算法与数据结构

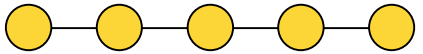
二、线性表

setoy@qq.com 2018.8

1. 线性结构

DATA STRUCTURE & ALGORITHM

- 由n个数据元素组成的有限序列
  - 除头元素外，每个元素都有一个前趋
  - 除尾元素外，每个元素都有一个后继
    - 26个英文字母表是一个线性表
    - 一张学生信息表也是一个线性表



姓名	性别	出生日期	家庭住址	联系电话	Email	入学成绩
陈词	女	1993/10/5	中山路5号	8526789	chenc@163.com	496
张然	男	1992/5/20	南京路10号	8324358	zhangr@sina.com	550
…						
林峰	男	1993/2/9	和平大街	8523456	lingf@sohu.com	520

2.1 线性结构的存储方式——顺序存储

DATA STRUCTURE & ALGORITHM

- 顺序存储结构（顺序表）：数组
  - 连续存储
  - 定位时间复杂度：$O(1)$
  - 插入、删除时间复杂度：$O(n)$
- 定位（查找）：第6个数是多少？89这个数在不在？在哪里？
- 插入：32
- 删除：52

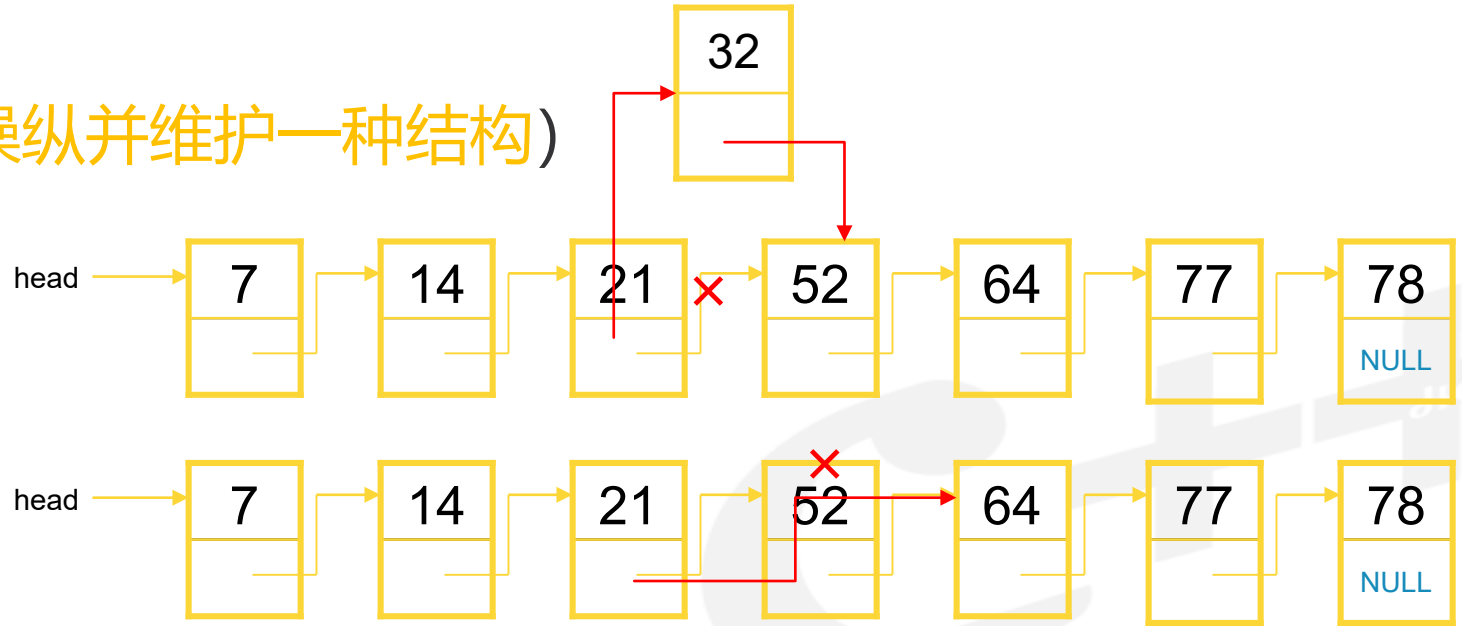
- 数组插入与删除均需移动后面的元素

1	2	3	4	5	6	7	8	9	10
7	14	16	21	52	64	77	78	89	97

2.2 线性结构的存储方式——链式存储

DATA STRUCTURE & ALGORITHM

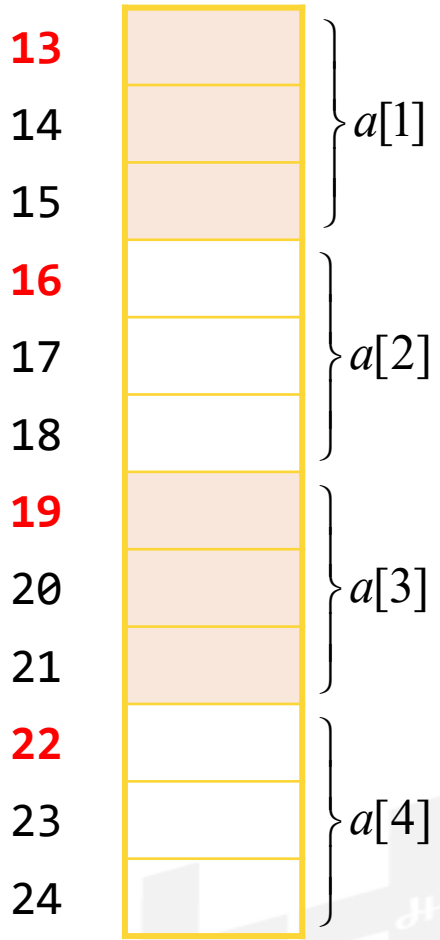
- 链式存储结构（链表）：链表
  - 非连续存储
  - 定位时间复杂度：$O(n)$
  - 插入、删除时间复杂度：$O(1)$
- 定位（查找）：第6个数；64在不在?
- 插入：32
- 删除：52
- 无需移动后面的元素
- （数据结构：使用数据、操纵并维护一种结构）



3. 顺序表的元素访问

DATA STRUCTURE & ALGORITHM

- 内存以字节为基本存储单位。
- 数组：从每个地址开始分配一段连续的内存空间。
- 一维数组按照下标递增的顺序访问表中元素
  - a[0] → a[1] → a[2] → ……→ a[n]
- 多维数组一般按照行优先方式存储：
  - a[0][0] → a[1][1] → a[1][2] → a[1][3] →
    a[1][0] → a[1][1] → a[1][2] → a[1][3] →
    … →
    a[3][0] → … → a[3][3]
- 如： type a[4], a[4][4];
- 假如每个元素需要3个字节（长度是3）。共需要3*4=12个字节和3*16=48个字节的空间。
- 一维数组中4个元素的(起始)地址分别为：13 16 19 22



4. 例1：有序表的合并

DATA STRUCTURE & ALGORITHM

- 已知有序表a和b中的数据元素都是按非降的顺序排列的，现要求将a、b合并为一个新的线性表c，c中的数据元素仍按非降的顺序排列。例如：
- a：2 5 9 13
- b：1 6 9 12 13 20
- 输出c：1 2 5 6 9 9 12 13 13 20

4. 例1：有序表的合并

DATA STRUCTURE & ALGORITHM

- 思路一：
  - 分别设定a、b两个线性表两个下标指示器i，j；比较a[i]和b[j]，将较小的元素放到c[k]中。
  - 检测a、b哪个线性表还未处理完，将未处理完的表中数据复制到c中。
  - an、bn分别表示两个线性表的长度

$O(an+bn)$

```
int i=1, j=1, k=0;
while ((i<=an) && (j<=bn))
{
    if (a[i]<b[j])
    {
        _____
    } else {
        _____
    }
}
while (i<=an)
{
        c[++k] = a[i++];
}
while (j<=bn)
{
        c[++k] = b[j++];
}
```

4. 例1：有序表的合并

DATA STRUCTURE & ALGORITHM

- 思路二：
  - 在思路一基础上，放置哨兵元素，减少比较次数和代码长度。

$O(an+bn)$

```
int i = 1, j = 1;
int cn = _____
a[an+1] = b[bn+1] = 0x7fffffff;
for (int k=1; k<=cn; k++)
{
    if (a[i]<b[j])
    {
        c[k] = a[i++];
    } else {
        c[k] = b[j++];
    }
}
```

4. 例1：有序表的合并

DATA STRUCTURE & ALGORITHM

- 思路三：
  - 用单向链表实现。

$O(an+bn)$

```
node * hc, * pc;  node * pa = ha;  node * pb = hb;
if (                    )
{
    hc = ha;  pa = pa->next;
} else {
    hc = hb;  pb = pb->next;
}

                    
while ((pa != NULL) && (pb != NULL))
{
    if (pa->data < pb->data)
    {
                    ;  pc = pa;   pa = pa->next;
    } else {
                    ;  pc = pb;   pb = pb->next;
    }
}
if (pa != NULL)  pc->next = pa;
else  pc->next = pb;
```

4. 例2：多项式相加

DATA STRUCTURE & ALGORITHM

给定一个n次的多项式$P_n(x)$和一个m次的多项式$Q_m(x)$，求他们的和。

比如：

$$P_{17}(x) = 7 + 3x + 9x^8 + 5x^{17}, \quad Q_8(x) = 8x + 22x^7 - 9x^8$$

则和为$7 + 11x + 22x^7 + 5x^{17}$

INPUT

第一行：两个整数a和b，表示$P_n(x)$共有a项，$Qm(x)$共有b项。（$1 \leq a, b \leq 1000, 0 \leq n, m \leq 1000000$）

第二行给出了多项式$P_n(x)$：共a对整数，每一对整数的第一个表示系数，第二个表示指数。系数值不超过100。

第三行给出了多项式$Q_m(x)$：共b对整数，形式同$P_n(x)$。

两个多项式都是以指数递增的形式给出。

OUTPUT

多行，每行两个数：系数和指数，以指数递增的形式输出，系数为0时无需输出。

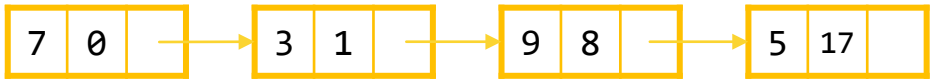
4. 例2：多项式相加

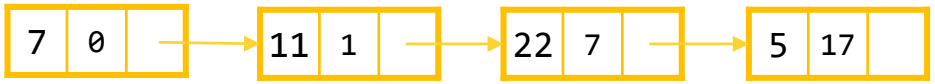
DATA STRUCTURE & ALGORITHM

- 项数：最多1000项；指数：最高1000000
- 用链表模拟

$$P_{17}(x) = 7 + 3x + 9x^8 + 5x^{17}$$

$$Q_8(x) = 8x + 22x^7 - 9x^8$$






```
struct node
{
    double coef;   //系数
    int exp;       //指数
    node * next;
} * p, * q;
```

4. 例2：多项式相加

DATA STRUCTURE & ALGORITHM

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
using namespace std;
struct node {
    int coef;
    int expon;
    node * link;
} a, b;
int main() {
    int n, m;
    node *ha, *hb, *hc;

    cin >> n >> m;
    ha = create(n);
    hb = create(m);
    hc = padd(ha, hb);
    print(hc);

    return 0;
}
```

```
node * create(int n)
{
    node *head = NULL, *rear;
    for (int i=0; i<n; i++) {
        node *q = new node;
        cin >> q->coef >> q->expon;
        q->link = NULL;
        if (head == NULL) {
            _____
        } else {
            _____;
        }
        rear = q;
    }
    return _____
}
```

4. 例2：多项式相加

DATA STRUCTURE & ALGORITHM

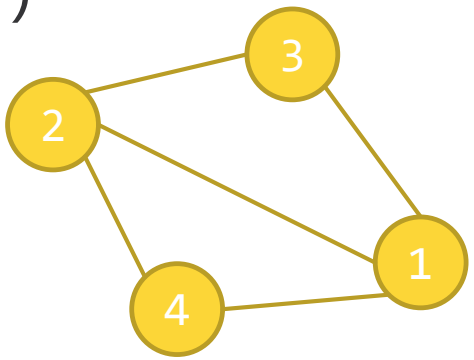
```
void attach(int coefficient,
            int exponent, node * &ptr)
{
    node * q = new node;
    q->coef = coefficient;
    q->expon = exponent;

    _____

    _____
}
void print(node *hc)
{
    node *rear = hc;
    bool first_node = 1;
    while (rear) {
        cout << rear->coef << " "
             << rear->expon << "\n";
        rear = rear->link;
    }
}
```

```
node * padd(node *a, node *b)
{
    node *c, *rear = new node, *tmp;
    int sum;
    c = rear;
    while (a && b) {
        if (a->expon > b->expon) {
            _____;
            b = b->link;
        } else if (a->expon == b->expon) {
            sum = a->coef + b->coef;
            if (sum) attach(sum, a->expon, rear);
            a = a->link;
            b = b->link;
        } else {
            _____;
            a = a->link;
        }
    }
    rear->link = a ? a : b;
    tmp = c, c = c->link, delete(tmp);
    return _____
}
```

4. 例3：小A的烦恼

DATA STRUCTURE & ALGORITHM

- 小A生活在一个神奇的国家，这个国家有N(N <= 100000)个城市,还有M(M <= 500000)条道路连接两个城市。道路连接的两个城市可以互相直接免费到达。小A比较烦恼，因为他想知道每个城市能直接到达哪些城市，你能帮帮他吗？保证每个城市都有道路与其连接（注:按照输入的道路顺序输出每个城市直接连接的城市）



- 输入：
- 4 5 //N，M
- 2 3 //2号和3号相连
- 3 1
- 1 4
- 2 4
- 1 2
- 输出：
- 3 4 2 //1号直接相邻的城市编号
- 3 4 1 //2号
- 2 1 //3号
- 1 2 //4号

$$O(n^2) \approx (10^5)^2 \approx 9GB$$

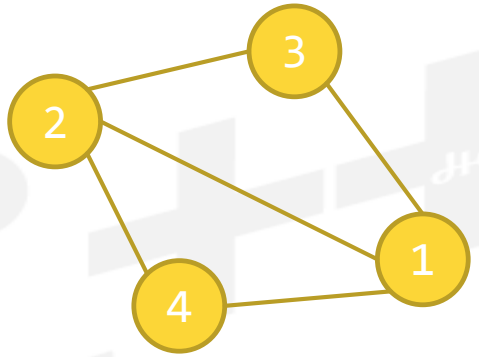
4. 例3：小A的烦恼

DATA STRUCTURE & ALGORITHM

i	1	2	3	4	5	6	7	8	9	10
city										
next										

i	1	2	3	4
head				
tail				

- 输入：
- 4 5 //N，M
- 2 3 //2号和3号相连
- 3 1
- 1 4
- 2 4
- 1 2



4. 例3：小A的烦恼

DATA STRUCTURE & ALGORITHM

```
const int N = 1e5+2;
int n, m, total = 0;
int Head[N], Tail[N];
int Next[N*10], City[N*10];

void add(int x, int y)
{
    City[++total] = y;
    if (Head[x] == 0) Head[x] = total;
    else ____________________;
    Tail[x] = total;
}
```

```
int main() {
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= m; i++)
    {
        int x, y;
        scanf("%d%d", &x, &y);
        add(x, y), add(y, x);
    }
    for (int i = 1; i <= n; i++)
    {
        for (int j = _______; _; ___________)
            printf("%d ", City[j]);
        puts("");
    }
    return 0;
}
```

练习

DATA STRUCTURE & ALGORITHM

- 1698: 线性表练习（1）
- 1699: 线性表练习（2）
- 1700: 线性表练习（3）
- 1701: 线性表练习（4）
- 2760: 合并珠子
- 2758: 维护序列
- 2761: 走出泥潭
- 1151: 多项式相加
- 3379: 小A的烦恼