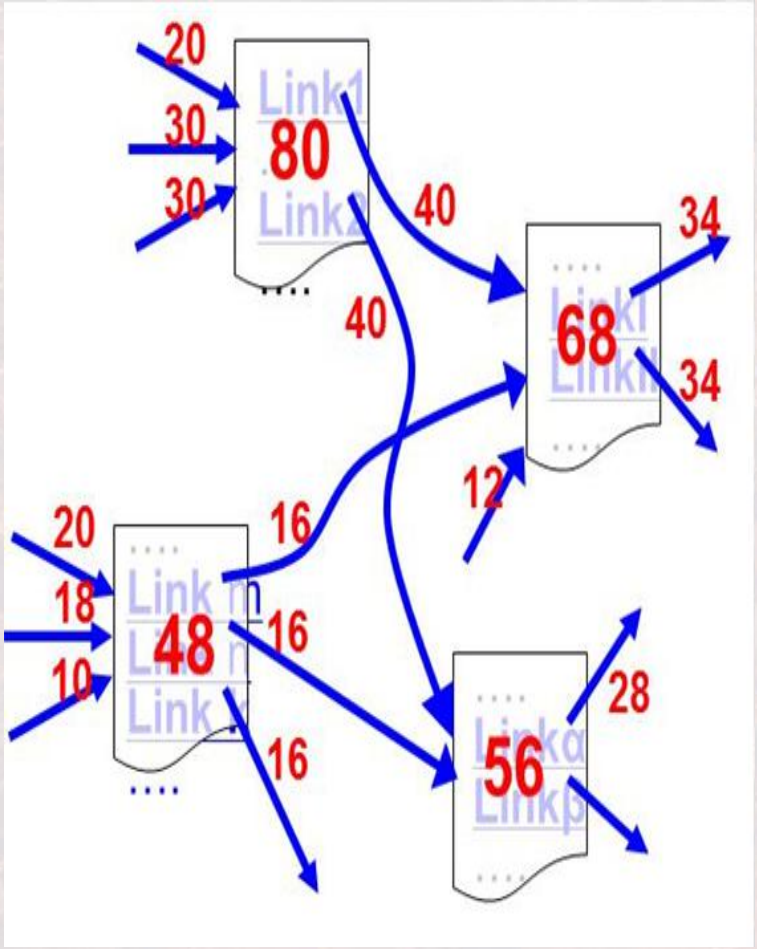
关于网页的基本观点



- 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网页排序算法I--网页排序问题及思想
(4)PageRank的基本思想

网页重要度



●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

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

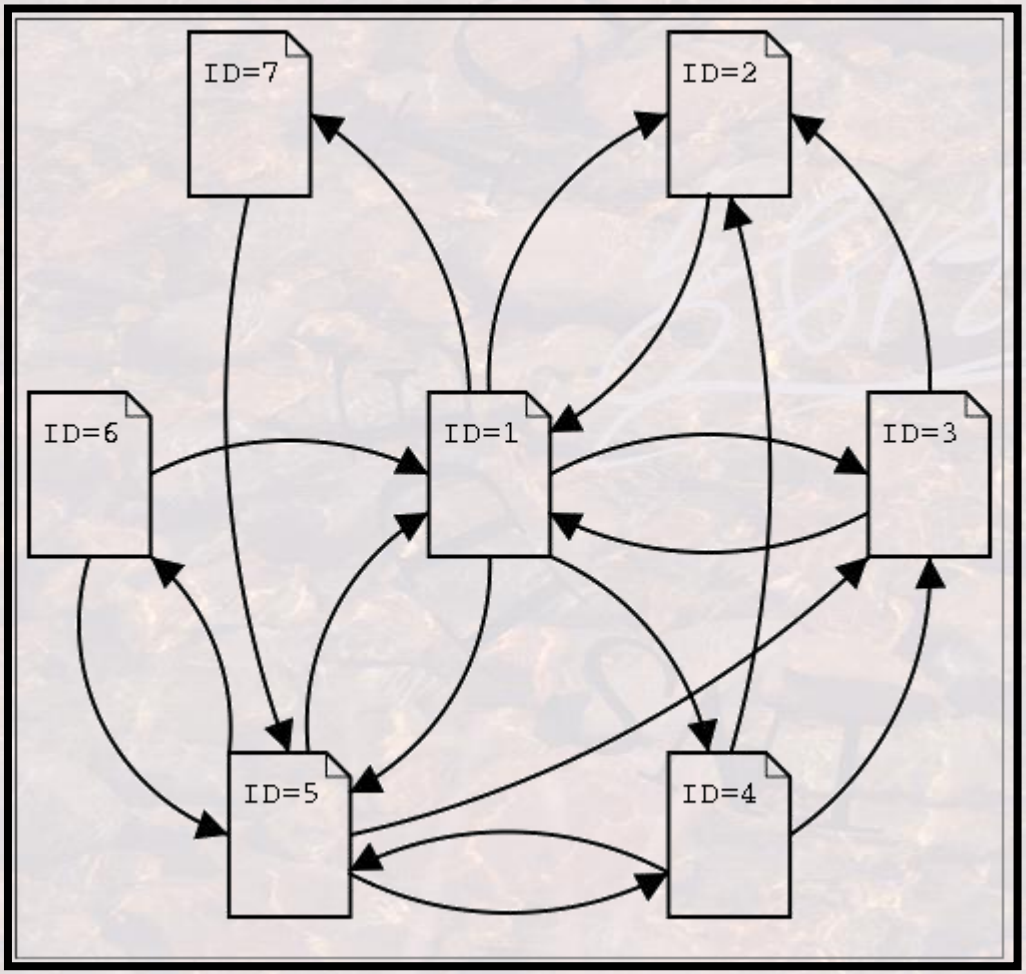
●一个正向链接的权值等于网页的重要度除以其正向链接数, 即：网页重要度R, 其正向链接数m, 则其每一个正向链接的权值

$$z = R/m。$$

怎样计算网页的重要度呢?

PageRank排序算法II
--网页排序问题的表达与建模

数学建模--示例



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网页排序算法II--网页排序问题的表达与建模
(1)问题的数学建模

数学建模--邻接矩阵

行i，列j 均是网页编号

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

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

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

PageRank网页排序算法II--网页排序问题的表达与建模
(2)正向链接的权值矩阵---转移概率

数学建模—转移概率

反向链接 →

邻接矩阵

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

反向链接的权值 →

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

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

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：

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

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

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

PageRank排序算法III
--网页重要度的迭代计算方法及讨论

PageRank网页排序算法III--网页重要度的迭代计算方法及讨论

(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：

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

第n-1次的网页重要度

第n次的网页重要度

转移概率矩阵M

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

矩阵乘法

迭代计算

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

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

… …

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

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

R的初始值是多少呢？从哪一个R_i开始计算呢？

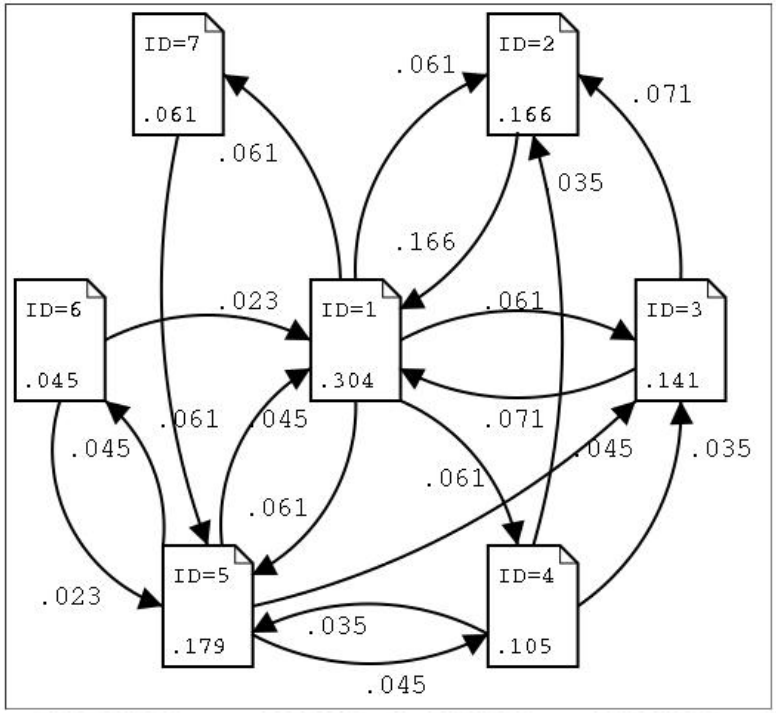
PageRank网页排序算法III--网页重要度的迭代计算方法及讨论
(2)PageRank的计算结果分析

PageRank—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排序算法IV
--PageRank与数学及算法总结

PageRank网页排序算法IV--PageRank与数学及算法总结
(1)PageRank计算 vs. 数学的特征方程

网页重要度的迭代计算

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

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

迭代计算网页重要度

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

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

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

…

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

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网页排序算法IV--PageRank与数学及算法总结
(2)PageRank算法总结

数学的语义：特征方程

求解思想：求稳定性

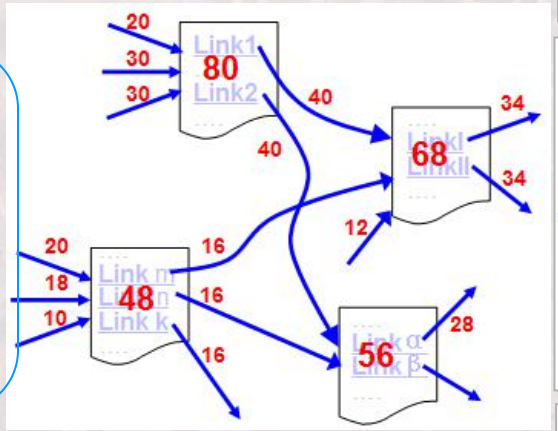
表达成数学：0,1矩阵
权值矩阵—转移概率矩阵

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：

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

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

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

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

… …

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

$R=R^{(n)}=R^{(n-1)}$----收敛状态暨稳定状态

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

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