1. 试以**表格**形式，给出串P所对应的next表及改进后的next表

j	0	1	2	3	4	5	6	7
P[j]	M	A	M	A	M	M	I	A
next[j]	-1			1	2	3	1	
improvedNext[j]	-1		-1		-1	3	1	

2. 简答：BTree::search()

B-树的查找过程中在每个节点内部，更适宜于采用哪种查找算法？为什么？

顺序查找

实际应用中B-树阶次m的设定，都能保证各节点内的关键码向量可整体容纳于一个对换页面之中，此时顺序查找反而比二分查找等算法更快

3. 简答：d-ary Heap

随着分叉数d的增加，多叉堆insert()、delMax()接口的时间复杂度将会如何变化？为什么？

insert()的时间决定于上滤成本 = log(d,n)，逐渐下降

delMax()的时间决定于下滤成本 = d * log(d,n) = d*log(d,2) * log(2,n)，逐渐上升

（只需考虑2 << d，d=3不必在意）

4. 证明：Vector::deduplicate

试证明，无序向量的去重算法（讲义122页，此处略）是正确的。

只需证明该算法具有如下不变性、单调性。

首先，可数学归纳证明如下**不变性**：

A）【不重】前缀[0,i)中不含重复元素；

B）【不漏】原向量中每一元素，在当前向量中都有至少一个副本

归纳基：初始[0,i=1)只含单个元素，A自然成立；尚无元素被删，B也成立

设不变性一直保持到i-1，现考查接下来i对应的那步迭代

无非if/else两种可能：

If，亦即[i]未在[0,i)中出现

随着i++，前缀扩充一个单位，A依然成立；未删除元素B亦保持

else，即[i]在[0,i)中出现（恰一次）

删除[i]后前缀不变，A保持；被删除者在前缀中仍有副本，B亦成立 [QED]

另一方面，不难验证该算法的**单调性**：

无论IF或ELSE，后缀的长度（n-i）都会减一

5. 算法：Top 10

试设计一个算法，用$\mathcal{O}(n)$的时间从互异且可比较大小的任意n个元素$\{ x_1, x_2, x_3, \ldots, x_n \}$中找出最小的$\lfloor n/10 \rfloor$个。

用**伪代码**描述你的算法，并证明时间复杂度符合要求。

（课程中讲授过的算法可直接**引用**，不必重复描述细节。）

```
Input: a set of n comarable elements: S = { x1, x2, x3, ..., xn }
output: the subset of the n/10 minimums
Algorithm Top10
  p = LinearSelect(S, n, n/10)  //Find the n/10 pivot by LinearSelect
  For each x in S  //Classify all elements into L and G w.r.t. p
    If x < p, output x  //And output those in L
```

课上已证明，只要Q值选取得当，LinearSelect便可在线性时间内完成；而Classification无非一趟扫描，亦不过O(n)时间。